



Lancaster University

School of Computing and Communications

Monitoring, Analysing and Predicting Network Performance in Grids

Yehia El-khatib

M.Sc., Lancaster University, 2006

B.Sc., Alexandria University, 2002

Thesis submitted for the degree of

Doctor of Philosophy

September 2011

Abstract

Monitoring, Analysing and Predicting Network Performance in Grids

by Yehia El-khatib

M.Sc., Lancaster University, 2006

B.Sc., Alexandria University, 2002

Thesis submitted for the degree of *Doctor of Philosophy*

September 2011

School of Computing and Communications

Lancaster University

The grid computing paradigm has facilitated the instrumentation of complex, highly-demanding collaborative applications. The technologies that make grid computing possible have mostly evolved from parallel and cluster systems. Although this has certainly empowered the grid computing field, part of the heritage has been the perception that required network resources are taken for granted. This is precarious, considering that most grids rely on public IP networks, like the Internet, as the underlying network. This assumption has obstructed the path of grid computing.

This thesis aims to improve the performance of grid applications by facilitating network-aware grid scheduling. This is achieved by providing network performance information to grid schedulers, allowing them to adapt to changes in the network. The contribution of this thesis is twofold: a novel approach to network measurement that is particularly suitable for grid environments; and a distributed system that collects and manages these measurements, predicts future network performance, and disseminates this information to schedulers.

The accuracy and effectiveness of this system is evaluated on a production grid infrastructure used for e-science applications. The outcomes of this evaluation provide a strong argument for the introduction of network-aware grid schedulers, information systems, and job and resource description standards.

Declaration

I declare that the work in this thesis is my own work and has not been submitted either in whole or in part for the award of a higher degree elsewhere. Any sections of the thesis which have been published are clearly identified.

Thesis originally submitted on 29 October 2010

Recommended corrections submitted on 19 September 2011

.....

Yehia El-khatib

© Copyright by Yehia El-khatib.

All rights reserved.

Acknowledgements

Working for the past 4 years on my PhD has been a remarkable experience. It has definitely not been an easy ride; nay, it was amply embellished with various challenges that prompted endurance and self discovery. I am pleased with its outcome, one that would have not been possible without the help of many people. To them I am indebted.

I am very grateful to Chris Edwards for his endless support, patience, and trust. He has been a wonderful advisor, line-manager, and friend. He struck a critical balance between giving me enough space to pursue my goals, and providing experience to guide my path. I learned from him how to ask the right questions, stay focused, and not shy away from any challenges. He charged me with confidence and optimism at times when I bitterly needed them. Chris is also an exceptional individual that I have greatly enjoyed being around.

I was privileged to have David Hutchison as a second supervisor. His insights have been enlightening and immensely helpful.

I am very thankful to my examiners Geoff Coulson and David De Roure for their feedback and for one of the most enjoyable discussions I've had so far.

I am grateful for the funding provided by the European Commission through the EC-GIN project that allowed me to carry out my research. Working on EC-GIN has been an enriching and enjoyable experience. I had the chance to meet and collaborate with some wonderful people on work that I feel passionate about.

I acknowledge the use of the UK National Grid Service and its applications in carrying out parts of this work. I thank the NGS helpdesk, and in particular Jonathan Churchill and John Kewely, for their support.

I am grateful to Gordon Blair, Dimitrios Pezaros, François Taïani, and Michael Welzl for discussing my work. I thank Alistair Baron, Panagiotis Georgopoulos, and Gareth Tyson for their feedback on earlier versions of this thesis. I am especially thankful to Gareth for his helpful input, coffee-break brainstorming, and random jokes.

I thank Ammar Oukil, Andreas Mauthe, Bandar Al-Hejin, Emad Abdulatif, Imran Tahir, Kashif Munir, and Muhammad Murtaza Yousaf for their advice and encouraging words. I also thank Barbara Hickson for calmly dealing with the trouble I seemed to constantly throw her way.

I am thankful to everyone in the Computing Department for the stimulating experience of working as part of a group of competent, friendly, and helpful professionals.

I am very grateful to my siblings, Mona, Radwa, and Mohammad, for their unconditional love and continuous support. They are a great blessing, and each has inspired me in many different ways.

I am incalculably indebted to my parents for their unique love and invaluable lessons that guide me every day. I am also very thankful for their unwavering belief in my abilities.

I am unable to thank my wife, Mariam, enough. She was with me through every step and I am grateful for her understanding, support, encouragement, and bearing with me and lifting my spirits during my most difficult times. None of this would have been possible without her by my side.

Last, but certainly not least, thanks to little Hajar for the sheer joy she brings.

Yehia El-khatib

Lancaster, 19 September 2011

إلى أُمِّي و أبي

*To my parents, Asma' and Sa'id,
who nurtured the perfectionist realist in me,
keeping my head in the clouds but my feet firmly on the ground.*

Contents

Contents	vi
List of Acronyms	xii
List of Figures	xv
List of Tables	xvii
I Introduction	1
1 Introduction	2
1.1 Background	3
1.1.1 Grid Computing	3
1.1.2 Resource Management	4
1.1.3 Scheduling	5
1.2 Network-Aware Scheduling	7
1.3 Critique of Current Grid Monitoring Systems	8
1.4 Research Goals	9
1.5 Thesis Organisation	9
II Background & Related Work	12
2 Network Measurement	13
2.1 Background	13
2.2 Network Metrics	14
2.2.1 Delay	15
2.2.2 Capacity & Bandwidth	16
2.2.3 Loss	17

2.3	Measurement Approaches	18
2.4	Measurement Tools	19
2.4.1	Delay	20
2.4.2	Bandwidth	22
2.4.3	Loss	25
2.4.4	Passive Measurement	25
2.4.5	Control Monitoring	26
2.4.6	Closing Remarks	27
2.5	Discussion	28
2.5.1	Intrusiveness	28
2.5.2	Reliability	29
2.5.3	Accuracy	29
2.5.4	Integration	30
2.6	Summary	30
3	Distributed Resource Management	32
3.1	Resource Management in Distributed Systems	32
3.1.1	The Origins of Distributed Systems	33
3.1.2	Homogeneous Distributed Systems	34
3.1.3	Heterogeneous Distributed Systems	38
3.2	Resource Supervision in Grids	39
3.2.1	Challenges of Resource Supervision in Grids	40
3.2.2	Standards	41
3.2.3	Tools	42
3.3	Summary	54
4	Requirements	56
4.1	Taxonomy of Grid Applications	56
4.1.1	Computational Grids	56
4.1.2	Data Grids	57
4.1.3	Service Grids	58
4.2	Grid Application Network Characteristics	58
4.2.1	Data Transfers	59
4.2.2	Control Messaging	59
4.2.3	Time Urgency	61

4.2.4	Distribution	61
4.3	Suitability of Current Grid Monitoring Solutions	62
4.3.1	Information Collection	64
4.3.2	Information Management	67
4.3.3	Information Dissemination	67
4.3.4	Information Consumption	68
4.4	Requirements for Network-Aware Grid Scheduling	68
4.5	Summary	71
 III Proposed Solution		 73
 5 Design		 74
5.1	Overview of the GridMAP Solution	74
5.2	Pasinemda	75
5.2.1	Rationale	75
5.2.2	Measurement Technique	76
5.2.3	Potential Advantages	79
5.3	The GridMAP Service	79
5.3.1	Overview	79
5.3.2	Architecture	80
5.3.3	Integration	82
5.3.4	Potential Advantages	84
5.4	Functions & Features	85
5.4.1	Performance Measurement	85
5.4.2	Data Accretion	87
5.4.3	Data Management	88
5.4.4	Information Publishing	89
5.4.5	Information Consumption	89
5.5	Summary	90
 6 Implementation		 91
6.1	Performance Measurement	91
6.2	Data Accretion	93
6.3	Data Management	94
6.4	Information Publishing	96

6.4.1	Service Invocation	96
6.4.2	Data Analysis	97
6.4.3	Dissemination	101
6.5	Information Consumption	101
6.5.1	Source Code Modification	102
6.5.2	Runtime Modification	102
6.5.3	Wrapping	103
6.6	Summary	103
IV Evaluation		105
7	Evaluating Pasinemda	106
7.1	Evaluation Questions	106
7.2	Methodology	107
7.3	Results	108
7.3.1	Round Trip Time	108
7.3.2	Throughput	114
7.4	Outcomes	114
7.5	Summary	118
8	Evaluating GridMAP	119
8.1	Evaluation Questions	119
8.2	GridMAP Deployment	120
8.2.1	Pasinemda	120
8.2.2	GridMAP Service	120
8.2.3	Induced Load	121
8.3	Methodology	121
8.3.1	Grid Testbed	122
8.3.2	Grid Applications	125
8.3.3	Implementing GridMAP in the Context of the NGS	130
8.3.4	Evaluation Metrics	132
8.3.5	Methodology Summary	135
8.4	Comparing the Performance of gLite WMS and gSched with GridMAP .	136
8.5	Analysis of Impact on Network Utilisation & Scheduling Performance .	136
8.5.1	Experiment A: AutoDock	137

8.5.2	Experiment B: Amber PMEMD	139
8.5.3	Experiment C: LogStore	140
8.5.4	Findings	141
8.6	Analysis of Impact in Alternative Network Environments	142
8.6.1	Rationale	143
8.6.2	Results	144
8.7	GridMAP's Overheads	146
8.7.1	Storage Overhead	146
8.7.2	Computational Overhead	148
8.7.3	Network Overhead	149
8.7.4	Findings	150
8.8	Outcomes	150
8.9	Summary	152
V	Conclusion	153
9	Conclusion	154
9.1	Thesis Summary	154
9.2	Revisiting the Requirements for Network-Aware Scheduling	156
9.3	Contributions & Implications	157
9.3.1	Revisiting the Research Goals	158
9.3.2	Implications	159
9.4	Future Work	160
9.4.1	Horizontal Expansion	160
9.4.2	Vertical Expansion	162
9.4.3	Autonomy	162
9.5	Future Trends	162
	Bibliography	165
A	Rise of the Grid	198
A.1	Beginings	198
A.2	Metacomputing	199
A.3	Distributed Communities	200
A.4	Standards	202
A.5	Service-Oriented World	202

B	The National Grid Service	204
C	gLite WMS	206
C.1	Architecture	206
C.2	Job Description	207
C.3	Scheduling Process	208

List of Acronyms

Acronym	Description	Defined in/on	
		Section	Page
ACK	Acknowledgement Packet	2.4	19
Amber	Assisted Model Building with Energy Refinement	8.3.2	128
AMP	Active Measurement Project	2.4.1	20
APAN	Asia-Pacific Advanced Network	3.2.3	50
API	Application Programming Interface	4.1.1	57
Argus	Audit Record Generation and Utilization System	2.4.5	27
AS	Autonomous System	2.4.1	21
BDP	Bandwidth Delay Product	2.2.3	17
CV	Coefficient of Variation	7.3.1	113
DIANA	Data Intensive and Network Aware	3.2.3	53
DIT	Directory Information Tree	3.2.3	44
DOA	Distributed Object Architecture	A.3	201
DRS	Data Replication Service	6.3	95
EGA	Enterprise Grid Alliance	A.3	202
EGEE	Enabling Grid for E-science	4.3.1	66
EGI	European Grid Infrastructure	9.5	163
FAFNER	Factoring via Network-Enabled Recursion	A.2	200
FLOPS	Floating point Operations Per Second	3.1.2	36
Gfarm	Grid Datafarm	6.3	95
GGF	Global Grid Forum	A.3	201
GIIS	Grid Index Information Service	3.2.3	43
GIS	Grid Information Systems	1.1.3	7
GLUE	Grid Laboratory Uniform Environment	3.2.2	41
GMA	Grid Monitoring Architecture	3.2.2	41
GridMAP	Grid Monitoring, Analysis and Prediction	5.1	74
GRIP	Grid Information Protocol	3.2.3	43
GRIS	Grid Resource Information Service	3.2.3	43

Acronym	Description	Defined in/on	
		Section	Page
GRRP	Grid Resource Registration Protocol	3.2.3	43
GSA-RG	Grid Scheduling Architecture Research Group	3.2.2	42
GSI	Grid Security Infrastructure	B	204
GT	Globus Toolkit	1.1.1	3
HPC	High Performance Computing	4.1.1	57
HTC	High Throughput Computing	4.1.1	57
ICMP	Internet Control Message Protocol	2.4.1	20
IGI	Initial Gap Increasing	2.4.2	24
IPDV	IP Packet Delay Variation	2.2.1	15
IPFIX	IP Flow Information Export	2.4.5	27
JDL	Job Description Language	C.2	207
KPI	Key Performance Indicator	2.2	14
LDAP	Lightweight Directory Access Protocol	3.2.3	43
LDIF	LDAP Data Interchange Format	3.2.3	44
LCG	LHC Computing Grid	1.1.1	4
LFT	Layer Four Traceroute	2.4.1	21
LHC	Large Hadron Collider	1.1.1	4
LRM	Local Resource Manager	8.3.1	124
MDS	Monitoring and Discovery System	3.2.3	43
MPI	Message Passing Interface	4.1.1	57
MRTG	Multi-Router Traffic Grapher	2.4.5	27
MSE	Mean Square Error	6.4.2	100
MTC	Many-Task Computing	4.1.1	57
MTU	Maximum Transmission Unit	2.4.2	23
NGS	National Grid Service	4.3.1	65
NGO	Network-based Grid Optimizer	3.2.3	51
NM-WG	Network Measurements Working Group	3.2.2	42
NWS	Network Weather Service	3.2.3	47
OASIS	Organization for the Advancement of Structured Information Standards	A.5	202
OeRC	Oxford e-Research Centre	7.2	107
OGF	Open Grid Forum	3.2.2	41
OGSA	Open Grid Services Architecture	1.1.1	4
OGSI	Open Grid Services Infrastructure	1.1.1	4
OS	Operating System	1.1.2	4
Pasinemda	Passive Network Measurement Daemon	5.2	75

Acronym	Description	Defined in/on	
		Section	Page
PE	Processing Element	3.1.1	34
PIC	Practical Internet Coordinates	3.2.3	52
PMEMD	Particle Mesh Ewald Molecular Dynamics	8.3.2	128
PoP	Point-of-Presence	4.3.1	65
PTR	Packet Transmission Rate	2.4.2	24
RAL	Rutherford Appleton Laboratory	8.6.1	143
R-GMA	Relational Grid Monitoring Architecture	3.2.3	50
RTD	Round-Trip Delay	2.2.1	15
RTFM	Realtime Traffic Flow Measurement	2.4.5	27
RTT	Round-Trip Time	2.2.1	15
Sander	Simulated Annealing with NMR-Derived Energy Restrains	8.3.2	128
SNMP	Simple Network Management Protocol	2.4.5	26
SOA	Service-Oriented Architecture	A.5	202
SOAP	Simple Object Access Protocol (originally)	1.1.1	4
SONoMA	Service-Oriented Network Measurement Archi- tecture	3.2.3	53
SRB	Storage Resource Broker	6.3	96
TOPP	Trains of Packet-Pairs	2.4.2	24
TTL	Time To Live	2.4.1	21
VO	Virtual Organisation	8.3.1	123
VLSI	Very-Large-Scale Integration	3.1.1	33
UDDI	Universal Description, Discovery and Integration	9.4.1	161
UDT	UDP-based Data Transfer	5.2.1	75
URI	Uniform Resource Identifier	5.3.1	79
WMS	Workload Management System	8.3.1	125
Wren	Watching Resources from the Edge of the Network	3.2.3	49
WSDL	Web Service Definition Language	1.1.1	4
WSRF	Web Services Resource Framework	A.5	202
XML	Extensible Markup Language	1.1.1	4
XPath	XML Path Language	A.5	203

List of Figures

1.1	Resource Management Functions	5
3.1	An Example of a Parallel Architecture	36
3.2	An Example of a Cluster Arranged Hierarchically	37
3.3	An Example of a Cluster Arranged Symmetrically	38
3.4	An Example of a Heterogeneous Distributed System	38
4.1	Histogram and Cumulative Distribution Function of Dataset Sizes	60
4.2	Grid Application Types in Respect to the Level of Parallelism and the Number of Processes	61
5.1	Closed-Loop Network-Aware Scheduling System	75
5.2	TCP Three-Way Handshake Process	77
5.3	GridMAP Architecture	80
5.4	Interactions of GridMAP Service with Pasiemda and Grid Scheduler	83
5.5	Query and Response Exchange between Scheduler and GridMAP Service	83
6.1	Pasiemda's Measurement Process	92
6.2	Measurement Database Entity-Relationship Model	95
6.3	Possible Extension to the Measurement Database Model	95
7.1	RTT Measured by Ping and Pasiemda on the Ethernet Connection	109
7.2	RTT Measured by Pasiemda on the DSL Connection	109
7.3	RTT Measured by Ping and Pasiemda on the OeRC Connection	110
7.4	RTT Measured by Ping and Pasiemda on the EMANICS Connection	110
7.5	RTT Measured by Ping and Pasiemda on the Innsbruck Connection	111
7.6	Ratio of the RTT Measurements of Ping and Pasiemda on the Ethernet Connection	111

7.7	Ratio of the RTT Measurements of Ping and Pasinemda on the OeRC Connection	112
7.8	Ratio of the RTT Measurements of Ping and Pasinemda on the EMANICS Connection	112
7.9	Ratio of the RTT Measurements of Ping and Pasinemda on the Innsbruck Connection	113
7.10	Values and Ratio of Throughput Measured by iperf and Pasinemda on the Ethernet Connection	115
7.11	Values and Ratio of Throughput Measured by iperf and Pasinemda on the DSL Connection	115
7.12	Values and Ratio of Throughput Measured by iperf and Pasinemda on the OeRC Connection	116
7.13	Values and Ratio of Throughput Measured by iperf and Pasinemda on the EMANICS Connection	116
7.14	Values and Ratio of Throughput Measured by iperf and Pasinemda on the Innsbruck Connection	117
8.1	The NGS' Islands of Grids Configuration	124
8.2	Scheduling Process using gLite WMS	131
8.3	Scheduling Process using gSched	132
8.4	An Example of Weighting Compatible Resources Using gSched	133
8.5	AutoDock Results With and Without GridMAP	138
8.6	Amber PMEMD Results With and Without GridMAP	139
8.7	LogStore Results With and Without GridMAP	140
8.8	An Example of Weighting Compatible Resources Using gSchedInv	144
8.9	Summary of the GridMAP Deployment Configuration	146
C.1	gLite Job Workflow	207
C.2	Job Cycle Using gLite WMS	208

List of Tables

3.1	Classification of Distributed Systems	35
6.1	Examples of Sample Weights used by the Exponential Smoothing Model	100
7.1	Summary of the Connections Used to Evaluate the Accuracy of Pasinemda	107
7.2	Percentage Difference Between Pasinemda and Ping Measurements	111
7.3	Standard Deviations of Pasinemda and Ping Measurements	113
7.4	Coefficients of Variation of Pasinemda and Ping Measurements	114
8.1	Evaluation Metrics	135
8.2	Summary of the Case Studies' Characteristics	136
8.3	Summary of the AutoDock Experiments Using gLite WMS and gSched	137
8.4	Summary of Amber PMEMD Experiments Using gLite WMS and gSched	137
8.5	Summary of LogStore Experiments Using gLite WMS and gSched	137
8.6	Summary of the AutoDock Experiment With and Without GridMAP	138
8.7	Summary of Amber PMEMD Experiment With and Without GridMAP	139
8.8	Summary of LogStore Experiment With and Without GridMAP	140
8.9	Summary of the AutoDock Experiment Using gSchedInv and gSched	145
8.10	Summary of Amber PMEMD Experiment Using gSchedInv and gSched	145
8.11	Summary of LogStore Experiment Using gSchedInv and gSched	145
8.12	Computational Costs of Pasinemda	148
8.13	Computational Costs of the GridMAP Service	149


Part I

Introduction

*Science must begin with myths, and with the criticism
of myths.*

KARL POPPER

Introduction

 GRIDS are made up of heterogeneous, geographically distributed resources that are not dedicated to any single user. Grid systems deal with this heterogeneity, high contention and lack of control over resources by finding out which computational resources best suit a job. However, no similar action takes place to determine network resources. On the contrary, the state of the network has been left out of the grid formula for a long time.

This thesis provides a case for network-aware grid scheduling. The thesis argues for the consideration of the network as a system resource, defines a set of requirements in order to attain network-aware scheduling, and proposes a solution to meet these requirements.

The remainder of this chapter is organised as follows. Section 1.1 outlines the concept of grid computing and explains the functions of resource management and scheduling. Section 1.2 discusses the significance of network-aware grid scheduling. Section 1.3 discusses the limitations of current grid monitoring systems with respect to achieving network-aware scheduling. Section 1.4 presents the research goals of the thesis. Finally, section 1.5 outlines the structure of the thesis.

1.1 Background

1.1.1 Grid Computing

A grid is a distributed system that is made up of a number of independent organisations or individuals that share their respective resources with each other. The aim of such federation could be to assemble enough collective computational power to tackle a highly complex problem, to share valuable resources, or to create a distributed collaborative community. Distilled from the literature, this definition of grid computing is the ‘traditional’ or most agreed upon definition [155]. In order to avoid confusion with other interpretations, it is important to affirm that the preceding definition is the one adopted in this thesis.

Grid computing followed parallel and cluster computing as distributed paradigms that capitalise on technological advancements in the fields of microprocessor design and networking. Compared to its predecessors, however, grid computing enabled autonomous, dissimilar computers of different organisations to interact in a loosely-coupled manner. This created potential for large scale collaboration over long distance networks. Unsurprisingly, the initial adopter of grid computing was scientific applications. This *e-science* scene provided use cases that presented significant challenges to grid computing, such as creating a distributed computing environment that enables collaboration across different administrative domains, architectures, platforms and languages; managing a large number of resources that are distributed both geographically and administratively; securing access to these resources through shared networks; and reliably transferring and replicating very large files. The grid computing community evolved significantly by embarking on these challenges.

In the following, we offer a brief survey of the technologies that enable grid computing. Appendix A gives a more detailed overview of the technological progression that both culminated in and further developed the grid computing paradigm.

The *grid stack* is a collection of software tools that enable grid computing. The principal element in this stack is the *middleware*, which provides solutions to the main challenges of grid computing such as intercommunication, resource management, job management and security. In doing this, the middleware creates a platform that conceals much of the complexity of dealing with the underlying infrastructure and allows the grid to behave like a single system. The Globus Toolkit (GT) [154] became the de facto middleware solution but was later followed by others, such as UNICORE [139] and gLite [234].

The stack may also contain other solutions that complement the middleware's functionality. Examples include data management solutions such as SRB [75] and GridFTP [57]; job managers such as Condor-G [162] and Nimrod/G [51]; and integration abstractions such as Cactus [58] and DataGrid [166]. The advanced distributed computing capabilities offered by such grid stacks facilitated a myriad of e-science projects, of which famous examples include the LHC¹ Computing Grid (LCG) [40] and SETI@Home [66].

The great interest in grid computing also resulted in the development of a large number of software tools and applications. In an effort to encourage interoperability and enhance usability of grid technologies [157, 156], the Web service-based Open Grid Services Architecture (OGSA) [158] and Open Grid Services Infrastructure (OGSI) [344] were put forward as standards for constructing service-oriented grid infrastructures. Using these standards, grid services need to be defined using the Web Service Definition Language (WSDL) and exchange information structured using eXtensible Markup Language (XML) over SOAP (originally an acronym for Simple Object Access Protocol).

1.1.2 Resource Management

The terms *resource* and *system resource* refer to any asset that a computer can use to facilitate the execution of applications. This includes physical resources such as processor cycles, memory space and hard disk capacity. The term also refers to logical resources such as files, interrupts and programs.

A number of operations are needed in order to enable applications, and hence users, to efficiently utilise different system resources. These operations are collectively referred to as *resource management*.

Resource management is as old as the computer. An *operating system* (OS) on a uniprocessor computer is required to implement mechanisms that facilitate the use of internal and peripheral resources. For example, an OS needs to detect and record its hardware resources. It also needs to obtain a means for interacting with each hardware device (i.e. a *driver*). The OS then needs to keep track of the usage and operation of these devices: What program has access to what device? When was this access granted? Is the current operation preemptive? Does the device appear to be operating properly? Are there any other programs requesting access to the same

¹The Large Hadron Collider (LHC) [39] at CERN.

resource? etc. These are examples of issues associated with managing a hardware resource, e.g. a hard disk drive. Although these issues have been thoroughly dealt with over the years, they still form the core set of issues that need to be addressed by any resource management scheme.

Resource management can be divided into three main functions, namely **Planning**, **Supervision** and **Control** [353]. This is illustrated in Figure 1.1.

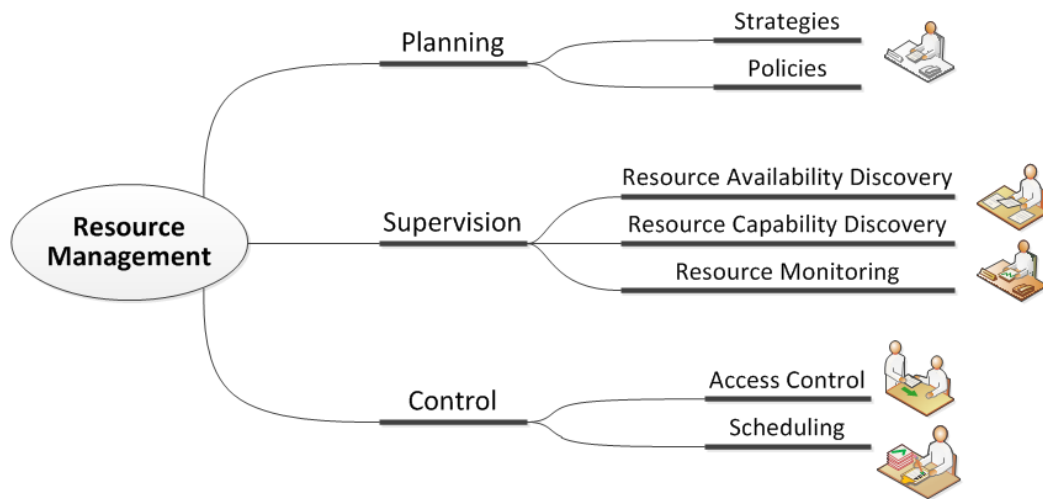


Figure 1.1: *Resource Management Functions*

Planning relates to long-term strategic arrangements and policy setting. The other two main functions relate to tactical management of the system. They are synonymous with the general functions of *resource state acquisition* and *resource state modification*, respectively. Supervision could itself be broken down to three functions: *resource availability discovery*, *resource capability discovery*, and *resource monitoring*. Control consists of functions that alter the state of resources. The most important control function is *scheduling*, which is discussed next.

1.1.3 Scheduling

Scheduling, a resource control function, is the process of coordinating the use of resources to execute a job so as to meet certain targets. In a distributed setting, such goals could be completion time, total amount of CPU cycles used, or fairness. In light of this, schedulers are also referred to as *resource brokers* since their fundamental duty is to oversee the execution of a job, ideally with the lowest possible costs.

From a purely theoretical viewpoint, the life of a grid job can be broken down into the following five phases.

1. **Matchmaking:** This phase consists of a number of tasks that need to be carried out by the scheduler to facilitate optimal job execution. The main three tasks here are *requirement validation* (determining required resources), *resource discovery* (learning resource availability) and *resource allocation* (mapping requirements to availability and expected performance).
2. **Submission:** This phase commences once the scheduler picks the resources to execute the job, i.e. the *execution theater*. Important processes of this phase are *mediation* (liaising with the manager of the execution theater), *staging* (transferring data files and parameter specifications to the execution theater), as well as *housekeeping* and *accounting*.
3. **Queuing:** Here, the job waits until the execution theater is available and set up for execution to commence.
4. **Execution:** This phase is the one where the job gets executed. All previous phases are carried out to ultimately reach an execution phase that satisfies the job requirements. The scheduler is also responsible to rectify any execution faults.
5. **Output retrieval:** This final phase involves transferring any output files, logs and error messages from the execution theater to a location where they can be referenced at a later time, such as user or shared space.

The manner in which these phases are implemented differs from one scheduler to another, as does the scheduler's range of responsibility. Many schedulers are responsible for administering and managing the job workflow on behalf of the user who submitted the job, covering all phases identified above. On the other end of the scale, some schedulers have a minimalistic orchestration role that only covers the matchmaking phase, in which case the responsibility of managing the workflow would be delegated to other components in the grid e.g the resource manager at the execution theater.

In any case, matchmaking is a scheduler's core function. This task of mapping jobs to resources relies on two important pieces of information. The first is the job requirements. This is defined either explicitly by the user submitting the job or in an automated fashion based on input parameters and pre-defined job profiles. The other vital information are the resource properties. This consists of their characteristics and status reflecting capacity and performance, respectively. Such information is

collected and made available by grid monitoring technologies commonly referred to as *Grid Information Systems* (GISs).

The performance of a scheduler is a measure of its ability to match jobs to resources that satisfy them. Given the relative ease of obtaining job requirements, a scheduler's performance hinges on its ability to identify the set of available resources and to accurately assess their capabilities. If the scheduler does not have accurate information about which resources are currently available, it will perform poor mapping and consequently inappropriate resource allocation. In other words, scheduling decisions can only be as good as the resource availability information provided to the scheduler [299].

1.2 Network-Aware Scheduling

Scheduling is essentially an optimisation problem. The scheduler is the element of the grid stack that is assigned the task of solving this problem. Thus, in order to fulfil its purpose, a scheduler needs to cooperate with GISs to be informed about the characteristics and operational state of system resources.

However, current schedulers do not consider the network as a system resource and hence do not cater for the network requirements of applications. At the same time, other grid technologies, such as middleware and GISs, provide extremely rich functionality for describing, discovering and monitoring computational resources, such as CPU and memory, but hardly anything about network resources. In brief, current grid technologies including schedulers are largely *network-oblivious*. Evidence of this will be given throughout chapters 2, 3 and 4.

This phenomenon is surprising considering the unpredictability of the networks used by grids. Moreover, the highly demanding nature of many grid applications, as will be investigated in chapter 4, creates high contention that can cause the amount of available network resources to fluctuate significantly. The reason behind this phenomenon lies in the heritage of grid computing, a topic that will be highlighted in chapter 3.

These circumstances call for accurate and fairly frequent updates about the state of the network to be provided to grid schedulers in order to allow them to adapt to changes in the network as they do to changes in the state of computational resources. Such adaptation to network state is referred to as *network-awareness*. Essentially, it implies considering the network as a system resource in need of management.

We argue that a network-aware grid scheduler is better at locating resources than a network-oblivious one. Consider, for instance, a job to perform an analysis procedure on a large dataset. It is considered important by any scheduler to locate a grid node with enough CPU power and memory space to satisfy the job requirements. A network-aware scheduler will also deem it important to choose a node reachable via a short network distance so that the dataset could be transmitted quickly. Such a scheduler is hence striking a balance between computational resource availability and network performance as resource allocation criteria. This enables the scheduler to choose resources that satisfy the user's job requirements as well as optimising overall performance.

In order to attain network-aware scheduling, certain changes need to be made to the matchmaking processes described in section 1.1.3. First, requirement validation needs to be revised to allow job descriptions to contain network requirements. Second, resource discovery should be amended to provide information about network performance. Third, resource allocation processes need to be adjusted to accommodate job network requirements and provide means of mapping between those and the performance of the network. These changes translate into increased complexity in three of the main grid technologies; namely job description conventions, GISs and schedulers, respectively. Amongst other things, this thesis aims to augment the second of these technologies by providing current and predicted network performance.

1.3 Critique of Current Grid Monitoring Systems

The importance of network-aware grid technologies has been increasingly recognised during recent years. A number of works have proposed ways of measuring and managing the network in the context of grid environments. The thesis critiques these solutions, highlighting a number of concerns.

There are four primary problems with GISs proposed in the literature. First, many of these solutions rely on network measurement techniques that suffer from high network overhead, poor accuracy or both. Neither of these is acceptable for grid systems as added network overhead further increases the high contention for network resources, and poor accuracy affects scheduling performance as already discussed. Second, many of them are quite heavyweight in that they require peer coordination. This poses an obstacle in the way of system-wide deployment, particularly in production grid infrastructures as it brings disruption to ongoing services. Third, the data

delivery mechanisms of some of these solutions are designed specifically for human consumption and are not suitable for schedulers. Finally, the architecture of many GISs prevents analysis of the collected information, and hence predictions of future network performance cannot be provided.

Detailed examination of current GISs is presented in chapter 4 after investigating the requirements of grid applications.

1.4 Research Goals

In order to address the identified lack of appropriate means to enable network-aware grid scheduling, the thesis posits three main arguments:

1. *There is room to improve grid scheduling by incorporating network performance into the resource allocation process.* This is based on the realisation that many grid applications are deployed over public network infrastructures, over which they have very little or no control. The contention experienced in such networks thus poses a performance bottleneck for these grid applications. The thesis also deals with grid applications deployed on over-provisioned networks.
2. *Network performance information could and should be accurately and reliably collected using passive techniques.* Such non-invasive conduct causes many of the overheads associated with network measurement to subside resulting in a solution that could be easily deployed and scaled without disrupting grid operations.
3. *Network performance information needs to be made available in a manner that avoids the shortcomings of existing GISs.* For instance, the collected information needs to be made available through one publisher to simplify information retrieval and avoid inconsistency problems between interdependent publishers. Another related weakness to be evaded is the use of a centralised architecture which is inherently prone to a single point of failure.

1.5 Thesis Organisation

The thesis is divided into 5 parts. The following gives an overview of each part.

- **Part I – Introduction** outlines the work of the thesis.

- **Part II – Background & Related Work**

Chapters 2 and 3 present and discuss work that is considered background and related to network-aware scheduling. This topic is covered by coarsely dividing it under two headings: *information collection* and *information dissemination*. Chapter 2 covers the former by exploring the vast field of network measurement, paying particular attention to techniques related to the one presented in this thesis. Chapter 3 then covers the latter heading, featuring a state of the art study of information systems used in resource management in grids.

Chapter 4 begins with describing the different types of grid applications and their requirements. This is used to prove the case for better information systems to allow network-aware grid scheduling. The chapter concludes by specifying a series of requirements for a solution to fill this void.

- **Part III – Proposed Solution**

The thesis argues that network-awareness is important to improve grid scheduling and proposes a set of requirements that ensure efficient network-aware scheduling. Part III proposes *GridMAP*, a GIS that satisfies these requirements. Linking back to the terminology of subsection 1.1.2, GridMAP aims to enrich Supervision functions as a means of enhancing one of the key Control functions, i.e. scheduling. GridMAP is designed to meet requirements which stipulate low overhead, easy deployment, interoperability, and ability to process and predict network performance.

Chapter 5 introduces the GridMAP architecture, presents its mechanisms and highlights the functions and features of its modular distributed design.

Chapter 6 details the manner by which GridMAP was implemented. This includes descriptions of four of the five main functions, and an overview of the implementation options for the final deployment-specific function.

- **Part IV – Evaluation**

This part evaluates the accuracy of the collected measurements as well as the impact of the network-awareness introduced by GridMAP.

Chapter 7 presents a set of experiments that were developed to assess the accuracy of GridMAP's passively-obtained network metrics against those obtained using well known network measurement tools.

Chapter 8 describes another set of experiments that were designed to evaluate the impact of using GridMAP on grid scheduling. This is achieved through using an advanced grid scheduler along with GridMAP to allocate resources for real-world e-science applications on a production grid infrastructure. As part of this evaluation, a particular adaptor was implemented to enable this scheduler to make use of GridMAP.

- **Part V – Conclusion**

Chapter 9 concludes the thesis by reviewing its contributions and comparing them to the desired requirements prescribed in chapter 4. Chapter 9 also comments on future work stemming from this thesis and future directions of grid computing in general.

Part II

Background & Related Work

Science is a lock; enquiry is its key.

AL-KHALIL IBN AHMAD AL-FARAHIDI

Network Measurement

NETWORK measurement is one of the most thriving research fields in networking. Over the past thirty years or so, a large host of work has been done on this topic. This chapter discusses some of this work. First, section 2.1 gives a background on the origins of network measurement and the motivation behind it. Then, two of the main aspects that are used to characterise network measurement tools are discussed, namely the measured metrics and the technique used to obtain them. These are covered in sections 2.2 and 2.3, respectively. Section 2.4 presents and discusses a multitude of network measurement tools proposed in the literature. Section 2.5 comments on the presented measurement techniques and tools. Section 2.6 concludes the chapter by commenting on the suitability of these tools in different situations.

2.1 Background

A computer network is made up of a number of different components that interact together in order to deliver a set of binary digits from one location (i.e. computer) to another. The fact that different independent components with different standards work together to make this happen, and the fact that the vast majority of networks are shared by much more than one pair of computers potentially owned and administered by different entities makes it a complex system. The manner of intercommunication of different components is governed by protocols and standards. However, due to the explosion of the Internet, networking standards have been independently

developed at such a rapid rate that it makes most improvement efforts out of date.

Such severe heterogeneity might have been one of the primary goals of designing networked systems, yet it creates an ample supply of operating problems that can be summarised as ‘unpredictable behaviour’. Moreover, post-deployment tweaking is quite important to Internet protocols and applications due to the virtual impossibility of modelling the Internet [151]. In a talk given in 1997 [203], Van Jacobson summarised these circumstances by saying:

The Internet has no central control or administration. In terms of ability to grow and scale this is a great strength. But in terms of ability to diagnose and fix problems it has been a serious weakness – packets usually cross many administrative boundaries on their way from a source to a destination and often the only point of agreement between those separate administrations is that all problems are someone else’s fault.

The number of network measurement tools that were developed in order to detect faults, gauge performance and tune networks is huge. This chapter will introduce a number of these tools that are of relevance to this thesis, but only after discussing the most common metrics and techniques used in evaluating network operations.

2.2 Network Metrics

A network *metric*, or *Key Performance Indicator* (KPI), is one of the network’s traits that describes its characteristic and operational state, or in other words its capabilities and performance.

Metrics can be used to describe either an end-to-end path or a single-hop link. Every metric indicates a particular property of the network or a certain aspect of its performance. Consequently, different applications have interest in potentially different metrics.

In order to avoid any confusion, this section introduces and defines the most commonly used network metrics, namely delay, capacity and loss¹, as well as their variations.

¹We consider packets with errors as lost.

2.2.1 Delay

End-to-end delay is very important to a lot of applications for many reasons. Some applications are sensitive to high levels of delay. In others, the overall application performance could be significantly affected by variations in the delay from one packet to another. For high-bandwidth applications, like many grid applications, high delay prevents long-term exploitation of the available bandwidth.

The three types of delay of interest here are the following.

One-way Delay

The *one-way delay* of a path is the latency introduced by the network in the delivery of a data unit from one end of the path to the other. One-way delay is an important metric of asymmetric paths in particular, where the qualities of the path in one direction might not match those of the opposite direction. Even in symmetric paths, the queues in the different directions of the path might have different loads.

Round-trip Delay

The *Round-trip Delay* (RTD) or *Round-Trip Time* (RTT) of any network path is the total time it takes to traverse the path from one end to the other and back. In other words, it is the amount of time from sending a data unit till the time its acknowledgement of arrival has arrived back to the sender. In that sense, RTT is very important to applications that use connection-oriented transport protocols. RTT is easier to measure than one-way delay: RTT can be easily measured using any of the tools that employ echo messages, while measuring one-way delay requires some sort of time synchronisation between sender and receiver.

Jitter

IP Packet Delay Variation (IPDV), commonly referred to as *jitter*, describes the variation between the delays of consecutive packets. Jitter reflects several network properties, most important of which is queueing delays within the network. Jitter is considered an important metric mainly by applications that rely on consistent packet delivery, such as multimedia streaming applications.

2.2.2 Capacity & Bandwidth

Bandwidth is used to indicate the amount of data that can be transferred across a particular network path per unit time. Bandwidth (and especially available bandwidth, described below) is an important metric to obtain. It is used in congestion control, peer selection, traffic engineering, accounting and QoS verification.

Four different types of bandwidth are described as follows:

Capacity

The *capacity* of a path is the maximum bandwidth it can offer, i.e. when no contention is present. Capacity information can be retrieved from routers. Capacity is sometimes defined as the maximum bandwidth that can be achieved at the IP layer. Such subtle difference in definition makes a huge difference as the value of the latter cannot be obtained from routers.

Utilisation

The *utilisation* of a path is the total amount of traffic sent by all flows currently using the path per unit of time. Single-hop utilisation levels can be easily obtained from networking devices and interfaces, but the utilisation of end-to-end paths is not always that straight-forward to obtain.

Available Bandwidth

Available bandwidth (sometimes referred to as just 'bandwidth') is used to refer to the residual path capacity that can be provided to the IP layer after considering the current contending cross-traffic. The amount of available bandwidth can be deduced by finding out capacity and utilisation information. It can also be obtained by direct measurements.

Achievable Bandwidth

Achievable bandwidth, or *achievable throughput*, is the amount of throughput an application can achieve out of the available bandwidth after considering the capabilities and limitations of the used protocols, operating system, and host utilisation level.

2.2.3 Loss

The delivery of packets sent across a best-effort network is not guaranteed and, hence, it is not unanticipated that a packet gets lost en route. A high level of packet loss is probably the most obvious indicator of poor network performance due to poor physical layer conditions, congestion, etc.

Packet loss is an important metric to study and monitor. High packet loss can diminish the throughput achieved by any application. As for TCP, even a low number of lost packets could significantly affect the throughput it achieves. This is because TCP is a mechanism that reacts similarly to both loss and congestion by being less aggressive. This characteristic creates an inverse proportional relationship between TCP efficiency and the degree of asymmetry of the path [232], as well as between TCP efficiency and the bandwidth delay product (BDP) of the path [74, 355, 220]. This affects satellite, wireless and ADSL communication channels but perhaps its greatest punishment is reserved for applications that use ‘long fat pipes’ such as transatlantic connections. Poor handling of packet loss has a crippling effect on such applications.

Furthermore, some applications are much more sensitive to packet loss than others. For example, some multimedia streaming applications have low tolerance for the delay and loss of certain packets (e.g. part of key frames) as they lead to depreciated delivered media quality.

There are two main types of packet loss:

One-way Packet Loss Rate

One-way loss rate is the most commonly used metric when packet loss is of interest. Measurements of this metric indicate the percentage of packets that have been sent by the sender but not received by the recipient.

Two-way Packet Loss Rate

Two-way or *round-trip loss rate* specifies the percentage of packets that have been sent but with no corresponding acknowledgement received from the recipient. Such property might be misleading as the loss can take place in either path directions. It is thus difficult to tell the direction of the path in which the problem causing the packet loss is.

2.3 Measurement Approaches

There are different approaches to measuring network performance. One approach is *active measurement* where the network is probed to observe how the traffic is dealt with. This approach obligates the network to accommodate additional traffic in the form of probes in addition to real traffic, which potentially yields accurate metrics. Active measurement can be carried out at any of the different network layers, offering a variety of granularities: packet, flow, or network.

However, the overhead incurred by the artificial traffic decreases the overall performance of the network. Moreover, it could be argued that it also affects the accuracy of the measurement. We shall revisit this issue later.

Other approaches employ *messaging*, the most prevalent being *ICMP messaging* as it uses light-weight probes, is universally implemented and is relatively easy-to-use. Unfortunately, the accuracy and reliability of ICMP-based approaches is low due to the tendency of administrators to configure their networks to discriminate against ICMP traffic [356]. It is not uncommon for such traffic to be treated differently than other traffic (using rate-limiting, etc.) or to be blocked altogether (c.f. [179]). Other messaging approaches use UDP or TCP instead of or in conjunction with ICMP.

There are also a large number of tools that measure the performance of the network without injecting any traffic into it. These are known as *passive* tools, and they rely on calculating or estimating network metrics based on observed traffic. Although this approach is able to obtain an abundant amount of information, such information is usually limited in descriptive capacity. It is for this reason that metrics obtained passively are sometimes inaccurate. Just like active measurement, passive measurement can be done at any network layer.

Control monitoring or *network traffic accounting* is a passive approach where a host utilises the information gathered by networking devices, such as routers, to obtain knowledge about the network and its performance. Such information could be used to build a fairly complete status of the network environment which can then be utilised to adapt host or network device configuration, or for accounting. Different protocols have been developed to secure access to, and to enrich such information.

2.4 Measurement Tools

The performance of networks has vastly improved over the years with advancements in telecommunication technologies: communication media could accommodate more traffic; switching technology improved enabling the processing of more frames per second; network protocols became more efficient, etc. These advances have encouraged the development of more network-dependent applications. Computers connected to the network became increasingly greedy for the network. Eventually, increased contention caused network performance delivered per application to deteriorate and become quite unpredictable.

This situation was the motivating influence behind two very important changes in networking. First, communication protocols and standards became more sophisticated, adapting themselves to the state of the network. Two popular examples are CSMA/CD and TCP congestion control. The former uses very low-level techniques to detect a situation of possible frame collision on an Ethernet segment, while the latter uses acknowledgement packets (ACK) to appropriately throttle data transfers. In both cases, a closed-loop feedback system is employed to alter the way the network is used in reaction to changes in the state of the network.

The second important change was the need for advanced network measurement techniques. Users started realising that the networks, more often than not, operated differently to expectations. In order to understand how networks behave and operate, and subsequently investigate why they do, networks had to be measured in various ways. Numerous efforts have been made over the years to study the different properties of the network under different operating conditions. These efforts are useful not just to fine-tune network protocols and applications, but also to check, by providers and users alike, whether service levels are being maintained or not.

Tools that have been developed for measurement of network performance vary in many ways, but mainly they vary in the metrics they try to obtain and in the technique they use to obtain them. Every different technique requires certain software and/or hardware capabilities, has a certain degree of complexity, achieves a certain level of accuracy and possibly has certain side-effects. These are regarded as the requirements of the tools and the expected quality of their output, and they collectively define the suitability of any tool for use in a particular situation. This set of tool characteristics, however, can be very scrupulous to an extent that makes it difficult to produce a taxonomy based on them.

Instead, this section provides an overview of network measurement tools, initially being categorised based on the main metric they measure. Nevertheless, there is no distinct separation between tools that measure each different network metric as most tools measure a group of different metrics. Therefore, tools will be mentioned for each metric in chronological order. If a tool needs to be mentioned again under another metric, it will only be referenced with no further details. We then turn attention to passive measurement and control monitoring by commenting on some of the tools of these approaches.

2.4.1 Delay

Delay measuring tools can be categorised according to their technique: *ICMP Messaging*, *UDP Messaging*, *TCP Messaging* and *Packet Dispersion*. The review of Packet Dispersion is deferred to the following subsection about bandwidth measurement.

The Internet Control Message Protocol (ICMP) is a control protocol designed as part of the TCP/IP suite [86] to provide information about network congestion and errors, and to allow route manipulation albeit to a very limited degree. Accordingly, ICMP messaging was among the first network measurement techniques and is indeed the most universal.

Ping was initially developed to check the availability of a path to a remote host. ECHO_REQUEST, an ICMP message, is issued to the target host which upon receipt would reply with ECHO_REPLY, another ICMP message. The latency of the path can be measured by timing the delay between sending the former message and receiving the latter. The loss rate is calculated to be the ratio of unanswered to the total number of messages. By repeating this procedure, one can obtain the minimum, maximum and average RTT as well as the packet loss ratio.

The ping program was soon integrated in almost all operating systems for its simplicity and efficiency. It still remains the most prominent tool for obtaining RTT estimates.

Several tools have been based on ping, such as **fping** [130] and **Badabing** [312]. Other efforts have developed measurement infrastructures that employs ping. An important one is **Pinger** [256] which is a light-weight distributed tool that employs ping to acquire estimates of RTT, loss rate and jitter between a number of HPC nodes in almost real-time. Other measurement projects that work in a similar manner include **Surveyor** [214], the Active Measurement Project (**AMP**) [41], and **RIPE** [37].

As the Internet grew, so did the number of users and operators. Eventually, there were major routing problems due to inconsistencies in boundary router configurations and the exponentially growing BGP routing table size [194]. It became evident that isolating the characteristics of each separate hop would help in identifying routing problems. By utilising batches of IP packets with incrementing Time To Live (TTL) values, **traceroute** [202] produces a hop-by-hop analysis of the path to a remote host by keeping record of each router that responds with an ICMP ‘Time Exceeded’ message. In this manner, traceroute could be used to find troublesome hops in a route so that they can be rectified independently. It also allows drawing a virtual network topology. By default, traceroute uses UDP datagrams, although it can be configured to use ICMP type 8 echo requests. Further work on traceroute includes **AS^{II}-level traceroute** [250] and **Paris traceroute** [70] which aim to eliminate traceroute anomalies, along with **tcptraceroute** [339] and Layer Four Traceroute (**LFT**) [42] that are TCP implementations of traceroute.

More details about each hop can be made available using some traceroute-based tools such as **Pingroute** [115] and **Traceping** [44]. Beside the output of traceroute, the Pingroute script provides the RTT and loss rate for each individual hop. Traceping provides very similar output but over different periods of the day to help notice any reoccurring problems such as spikes in packet loss.

The output of periodic traceroute probes along with that of ping probes and DNS queries could be used to create visual depictions of network paths and/or nodes. Several tools have used this ‘geographical traceroute’ approach, including **GeoBoy** [13], **GTrace** [276], **NetGeo** [263], **NeoTrace** [20], **VisualRoute** [46], **WhatRoute** [107] and **Xtraceroute** [71]. It should be noted that parts of the network where ICMP is unsupported will not appear in such topology.

TCP messaging is a passive technique in which TCP behaviour is observed to extract network metrics. In other words, TCP exchanges, including signalling using TCP flags, are used to calculate metrics such as RTT and loss. This can be done either in real-time by capturing packet headers or post-mortem by looking at traffic traces. As this involves the TCP stack, the obtained metrics give a true transport-level assessment of the treatment of packets in the network. The TCP messaging technique has been used for various purposes such as inferring path characteristics, inferring and adjusting congestion window size, traffic engineering and traffic modelling.

Monitoring TCP handshakes in particular, a technique we refer to as *TCP-ping*,

^{II}Autonomous System

was first used by **Sting** [296] to obtain bi-directional loss rates. **tcping** [129] uses TCP-ping to select servers based on the least observed latency. The authors of [253] explain how TCP handshake packets could be used to obtain a good estimate of end-to-end RTT, while [190] as well as IEPM's **Synack** [36] applied the technique and reported on its advantages and disadvantages. [210] also adopted TCP-ping in addition to monitoring in-flow packets. There are further examples, such as [207, 233, 356, 244].

Passive TCP monitoring is used for purposes beyond replacing ping as a means of calculating RTT and loss. [59] monitored successful transmissions of in-flow TCP packets to control HTTP sessions and to produce realistic client-server delay models. **Benko and Verdes** [79] uses passive observations of TCP traffic from routers to estimate packet loss. **tulip** [249] uses the same approach to analyse packet delay, loss and reordering. Similarly, [195] relies on en route monitoring points to observe TCP flows and detect imminent congestion. **TCP Sidecar** [305] injects measurement packets into real TCP traffic flows, which is an added overhead and requires peer collaboration. [365, 366] analyses TCP kernel traces to optimise peer selection. **abget** [68] estimates available bandwidth by analysing delay variation in an artificial TCP stream, while [99] analyses the dispersion of artificial TCP packet pairs.

TCP messaging has proven to be reasonably accurate for measuring TCP connection properties as well as being significantly more reliable than ICMP- and UDP-based techniques [219, 150]. There are, however, arguments against observing TCP traffic to learn about the network. For instance, [54] argues that RTT can vary significantly through the lifetime of a TCP session. Moreover, TCP-ping techniques that use artificial TCP-SYN probes, such as [190], raise alarms of incipient TCP-SYN floods [131] and Denial-of-Service attacks [169]. This reduces the techniques reliability.

2.4.2 Bandwidth

The techniques to measure bandwidth can be divided into three types: *Link Saturation*, *ICMP/UDP Messaging* and *Packet Dispersion*.

Link saturation techniques measure available bandwidth by persistently sending as much traffic as possible. This is done by setting up either TCP or UDP flows between two end hosts, and obtaining the amount of bandwidth that would be available at the application level. In addition, jitter and packet loss can also be calculated by observing the injected traffic.

Such intrusive brute force approach of saturating the path causes deteriorated

performance for other flows that share one or more portions of the same path. Despite its disruption, the link saturation approach has been extensively used for its reasonable accuracy in providing three metrics that are of interest to many applications: available bandwidth, jitter and packet loss rate. The most notable link saturation tools are **TTCP** [45], **nuTTCP** [27], **Iperf** [338], **Nettest** [23], **NetPerf** [212], **NetSpec** [213], **Imbench** [259] and **UDPmon** [192].

An alternative approach to link saturation is hop-by-hop characterisation using ICMP/UDP messages. ICMP messages of varying TTL values are used to measure the delay and loss rate of each hop along the path, while UDP datagrams of increasing sizes are used to estimate the bandwidth. **Pathchar** [202] sends a series of packets with a random payload size. By isolating the metrics of each hop, Pathchar is able to determine the RTT and available bandwidth of each hop as well as drop rate at each router along the path. For Pathchar to work, it assumes that a probe and its corresponding reply go through the network with negligible queueing delays. Despite being slow, Pathchar has been quite successful as it allows the detection of network bottlenecks and their bandwidth. A number of other tools have been based on the Pathchar technique. For example, **pchar** [248] uses a number of repetitions with random packet sizes to get a more accurate measurement of path loss and queueing characteristics. **Clink** [127] is another tool that uses the Pathchar technique but sends more probes in order to improve accuracy when queueing delays cannot be ignored.

End-to-end IP capacity estimation is another area which has received considerable attention. **Keshav** [222] shows that such information, as well as bandwidth, can be calculated using inter-packet arrival rate. This is called *packet dispersion* or *packet-pair dispersion*. He states that one of the effects of network congestion is the extension and contraction of the space between the acknowledgements for consecutive packets. The dispersion of the packets sent at regular periods are analysed to derive a relationship with bandwidth, and with capacity. Keshav's groundbreaking work was based on pairs of packets. However, more recent techniques tend to use a chain of packets, known as *trains*, or probes over very short periods of time, known as *chirps* (suitable for large networks).

Pipechar [19] is a tool that performs per-hop measurement in order to detect bottlenecks in the network. It sends varied-sized packets to each router along the path and uses the relationship between delay and packet size to compute the available bandwidth and capacity of the whole path. However, Pipechar is not suitable for very fast network paths where the Maximum Transmission Unit (MTU) of 1500 bytes

results in small round trip delays which do not follow the employed relationship between delay and packet size.

The **Pathrate** tool [126] calculates the available bandwidth of a path using the dispersion of packets in packet pairs and trains. **SProbe** [295] and **CapProbe** [216] work in a similar way to measure end-to-end bandwidth. **AsymProbe** [103] is a CapProbe-based tool that measures IP capacity in both directions of a path by analysing the dispersion of packet-pairs. Other CapProbe-based tools include **AdHocProbe** [102], which measures end-to-end capacity of wireless paths, and **PBProbe** [101], which is used with high-speed paths. **Nettimer** [231] does hop-by-hop link characterisation and bottleneck detection through utilising the ICMP/UDP messaging technique of Pathchar and the packet dispersion one of Pathrate.

Pathload [206] estimates the range of available bandwidth based on the variation in one-way delays of a periodic packet stream. It attempts to reduce the effect of injecting so much traffic into the network by utilising only 10% or less of the available bandwidth. **abget** [68] provides similar output using artificial TCP streams, requiring no remote peer coordination.

pathChirp [291] uses chirp-trains instead of packet-pairs and packet-trains in order to increase accuracy [290]. The chirps in a chirp-train are spaced at exponentially decreasing periods which has proved to decrease the overall probing time on high-speed paths [307]. The algorithm estimates available bandwidth by gauging the rate at which a chirp congests the network.

Spruce [319] is another tool with statistically spaced probes. The 1500 byte packet-pairs sent by Spruce are Poisson-distributed but might be altered for the probing traffic throughput to meet the minimum of 5% of the path capacity. By measuring the change in inter-packet delay, Spruce estimates the bottleneck load and uses it to infer the available end-to-end bandwidth. However, it has been reported by Shriram et al. that Spruce is not very accurate when measuring high-speed paths [307].

Just like Spruce, Initial Gap Increasing (**IGI**) uses packet-pair probing to estimate the current bottleneck load [191]. However, IGI uses evenly-spaced packet-pairs. Packet Transmission Rate (**PTR**), designed also by Hu and Steenkiste [191], shares the same probing technique of IGI but directly calculates the amount of available bandwidth. Other examples of tools employing the packet dispersion technique are: **Trains of Packet-Pairs** (TOPP) [261] and **bprobe/cprobe** [94].

The reader is referred to [282] and [307] for further reading on the topic of bandwidth estimation.

2.4.3 Loss

Many of the aforementioned network measurement tools measure the magnitude of packet loss as well as delay and bandwidth. This includes delay tools (e.g. Synack, PingER, Pingroute), link saturating tools (e.g. Iperf, NetPerf) and ICMP-based tools (e.g. Pathchar, pchar). There are, however, both active and passive tools that concentrate mainly if not solely on measuring loss. We here review a selected few of these.

Being an active tool, **Badabing** [312] sends probes to measure packet loss but, in contrast with most active measurement tools, Badabing does so without using Poisson-compliant traffic probes. This achieves higher accuracy with less disruption to the network.

Benko and Verdes [79] introduced an algorithm to estimate packet loss rate based on TCP sequence numbers observed passively by en route network devices. The technique is particularly useful for network operators. The algorithm is, however, not accurate for short-lived flows [141].

PcktLoss [163] is a system designed for applications that run on high-speed network infrastructures. It calculates packet loss rate per flow by comparing the packet statistics collected by passive network monitoring points in the infrastructure. This approach obtains accurate results, but it assumes access to network metrics from all edge routers and it does not address asymmetric routing.

2.4.4 Passive Measurement

Thus far we have mainly discussed active tools that measure the performance of the network by disrupting it one way or another. Passive tools aim to avoid that. They can be divided into two main categories: *traffic analysis* and *packet collection* tools.

Passive traffic analysis tools are ones that monitor traffic looking for certain events or anomalies and trigger metric calculation accordingly. They are thus smarter, more selective about what to monitor, and are more specialised in what they measure. Examples include [243], [60], [207], [351], [365, 366] and [289] (along with **Benko and Verdes**'s work described in the previous subsection) where TCP headers and traces are used to infer metrics such as RTT, achieved throughput and packet loss. Other examples include the **TCP Sidecar and Passenger** [305] technique where TCP flows are used as a substitute for traceroute, and **multiQ** [221] where the packet-pair technique is applied to TCP traces in order to estimate bottleneck bandwidth.

Packet collection tools, commonly known as *packet sniffers*, record observed traffic by copying packet headers. This can be later used to replay the traffic for troubleshooting or load testing. They could also be used to calculate different performance metrics, including but not limited to throughput, RTT, packet loss, packet size, and port utilisation.

Packet sniffing can be performed using specialised hardware. However, software is much more commonly used as it is a cheaper and more flexible alternative. Examples of packet sniffing software packages include: **tcpdump** [204], **libpcap** [204], **libtrace** [47], **tcpflow** [138], **Pcapture** [32], **IPTraff** [208] and **SocketSniff** [311]. These tools keep logs of the sniffed (i.e. captured) packets and save them in what is referred to as ‘trace’ or ‘dump’ files. These dump files could then be compiled and analysed to closely inspect the observed traffic.

Packet sniffing can produce huge amounts of data in short periods of time. Such data would need to be filtered and perused in order to obtain useful statistics about the traffic. A simple way to retrieve a human-readable synopsis about the captured traffic is to feed a dump file to a summarising package such as **tcptrace** [270].

Another way to study the captured traffic is to graphically represent it and any information that can be drawn from it. There are numerous tools that accept different sorts of trace input to produce different visualisations and graphs. A lot of these tools are also capable of sniffing traffic themselves. The most common of these are **Wireshark** and its predecessor **Ethereal** [269] which are open source GUI-based packet-sniffing packages that provide added statistical functionality. Other packages include **BSOD** [48], the **NetBoy** suite [21], **Ntop** [124], and **PingPlotter** [33].

These tools, however, are developed primarily for the purpose of system administration and network troubleshooting. More meticulous examination of the characteristics of captured traffic could be achieved using generic statistical analysis packages, such as R [286], that provide even more advanced statistical tools.

2.4.5 Control Monitoring

The Simple Network Management Protocol (**SNMP**) [96, 97] offers a standardised method for remote access to information about routers, switches and other network devices. SNMP uses an extensible data structure to represent this information, which allows it to hold a wide range of metrics from running traffic statistics to room temperature and humidity.

Multi-Router Traffic Grapher (**MRTG**) [268] is an open-source package that collects traffic statistics from routers and other SNMP networking devices and uses them to plot graphs of traffic statistics and router performance. Similar efforts to visualise flow-level performance and discover any abnormalities include CAIDA's **CoralReef** [225] and QoSient's Audit Record Generation and Utilization System (**Argus**) [3].

The Realtime Traffic Flow Measurement (**RTFM**) architecture [87] was a standardisation effort to describe how a distributed set of SNMP devices could be employed to monitor flows. **NeTraMet** [88, 22] is an open-source implementation of the RTFM architecture.

The **NetFlow** protocol [6] was developed by Cisco to allow their routers to provide detailed statistics on all handled traffic. Such information is very useful not just for troubleshooting but also for accounting, traffic engineering, user profiling and security analysis. NetFlow grew to become a de facto standard for control monitoring; something that SNMP was originally designed to become but failed to do so because of its various shortcomings and vulnerabilities. However, the protocol was for a long time proprietary which induced similar efforts, such as **sFlow** [279] and **J-Flow** [7], to emerge. Concurrently, several tools were developed to accumulate NetFlow statistics for visualisation, analysis and publishing. These are called *NetFlow collectors* and examples include **flowd** [140], **NEye** [25] and **JNCA** [17]. The export format of version 9 of NetFlow was later made available in [111], upon which the IP Flow Information Export (**IPFIX**) Protocol [110, 285] open standard was based.

2.4.6 Closing Remarks

Finally, it is important to stress that this section does not serve as a comprehensive study of network measurement tools. This area is vast and stretches in many directions. For example, topology detection has not been discussed. Instead, this section merely presents an overview of network measurement techniques and tools that are of relevance to the topic of this thesis. For more information about network measurement in general, the reader may refer to [104], [298] and [5].

2.5 Discussion

IP networks are not designed to readily provide feedback about their performance^{III} despite the importance of network measurement. The plethora of network performance measurement techniques has covered different aspects of this problem. Yet these solutions have their drawbacks, some of which might not have been significant in the past but are so in the grid context. This section comments on four main shortcomings of these solutions: *Intrusiveness*, *Reliability*, *Accuracy* and *Integration*.

2.5.1 Intrusiveness

Various network measurement techniques probe the network to measure its metrics. Pathchar, for instance, is an active-probing tool that finds out network characteristics on a per-hop basis. Its variants, such as pchar and Clink, have an even more intrusive attitude to attempt to increase the accuracy of the measurements they acquire.

Probing corners the network to accommodate artificial traffic next to the real traffic. The network is also forced to treat both equally. This decreases the overall delivered performance. Of equal importance, every intrusive measurement technique has its own, sometimes quite significant, drawbacks. For instance, many link saturating techniques were not designed for high-speed networks and thus do not send probes large enough to force TCP to finish its slow-start phase. If the probe sizes are increased so that they would saturate the pipe, the overhead involved becomes quite significant.

Packet dispersion techniques are far less intrusive than other active techniques, but are limited in that they cannot distinguish between the metrics of the different directions of the path. If the reverse direction is, say, congested then the technique would conclude that the overall path is congested when it might not be so. Such an outcome can mislead applications where most of the traffic is going in one direction such as multimedia streams or bulk data transfers.

Finally, the accuracy of active network measurement techniques is sometimes overestimated [186]. We shall revisit this in subsection 4.3.1.

^{III}Although IPv6 headers provide an opportunity to develop such solutions (a good example is presented in [278]), the still prevailing [217] IPv4 does not.

2.5.2 Reliability

ICMP messaging techniques are unreliable as some network operators and system administrators configure their routers and hosts to disregard ICMP messages, especially those received at times of congestion [290]. In such situations, tools such as ping would give the false diagnosis that a remote host is unreachable.

Even if ICMP packets are allowed, they are more often than not treated differently as compared to TCP and UDP traffic (e.g. they are rate-limited). This makes any ICMP-based tool prone to giving incorrect metrics. For example, a congested router will queue if not drop most ICMP messages in order to give priority to server TCP and UDP flows. This would lead the loss rate calculated by ICMP-based tools to be much higher than it would really be for a TCP flow for example. This fundamentally affects the accuracy of ICMP-based tools. Take for instance Pathchar, on which many ICMP techniques are based on. Its accuracy hangs on obtaining a pair of messages (i.e. probe and error reply) that were hardly queued at all. This assumption is difficult to verify let alone meet on many networks [283], especially WAN paths [127].

2.5.3 Accuracy

In application scenarios with a large number of varying hosts, it is sometimes not feasible to measure all network traffic arriving at or leaving these hosts. Therefore, the majority of measurement tools designed for large distributed systems rely on projections to reduce the measurement overhead. This common technique is referred to as *network tomography* [349, 98]. In the following, we explain the adversary effects of tomographic tools on measurement accuracy.

There are two main forms of network tomography. *Global inference* uses measurements collected for paths between a subset of nodes to estimate metrics of paths between other edges of the network. *Fragmental inference* obtains traffic statistics about portions of paths from routers and other networking devices, and uses them to project end-to-end performance.

In either case, tomography produces questionable accuracy levels. In the case of global inference, this is because some of the measurements are real ones obtained directly from applying network measurement tools while others are estimations. Such estimations are of varying degrees of error due to the differences in correlation to the real measurements. In the case of fragmental inference, the control monitoring

information provided by routers and other network devices (usually via SNMP^{IV}) is rather limited in different ways. First, this information describes aggregate traffic traversing the particular networking device. This might be useful to administrators and network operators but is of little use to individual applications. Second, the properties of different portions of the path do not necessarily provide an accurate description of the overall path. Hence, this information only provides poor estimates of the end-to-end path performance. Furthermore, the quality of the core does not necessarily reflect that of the access network and vice versa [118].

Thus, tomography forsakes accuracy in order to control measurement overhead which in turn allows measurement systems to scale.

2.5.4 Integration

The majority of the measurement tools presented in this section are bespoke, expected to be triggered manually, and produce metrics in a format that is only to be consumed in a certain manner. These properties make these tools suitable for use in isolated situations to test connections of interest.

As long as the output of such tools is comprehensible by users, such employment is acceptable for generic Internet applications. The reason behind this is that in the traditional architecture of Internet applications, mostly designed around the client-server model or a variation thereof, the destination can only be one remote host. Hence, measurement is intrinsically a reactive and isolated activity.

However, such employment requires a high level of attention and cannot be scaled to be used by a large number of hosts. In heterogeneous distributed systems, there is usually more than one host that can provide a resource or service. It is thus important for acquiring network measurements using scalable, automated means. It thus becomes important not just to reduce measurement overhead but also to ensure that measurement output can be integrated with other system components in order to use feedback from the network to adapt applications in run-time.

2.6 Summary

This chapter has demonstrated that there are plenty of mechanisms to measure and estimate different network metrics. Each of these mechanisms introduces some

^{IV}The use of SNMP itself is plagued by different security vulnerabilities [314] and performance shortcomings [113, 251].

network, processing or hardware overhead. Usually, it is a combination of these in different ratios.

Due to their intrusiveness, active techniques are conventionally used only occasionally to diagnose network paths that are not performing as expected. On the other hand, passive techniques are conventionally used as a long-term solution to build a history of network performance. However, some passive measurement techniques retrieve metrics of limited accuracy. ICMP-based techniques are simple to use and universally implemented. However, they are commonly unreliable and inaccurate.

Distributed Resource Management



THIS chapter discusses the topic of distributed resource management. It provides a literature review of this field and reviews related work in the context of grids. First, section 3.1 outlines resource management in pre-grid distributed systems. These systems were largely homogeneous and tightly connected, and hence had different challenges than those of grids. Nevertheless, current distributed systems such as grids and clouds have distinct roots in the progress made to develop early distributed systems. By highlighting these roots, we aim to help the reader gain a better understanding of the state of the art and thereupon its shortcomings. Then, section 3.2 discusses resource supervision, one of the main functions that enable resource management. This function entails discovering resources, finding their capabilities and monitoring them. Particular attention is paid to the application of resource supervision in grid systems, reviewing major standardisation efforts and an array of proposed solutions in this area.

3.1 Resource Management in Distributed Systems

The story from the very first to the latest distributed computing paradigm is milestone by a series of abstraction levels. We regard it as pertinent to shed some light on this story. The objective here is not to offer a conclusive survey of resource management in early distributed systems. Instead, it is to provide a foundation for some of the work which will be introduced later in the chapter.

The rest of the section is organised as follows: section 3.1.1 gives a short back-

ground on the origins of distributed systems; section 3.1.2 outlines the prominent and relevant technologies in the field of resource management in homogeneous distributed systems; and section 3.1.3 presents the difficulties associated with resource management in heterogeneous distributed systems.

3.1.1 The Origins of Distributed Systems

Definition

In uniprocessor systems, resource management is a primary responsibility of the OS. Resource management became more complicated with the introduction of distributed systems. A *distributed system*, in its broadest definition, is one that comprises of more than one computer with the goal of reaching a level of performance and/or providing a service that is quite difficult or infeasible to do on a single computer. The catch is, as in the case of uniprocessor systems, a distributed system needs to operate in such a way that the interaction between its different components is transparent to its users.

Emergence

The revolutionising introduction of distributed systems came about because of two technological advances [117]. The first was the introduction of Very-Large-Scale Integration (VLSI) chip design which produced processors that were supremely powerful yet extremely cheap compared to their predecessors. These new processors, known as *microprocessors*, also required much less space and maintenance. This meant that computers became much more affordable and accessible to many more people than ever before. It also meant that they could be used for more demanding applications. The VLSI revolution eventually invalidated Grosch's Law¹, starting a still-ongoing race to design high-performance multiprocessor systems.

The second advance was the development of high-speed networks. The interconnection of computers, near and distant, opened the gates for an enormous number of new applications. The opportunities created by this advance are still increasing to this day as more and more technologies are developed to capitalise on connecting computer systems of different capabilities and assets.

¹Grosch's Law [177] states that the clock speed ratio of two computers is proportional to the square of their costs. The law was invalidated because it became possible to get more performance from a computer made up of a collection of less powerful processors than from a computer with one very powerful processor of the same (collective) cost.

These two technological advancements marked the dawn of a new era in computing where the von Neumann architecture was no longer sacred and where geographical locations are rather irrelevant. This was the birthdate of distributed computing.

Architectures

The increasing demand for computing power and software sophistication accelerated the growth and procreation of distributed architectures. These architectures can be classified into three main configurations:

- A *parallel system* is a computer that hosts more than one processing element (PE) in order to reduce execution time, increase the number of concurrent tasks, or gain more performance per cost.
- *Cluster systems*, or *clusters*, share the same goals as parallel systems but they comprise of a number of stand-alone computers. Clusters offer less parallelism but a higher degree of PE independence which provides support for more scalability and functional breadth.
- A group of largely or completely independent computers that aim to share resources through a common application.

The first two architectures are both made up of resources that are largely similar if not identical, reside in close proximity and are hence highly connected, and are managed by a single entity. They are referred to as *homogeneous distributed systems* and are discussed in subsection 3.1.2. Systems of the third architecture will be referred to as *heterogeneous distributed systems* and are discussed in subsection 3.1.3.

3.1.2 Homogeneous Distributed Systems

Challenges

The three main challenges faced by homogeneous distributed systems are as follows:

- *PE management*: What is a PE? Is it just a processor? Does it have any dedicated memory? How should individual PEs operate? Do all PEs execute the same instruction in parallel or should there be more autonomy? If memory (or any similar resource for that matter) is shared, how should access to that resource be regulated? If memory is not shared, what alternative mechanism should be implemented to coordinate between different PEs?

- *Task management*: How to break down tasks into sub-tasks that can be executed on separate PEs? How to ensure that division into sub-tasks would not compromise correct execution? Is there a need for sub-task communication, and if so how is it accomplished? How to avoid, detect and resolve execution failures? How to manage task queues in order to optimise execution?
- *Transparency*: The system should provide an interface through which users perceive and interact with it. It should allow users to employ available computational capabilities without worrying about the two previous issues.

Notice the lack of any network-related challenges or concerns. To such tightly-coupled systems, the network is merely a facility to be used, pretty much like a processor bus. From this perspective, the network is not a resource of unpredictable performance; it is not a system component that needs to be opened for inspection to ensure optimal operation; it is not a wild card. Hence, no special arrangements are needed to manage the network in homogeneous distributed systems. All that is required in such environments is some sort of interface between the network and other system components to translate and to regulate access.

Classification

Homogeneous distributed systems are classified based on the number of independent computers they contain and the number of administrative domains they span, as outlined by Table 3.1. Note that it is an intrinsic feature of any distributed system to *behave* as one computer. This notion is distinct from the number of self-sufficient computers that the system comprises of.

		Computers	
		<i>Single</i>	<i>Multiple</i>
Domains	<i>Single</i>	Parallel	Cluster
	<i>Multiple</i>	n/a	Meta

Table 3.1: *Classification of Distributed Systems*

Parallel Systems Figure 3.1 depicts a typical parallel computer setup. There are a number of vital architectural decisions to be made when designing a parallel computer [62]. These are mainly related to the challenge of PE management as already described. What is of interest here, however, is how to undertake the challenge of task management.

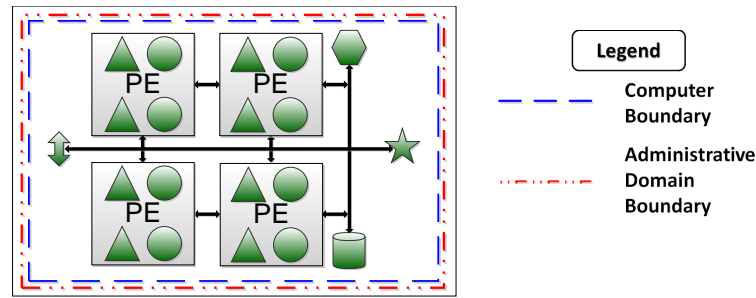


Figure 3.1: An Example of a Parallel Architecture

Division of tasks needs to be done in such a way that ensures that program correctness is not jeopardised and that the load across PEs is as balanced as possible in order to minimise idle time. This task is a demanding one so the associated overhead also needs to be kept to a minimum in order to maximise performance gain.

There are two approaches to assigning sub-tasks across PEs [119]. The first is *static allocation* where deterministic algorithms fed with certain task parameters are used to determine sub-tasks and assign them before run-time. Once done, the assignments do not change. Therefore, very little overhead is required to allocate resources and balance load. However, this approach is only successful if contention during the expected execution time is remarkably foreseeable. *Dynamic allocation*, on the other hand, reacts during run-time to imbalanced load (due to task termination, new task submission, PE failure, etc.). This obviously involves more overhead but is most efficient in achieving good resource allocation where contention or other system environment variables are unknown or unpredictable. The more variables there are in the system, the more unpredictable it becomes and consequently the more appropriate it is to use dynamic resource allocation.

Cluster Systems The evolution of parallel architectures paired with the continuing exponential growth in processor speed gave rise to extremely powerful parallel computers known as *supercomputers*. Supercomputers rapidly developed to break the Giga-FLOPS (FLoating point Operations Per Second) limit by the mid-1980s. However, they were extremely expensive and hence their use was limited to special-purpose, computer-intensive applications in business, science and the military.

However, the advance in processor technology also helped in the proliferation of workstations and personal computers. Soon enough there was a large number of desktop computers in almost every business, university or government institute. It eventually became possible to use network connectivity to harness underutilised

computational power. This was the concept of *cluster computing*, which was intended to be the cost-effective response to supercomputing. Clusters typically consisted of commodity hardware running a free OS (Linux), connected using high-speed LAN connections (Ethernet). The only missing element in this recipe was a distributed programming environment. Thanks to parallel computing, there was already a great deal of work done in this regard.

Resource management in cluster systems is of two main organisational schemes [329]. The first is *hierarchical organisation* in which a group of alike computers, called *workers*, are centrally managed and controlled by one computer, called the *master*. The master deals with all issues regarding resource allocation and provides an interface to submit jobs and retrieve output. In such a setup, illustrated by Figure 3.2, the master is much more advanced than the workers are in that it has all the mechanisms to manage, monitor and control. The workers, on the other hand, do not need much more than an OS. Systems that implement such an organisational scheme include *batch queuing systems* where the workers are dedicated for job execution (e.g. Beowulf [315]) and *cycle scavengers* where workers are undedicated machines (e.g. Condor [240] and Nimrod [52]).

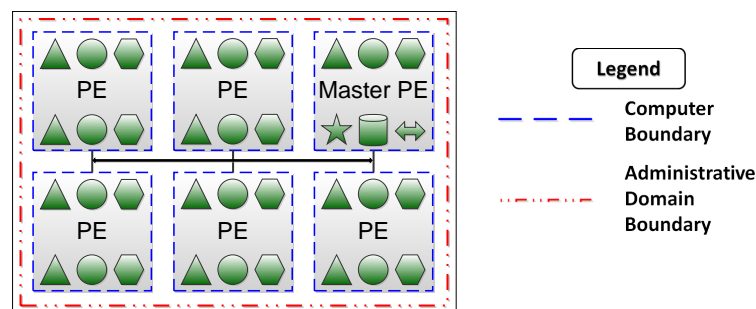


Figure 3.2: An Example of a Cluster Arranged Hierarchically

The second configuration is called *symmetric organisation* and is depicted in Figure 3.3. Here, all nodes are identical and each implements mechanisms not just to execute but also to manage jobs, monitor execution and provide a user interface. This collection of mechanisms is referred to as a *distributed operating system* [327]. To a program, a symmetric cluster appears as just one node. The transparency associated with such a setup facilitates easier process migration even during run-time. Examples of distributed OSs include Amoeba [328], Legion [176] and MOSIX [73].

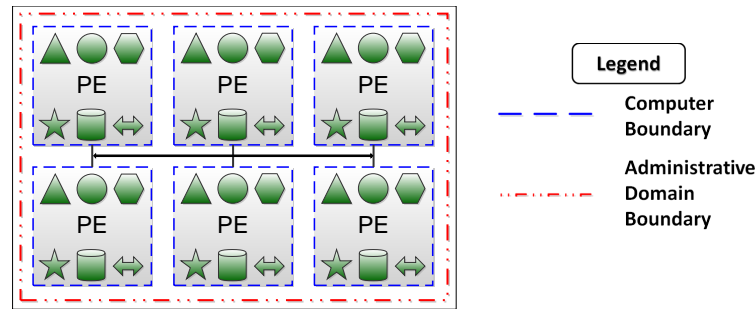


Figure 3.3: An Example of a Cluster Arranged Symmetrically

3.1.3 Heterogeneous Distributed Systems

Definition

Heterogeneous distributed systems, or *meta* computer systems as presented in Table 3.1, are ones that are made up of asymmetric resources across more than one domain. As such, few assumptions could be made about hardware, OS, policies, etc. Figure 3.4 offers an example of such system, where domain boundary, resource and platform variation are arbitrated to illustrate some of the sources of difficulty in such systems. Examples of heterogeneous distributed systems include grids and peer-to-peer systems.

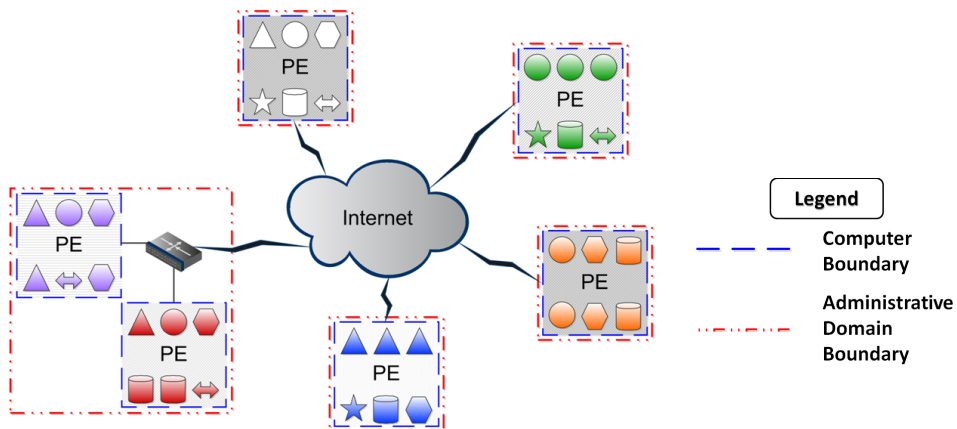


Figure 3.4: An Example of a Heterogeneous Distributed System

Middleware

In order to allow applications to run on top of such a system, there needs to be a means of overcoming its intrinsic heterogeneity. This requires facilitation of inter-domain coordination and resource sharing, something that is both the principal motive and the great challenge of such systems.

The heterogeneity in such systems is countered by introducing a constant across it. This constant enables different system elements to interact and work together. This constant is a software platform called the *middleware* which offers mechanisms to unify intercommunications, describe and manage resources, and handle security among other issues.

Challenges

Such systems are faced with the same challenges as their homogeneous counterparts, only these challenges are complicated further by the added heterogeneity as already discussed. There is, however, an added obstacle that was absent from homogeneous distributed environments.

Cluster computing stretched the parallelism paradigm beyond the boundaries of one computer to span across a collection of independent computers. This is illustrated by comparing Figures 3.2 and 3.3 to Figure 3.1. Compared to parallel systems, clusters are loosely-coupled systems where the PE is less of an atomic hardware resource. Though, PEs in a cluster still reside in one administrative domain.

Similarly, grids introduce another expansion by allowing distributed systems to transcend administrative boundaries. This further increases constituent autonomy and reduces coupling. The grid paradigm, thus, provides geographical emancipation. However, in order to do this, grids incorporate an additional element: wide-area networks.

The network is ever-present as an element in any distributed system. However, in the case of heterogeneous distributed systems such as grids, the network is no longer strictly controllable. The habitat of grid systems is commonly public best-effort networks of non-deterministic nature. Due to this nature and to its vitality in forging grid computing, the network could either be a great enabler or a stifling obstacle.

Therefore, the network in heterogeneous distributed systems can no longer be taken for granted but instead needs to be appreciated as a system resource and managed accordingly.

3.2 Resource Supervision in Grids

Grids are heterogeneous distributed systems that are made up of resources that are constitutionally, physically and administratively afar from each other. Additional

mechanisms are needed to accomplish resource supervision in grid environments. This section reviews different solutions that have been put forward in this regard.

First, however, more commentary is provided about the function of resource supervision in grids. This is necessary given that this thesis aims to enrich this function as a means of improving scheduling.

3.2.1 Challenges of Resource Supervision in Grids

Discovering and monitoring resources around the grid is much more difficult than doing so in parallel and cluster computing environments. The two main factors that distinguish the resource pool in grids from that in other distributed systems are:

- Grid resources are shared between all nodes and so contention can cause availability to vary significantly.
- The distributed nature of grid resources means that they are owned and administered locally by independent organisations with their own distinct policies.

Both these factors are true by the very definition of a grid. There is nothing that can nor should be done to change them. These factors (along with premature job termination, unexpected resource failure, maintenance, etc.) make resource availability in grids very dynamic.

There have been several approaches to solving the problem of resource supervision in grids. The most trivial one (that, surprisingly, seems to be popular in some grids) is to give the user the option as to where to run his/her job. This approach might be suitable if the user had adequate knowledge of the resource pool members and their capabilities, which is fairly presumptuous. Another approach is to rely on pre-defined tables of grid nodes and their local resources. This is comparable to the static allocation method used by early parallel systems, as discussed in subsection 3.1.2. The information provided with such an approach is more often than not outdated and in many cases quite limited. Hence, it is completely inadequate considering the dynamic nature of grids.

The alternative is to have an active mechanism for discovering and monitoring resources. Such mechanisms are referred to as Grid Information Systems (GISs). A GIS is typically part and parcel of the resource management element of the middleware. However, due to the insufficient support provided by many of such integrated GISs, supplementary GISs have been proposed in the literature and are commonly used.

These augmented solutions interact with the middleware to complement its resource management functionality.

3.2.2 Standards

Monitoring grid systems is not fundamentally different than monitoring any other computer system. However, there is more need to share such information in grids than any other system due to their dynamic, transient and multipartite nature.

Despite this, there does not exist a standard to represent grid metrics nor to assemble grid monitoring systems. Instead, there have been a number of efforts put forward as recommendations. This subsection presents the most prominent of these. The GMA is first presented as the model most widely recognised for designing GISs. This is followed by an outline of the main schemas proposed to represent grid resource properties.

Architectural Model

In 2002, the Open Grid Forum (OGF) released an open architecture as a recommendation for defining GISs called the Grid Monitoring Architecture (GMA) [333]. The GMA is composed of 3 main entities: *producers*, *consumers*, and a *directory service*. Producers are elements of the system that produce measurements of resource capability and availability. Consumers receive the monitoring data from producers. The directory service allows consumers and producers to find each other. Interactions between the different GMA elements are event-based and could be one of three types: *Publish/Subscribe*, *Query/Response*, or *Notification*.

The GMA has been adopted by several GISs. Its decentralised design allows a GIS to be implemented in a truly distributed fashion, suiting the distributed nature of grids. This is facilitated by the presence of a directory service that aids consumers in finding publishers, detaching the act of discovering monitoring information from that of acquiring it. This separation is, however, a cause for concern. One potential problem is the overlap of monitoring responsibilities resulting in data discrepancy. Another is that of consumption scalability as the number of producers grow.

Schema

The Grid Laboratory Uniform Environment (GLUE) Schema [174] provides a collection of objects with attributes to model grid resource capabilities, job requirements

and the relationships between different objects. GLUE originally emerged out of the DataTAG project [8] and was later developed under the umbrella of GLUE-WG [228], an OGF working group.

The schema was introduced to provide a framework-independent interoperability bridge between different grids. For example, grid infrastructures in North America typically use different technologies (middleware, GISs, etc.) than their European counterparts. Moreover, particle physics grids are commonly separate from those used for other e-science applications (such as meteorology, chemistry, etc.).

Despite its exhaustive model to meticulously describe computational grid resources, GLUE fails to describe network resources. This is a recurring deficiency in grid monitoring technologies, which is in itself a clear symptom of a lack of interest in network management within grid environments.

The OGF's Network Measurements Working Group (NM-WG) [24] developed an XML schema [324] for the representation of different network metrics. This provides a uniform way of describing and exchanging network measurements obtained using different tools and mechanisms. The NM-WG does not dictate any particular method of obtaining the measurements.

The Grid Scheduling Architecture Research Group (GSA-RG) is part of the OGF and is responsible for defining an architecture to support interaction between different schedulers in “an OGF-complaint ecosystem” [14]. The definition is set to support different grid resources including “network, software, data, storage and processing units” [361].

Several other schemas were proposed to address the representation and communication of grid resource properties. Beside NM-WG, other OGF groups proposed DAMED [180] for describing resource-related events, JSIM [318] which is an object-oriented CIM^{II}-based model, and [317] which provides examples on how to model resources in an OGSA environment. Efforts from outside the OGF include the NorduGrid Information System [227] which extends the LDAP schema used by Globus MDS (which will be presented shortly).

3.2.3 Tools

Over the past 10 years or so, there has been a significant interest in monitoring grid systems. This is a natural phenomenon as already discussed (see 2.1). The focus is

^{II}The Common Information Model (CIM) is a schema for representing management information in distributed systems [152].

slightly different for each solution, but the common aim is to measure or estimate the performance of different resources in the grid. This subsection first presents MDS due to its status as the de facto GIS and its wide use. The subsection then presents (in chronological order) and comments on some GISs, focusing on ones that are widely used or that monitor network resources.

MDS

GT's Monitoring and Discovery System^{III} (MDS) is a GMA-compliant GIS that provides access to information about resources and services in the grid using standard interfaces.

MDS2 [120], the information service of the Globus Toolkit version 2 (GT2), had an LDAP^{IV}-based client-server architecture. It defined two main roles, i.e. components, that work together using the two following protocols:

- Grid Information Protocol (GRIP) provides access to (i.e. enquiring about) information in aggregate directories and also discovery (looking up) of information providers. GRIP is LDAP-based, MDS2 employing OpenLDAP [30].
- Grid Resource Registration Protocol (GRRP) is used by grid entities to push notifications about their presence to other grid entities. GRRP uses soft-state to deal with remote resource faults.

The two roles MDS defines are:

- Grid Resource Information Service (GRIS) is a VO-neutral, generic (but configurable) information provider component. It notifies other components of the information it provides using a soft-state registration process. An information provider can use GRRP to notify an aggregate directory of its availability.
- Grid Index Information Service (GIIS) is an aggregate directory component that allows consumers to retrieve information from an information provider. GIIS maintains information about grid entities (typically of a single VO) and keeps an index of registered information providers. GRRP messages are used to gather information from GRIS servers and to invite GRIS servers to be registered with a GIIS.

^{III}Originally called the Metacomputing Directory Service [147].

^{IV}Built around the basics of the X.500 DAP protocol, the Lightweight Directory Access Protocol (LDAP) is a protocol for accessing directory services, i.e. one that facilitates maintaining a collection of similar objects. It is typically used for access control and authentication. It defines a model to represent the stored information, and to describe how it is to be organised and referenced.

GRIS providers are created by users to collect information about other grid nodes and publish the information into MDS. First, the user would have to define a schema for representing the information in the Directory Information Tree (DIT). Then, a program needs to be written to gather the information returning them as LDAP Data Interchange Format (LDIF) objects. Finally, the program needs to be activated by adding it to the appropriate list of active GRIS providers.

Like MDS2, MDS3 has a decentralised hierarchical architecture. However, MDS3 (as with the rest of GT3) is service-oriented, being based on OGSA standards. In effect, all resources are described as grid services that register to another grid service called the *Index Service*. The Index Service is composed of the *provider* and *aggregator* components. Every Index Service keeps information about a set of grid services (usually the ones in its VO) and provides an interface that can be used for querying. A VO can have one or more Index Services. This is described in the VO's Index Service.

The following version of the Globus Toolkit, GT4, underwent a major transformation by implementing the WSRF standard. This has given GT4 wide adoption compared to its predecessors. MDS4 introduces a number of information services. The *Index Service* is used to aggregate and publish resource monitoring information, the *Trigger Service* is used to generate notifications for certain events, and the *Archive Service* is used to create data archives about grid resources allowing users to query those archives.

MDS has been deprecated in the latest major release of GT^V due to limited adoption mostly for cataloging instead of monitoring purposes [343].

ReMoS

ReMoS [125] is a query-triggered interface that returns information about the current state of the network (either the whole topology or a predefined portion of it) in a network-independent format. An application could query about the bandwidth and delay associated with each flow. Moreover, ReMoS supports queries regarding multiple simultaneous flows and will take any shared resources into account.

Callbacks can be used by the interface to inform the application when network performance reaches certain thresholds. This relieves the application from polling. ReMoS uses flows as opposed to physical inter-node connections, making the interface network-independent and portable.

^VMDS is replaced by Integrated Information Services (IIS) in GT version 5.

The ReMoS implementation provides status information and predictions of bandwidth and topology. This is carried out by collecting information from routers and other networking devices using SNMP. Future usage of network resources is estimated using any resource reservations as a guide.

Although the manner in which applications invoke ReMoS is beneficial, there are several serious problems with its implementation. First, ReMoS relies almost entirely on SNMP information. Not all layer 2 devices support SNMP. Moreover, some networks (such as Ethernet and other contention-based networks) do not easily provide utilisation information. The authors suggest setting up dedicated nodes that monitor and report traffic load information using promiscuous mode, or in other words tomography. This solution is of limited scalability, and added overhead. Second, deducing network scope based on user queries and SNMP topology information can be inaccurate and exhaustive. Third, monitoring network delays is not carried out by default in most networks. The solution proposed by the authors is to use application-level measurements, introducing an artificial overhead to the network. Fourth, the prediction technique is unsuitable in grid settings as not all grid applications specify their future utilisation using resource reservations. In fact, requirements of most grid applications can only be defined accurately during runtime.

Subhlok et al.

In [320], Subhlok, Lieu and Lowekamp present a means of selecting the best nodes to use for a task based on a number of parameters such as the available path bandwidths and the CPU load on the nodes. Requirements are defined using an interface, while the ReMoS interface is used to provide levels of abstraction in order to obtain information about the topology and the current state of the network.

The approach is simple and not computationally-intensive. The scheme allows the setting of minimum global requirements, and supports unidirectional as well as bidirectional paths. However, it relies on a number of assumptions, one of which is that there is only one path between any two nodes in the network. Although static routing does yield a single path between any pair of nodes, contention and link failures can cause the network topology to change over short periods of time. The system also treats all nodes equally which would not be useful for applications where distinction between nodes according to their capabilities is necessary. Also, defined application requirements can not be changed during execution time.

The selection process of the proposed system is based only on CPU load and available bandwidth. It does not consider other network performance metrics such as latency or jitter, nor end host properties such as available memory or disk storage. Furthermore, the system does not handle more than one flow per application.

The authors state that the problem of optimal node selection would not exist if end hosts employ explicit resource reservation. Indeed, CPU and similar resources can be reserved on remote nodes. However, the performance of the network path to those nodes can not be as easily guaranteed. For example, if the path to a remote node is currently suffering from severe congestion, it might be better for the application to consider other remote nodes even if the guarantees of resource availability on those nodes are weaker. In light of such unpredictable network performance, the node selection problem is still of importance.

gloperf

GT was first proposed as a toolkit that can “respond to dynamic behaviours in the underlying metacomputing environment” [154], although no implementation for monitoring network changes had been suggested at the time. Eventually, a hierarchical tomography-based tool that keeps bandwidth and latency measurements, called *gloperf*, was proposed [235].

The *gloperf* tool addresses scalability issues over populated environments by reducing measurement frequency and by using a hierarchical host-pair representation. The *gloperf* daemons, *gloperfd*, are arranged in a hierarchical fashion and are by default members of a local group. Only one of the members of each group is promoted by the administrator to be a member of the global group. The set up of the arbitrary hierarchy is simple, but requires some configuration from the administrator of each local group.

A *gloperfd* instance is started on each node running GT. The daemon registers itself by creating an object into MDS. Once the daemon has joined a group, it will start periodic bandwidth and latency tests to a random set of its peers (i.e. members of the same group). The tests are unidirectional, end-to-end, based on *netperf*, and include the overhead of standard TCP/IP.

The service is made easy to discover and access by storing *gloperf* information in MDS. The measurements by any node are independent of all other measurements as the results are stored under different objects in MDS. The *gloperf* approach is

simple and decentralised. Moreover, its hierarchical group membership ensures decreased measurement overhead as opposed to that obtained if all daemons were global (i.e. $O(n^2)$). However, network paths that stretch outside the local group require aggregation of local measurements which might affect the accuracy of the overall measurement.

By randomly picking the nodes to which the path metrics should be measured, the probability of all paths being measured at least once decreases as more nodes join the local group and as the inter-measurement interval becomes longer. Furthermore, TCP might not get out of its slow-start phase if the measurement period is not large enough. gloperf does not implement any prediction mechanism.

gloperf was only adopted for a short period and was soon replaced by the Network Weather Service [342].

NWS

The Network Weather Service (NWS) is a distributed tomographic system that provides network and computational performance measurements and forecasts through periodic active probing [359]. These measurements and forecasts aid troubleshooting and enable schedulers to adapt seamlessly to the dynamics of a grid environment.

The network measurements in NWS are carried out by a hierarchical group of *sensors*, i.e. registered end-hosts that actively probe the network on a periodic basis to measure its performance. Sensors also measure local memory and CPU usage. Timestamped measurements are pushed to a shared memory which can be queried by other end-hosts for measurements and predictions in order to monitor utilisation levels and to guide application-level schedulers. These predictions are provided by a number of forecasting functions that access the shared memory to process the data and obtain short-term performance predictions. New performance metrics are compared to the forecasts provided by each function in order to evaluate the accuracy of each function. The NWS adaptive forecaster then relies on these calculated accuracies to minimise overall forecasting error.

Despite the NWS's advanced forecasting mechanisms, it relies on a tomographic measurement architecture that employs active probes. This yields in estimations of domain-to-domain measurements at the cost of high network overhead.

AMP

The AMP project introduced in section 2.4.1 is designed for analysing the network side of high-performance computing. AMP measures different types of metrics between HPC nodes and from core routers. It employs SNMP, sniffing and probing, as well as BGP information retrieval. All gathered information is stored centrally to be analysed. The results of the analyses are then distributed to the HPC community [258, 41].

As AMP focuses on the network performance between HPC sites, monitoring is only done at major nodes. This does not present enough information for or about nodes that are not classed as major. In addition, central storage of measurements forms a single point of failure.

Vazhkudai et al.

Vazhkudai et al. [350] amended the GridFTP implementation to log information about all transferred files. From these logs, the amount of achieved throughput can be easily deduced and utilised to predict future performance using three different mathematical procedures. This technique has a number of important advantages; It is completely passive and hence has no network overhead in the form of measurement probes. In addition, it avoids possible inaccuracies in performance measurement associated with active probes. Furthermore, it is also deployable on a single node as no peer coordination is required.

Measurements are published via MDS2. The advantage of this is that it allows nodes to benefit from the measurements collected from other nodes in their direct vicinity (a very conservative tomographic approach). This publishing scheme, however, exposes the GIS to the shortcomings of the GMA-based MDS model.

Gunter et al.

This work [181, 236] is concerned with pinpointing performance bottlenecks, whether it is in the application itself, the OS, the device drivers, the network interface cards, or the network infrastructure. The approach is to log all events to an SQL database. Bottlenecks can then be identified by comparing recent logs with old ones.

The GMA-compliant approach is broad enough to facilitate the identification of any bottlenecks in the system. However, the introduction of logging elements (i.e. event consumers, buffers, archive feeders, and archives) for each event type on every single node in the grid is a demanding task that produces excessive disk and

processing overheads. Such overheads are deemed acceptable by this system in order to achieve very high levels of reliability.

Wren

In [243], Lowekamp proposes an architecture for monitoring grids using a hybrid technique, i.e. one which uses both active and passive measurements. The study was done for the Watching Resources from the Edge of the Network (Wren) project [49] and it aims to transparently provide measurements of network performance at reduced intrusiveness and complexity.

While grid applications are actively using the network, the packet-pairs technique is passively applied to the ongoing traffic. When the created traffic is too low to use for obtaining measurements using passive monitoring, active probing is employed using the *topology-based steering* technique. Here, physical topology of the network is retrieved in a similar way to ReMoS, along with information about path capacities and utilisation. From the topology, potential bottlenecks are recognised as network connections that were highly utilised, even if for a portion of the time. These are then probed from the least utilised node in the subnetwork in order to obtain full performance metrics.

The topology is also used to deduce whether measurements can be ported from one pair of nodes to another. If the topology indicates that the network is created by a number of subnetworks then the measurement between one node in subnetwork A and another in subnetwork B can be used as an estimate for all pairs between these two subnetworks.

The use of both active and passive techniques, according to the current utilisation of the network path, achieves accuracy and non-intrusiveness, respectively. Along with portability of measurements, this mechanism also improves the scalability of the solution. However, active probing of overutilised links increases the possibility of them being definite bottlenecks over the period of the test and probably also afterwards. Moreover, the proposed solution relies on utilisation measurements received via SNMP which are by default reported on five minutes intervals. Such low resolution does not help in identification of acute spikes in utilisation levels.

eTOP

The Asia-Pacific Advanced Network (APAN) [2] consortium has proposed a tomographic measurement infrastructure called eTOP [237]. eTOP uses different active measurement techniques (such as ping and iperf) triggered from *Active Measurement Points* to obtain metrics like RTT, loss rate, available bandwidth and jitter. The collected information is used to populate a database, which is then used to extract and display data about any end-to-end path. However, the system is targeted towards human users only.

eTOP's architecture is quite similar to that of GMA and MDS: the role of the Active Measurement Points is quite similar to that of GRIS in the MDS design. However, publishing in eTOP is carried out in a more centralised fashion via the relational database. This shift away from the GMA model significantly improves the performance of information retrieval compared to GT MDS [63]. Nevertheless, there are a number of drawbacks with the eTOP design. It is not clear how many Active Measurement Points are required in order to achieve an acceptable level of accuracy. Moreover, the measurements are kept in a central database which becomes a single point of failure and also limits the number of queries that can be handled at once.

R-GMA

Relational Grid Monitoring Architecture (R-GMA) [112] has been proposed by the DataGrid project [166] as a better implementation of GMA. It defines a language and protocol to be used for queries and other interactions. It also offers a better hierarchy which allows it to appear to users as a centralised system. This, compared to strict GMA implementations such as MDS, maintains the scalability associated with a decentralised architecture and improves performance through a better information retrieval model [372]. However, R-GMA still relies on republishing which requires a high maintenance overhead to avoid data discrepancy. With GMA, this overhead is paid by the user of the system (i.e. the scheduler) while with R-GMA the cost is shifted from being external to internal through periodic synchronising messages.

Shavitt et al.

The system proposed in [304] is another tomographic solution that relies on a set of *tracer nodes* that are distributed within the network. Tracers measure the distances between themselves using ping. The paths between the tracers are then divided into

non-overlapping segments using topology information obtained by traceroute. The distance of each segment is identified as a separate variable and a set of simultaneous equations is formed by creating an equation for the measurements obtained for the route between the different tracer pairs. The equations are then solved to find out the distance of each individual segment. Using this, the system can estimate the distance between any two given nodes by finding out the segments involved. The algorithm is simple, inducing a small computational overhead. The network overhead scales well with the increase in number of nodes as not all paths are measured; only those between tracer nodes.

The main problem with this technique is the lack of self-organisation when it clearly is required for correct operation. The tracer nodes have to be in “key locations”; tracers have to be picked to be well-distributed throughout the network so that they discover as many non-tracer nodes as possible. Such an approach is not very fault-tolerant: the failure of one or a few of these tracer nodes will reduce the number of simultaneous equations and hence less segment distances would be calculable. Moreover, picking tracers is not an automated process. Within changing networks, tracers might fall out of their optimal positions affecting the correctness of the mechanism. Furthermore, tomographic techniques are of limited accuracy. Finally, the mechanism is based on ICMP messaging which is in many cases disabled or treated differently.

NGO

The Network-based Grid Optimizer (NGO) [143] was proposed as a “building block” that addresses the need for compound performance metrics. These compound metrics are calculated using fundamental network metrics, such as available bandwidth, provided by MDS2. These compound metrics are formulated using a range of cost functions for different optimisation scenarios, and are made available to any other grid element (e.g. middleware, scheduler, etc.). However, to our knowledge, the NGO building block was never tested. Also, it is unclear if NGO is to be deployed as a grid-wide or domain-based service.

perfSONAR

perfSONAR (Performance focused Service Oriented Network Monitoring Architecture) [183, 336] is a service-oriented monitoring framework to facilitate interoperabil-

ity across different networks. The architecture of perfSONAR consists of three layers. The *Measurement Point* (MP) layer is the lowest and is responsible for providing the measurements using active or passive mechanisms. Every domain specifies its own set of MPs, each of which is responsible for measuring a different network metric.

The next layer is the *Service Layer*. It provides an abstraction level above the measurement collection, and allows domains to exchange measurement data and information. The highest layer is the *User Interface Layer* which consists of a number of visualisation tools. More than one tool is provided to display the measurements in different ways to serve the purposes of different user groups, e.g. Customer Network Management, Network Monitor, VisualperfSONAR and perfSONARUI.

Yousaf and Welzl

The system proposed in [363] is a hybrid one which makes use of both passive and active measurement techniques. A node watches received traffic to calculate an estimate of the lower-bound RTT using the packet-pair technique. If a particular path is not receiving enough traffic, some artificial traffic is injected to probe the network.

The switch between passive and active techniques according to the current network utilisation is a good compromise to achieve acceptable levels of intrusiveness and accuracy. The reduced use of active probing also increases the scalability of the solution. The solution uses the user-level interface libpcap to capture packets from the network interface card and is hence easily deployable. The paper does not comment on the accuracy of the proposed mechanism.

Xie et al.

The authors of [360] use artificial distances to give each node in the network a *Practical Internet Coordinates* (PIC) [114]. The latencies between nodes are then predicted based on the distance between their PICs. Using this technique, a grid application would sort candidate destination nodes in ascending order of their latencies.

The technique is very lightweight, but is entirely focused on RTT. Not only does it not provide any means for obtaining other network metrics, but the paper affirms that RTT is sufficient to understand network status. On the contrary, there are many applications that are interested in network status beyond RTT. For example, data grid applications are interested in both available bandwidth and RTT. Similarly, visualisation grid applications might permit relatively high RTT provided that there is

sufficient bandwidth and low jitter.

Zangrilli and Lowekamp

This work [365, 366, 367] extends the measurement techniques of Wren to use TCP traces to find available bandwidth in order to optimise grid node selection. Timestamps and other information about traffic flows are collected in the kernel and then passed up to user-space to be processed. The technique is quite interesting due to its purely passive technique which does not affect the performance of the network. However, it requires modifications to the kernel, limiting its deployability.

DIANA

Data Intensive and Network Aware (DIANA) [257] is a scheduler developed to improve the performance of data-intensive grid applications by incorporating the state of the network into the scheduling process. Network performance is considered a primary factor of this process, in addition to CPU availability, data size and data location.

Basic network measurements, such as RTT and loss, are obtained through PingER. Throughput is deduced accordingly using the TCP model in [255]. These metrics are then used along with other information about the job to assign a ‘network cost’ that will be used to guide the scheduler in resource selection. There are, however, two main sources of erroneous measurements in this process, both inherited by using PingER: tomography and ICMP-based techniques.

SONoMA

The Service-Oriented Network Measurement Architecture (SONoMA) [193] is an open network measurement framework which supports on-demand network measurements for any network infrastructure. Such measurements are carried out in response to user queries (specifying metrics and node pairs) using a number of active and ICMP measurement tools like ping and traceroute. Obtained measurements are returned to the request originator and also stored in a public database. SONoMA is based on Web-services and hence offers great interoperability with a multitude of solutions.

SONoMA’s on-demand measurement capability is appealing for users and administrators. However, it is not of much use for schedulers as it does not provide any insight or forecast about how the network will act. SONoMA is also based on active

and ICMP-based measurements which create additional network overhead and are unreliable.

Other Efforts

There have been various other efforts proposed to monitor heterogeneous distributed and grid systems. Examples include GMA implementing designs such as JAMM [334], CODE [309], GridRM [72], iGrid [64], Mercury [281] and GridEye [165]. The Condor-based Hawkeye [15] does not implement GMA and is not tomographic. Every node carries out its own measurements of computational resources and reports it to a central directory, a model suited for the hierarchical organisation of Condor clusters. Ganglia [294] is a GIS of comparable architecture that monitors the computational resources in a grid of clusters. iPlane [247] is also of a similar approach but measures network performance using ICMP-based techniques. GlueDomains [108] is a grid-specific monitoring solution that is not dissimilar to the architectures of AMP, DIANA and SONoMA. Other centralised GISs include stream-oriented database management systems such as Tribeca [321] and Gigascope [116]. GHS [322] constructs neural networks to predict computational and network resource availability based on previous observations. These predictions are used to guide scheduling and migration of long-running jobs. Network measurement is done through collection and analysis of detailed trace files, which is a fairly static approach with high processing overhead.

For further reading on GISs, refer to [229], [171], [300], [368] and [239].

3.3 Summary

Section 3.1 introduced the problem of resource management and how it was embarked upon in homogeneous environments such as uniprocessor, parallel and cluster systems. In these systems, elements reside in the same domain, run the same OS, and are owned and controlled locally by a single entity. All this allows the application of central and stringent mechanisms to deal with issues of resource supervision and control. These mechanisms, however, focused entirely on computational resources and not the network, which is perceived as a granted facility rather than a resource that requires managing.


Section 3.1 also introduced heterogeneous distributed systems, in which the disparity and dissociation of resources complicate the task of resource supervision. In parallel and cluster computers, the OS takes care of this issue. In heterogeneous

distributed systems, however, additional mechanisms need to be implemented to supervise resources that lie in different administrative domains, are independent in that they can leave or join in an unforeseeable fashion, and are un-dedicated hence their utilisation level is also unpredictable. These mechanisms then need to be supplemented by a means of obtaining and disseminating resource description and status information. Such means needs to consider the network as a system resource to appreciate its switch from being a facility in the background in the context of homogeneous distributed systems to being a vital component on the forefront of resource allocation in heterogeneous distributed systems.

Section 3.2 discussed resource supervision in grid environments. Despite having no standards per se for this vital resource management function, different recommendations (such as GMA) are offered. This lack of standards allows for a great deal of flexibility in GIS design. However, it also results in interoperability problems and significant overlap between GISs.

Section 3.2 also presented a wide array of GISs. Some of these implement GMA and hence operate in more or less the same manner: sensors or producers collect information and push them to intermediate publishers which are in turn contacted by consumers. The problem with this design is that there are too many publishers to contact in order to build a 'big picture' of grid resources. It is a laborious and relatively taxing task to collect different pieces of information from discrete sources. It could also result in data discrepancies. Several other GISs introduced in this section follow a variant of the GMA recommendation in that they have a decentralised measurement acquisition mechanism but choose to publish through a centralised means. This provides the advantage of having a single publisher which greatly simplifies information retrieval. It also eliminates data discrepancy risks and facilitates collective processing. Depending on how this is implemented, however, it could present a single point of failure and a performance bottleneck. Moreover, such systems suffer from extremely high internal maintenance overhead. Many of the presented GISs adopt tomographic means to obtain measurements in an attempt to reduce measurement overhead. This is in contrast to active measurement frameworks, such as e2emonit [132]. However, tomographic mechanisms suffer from poor accuracy of their estimations as already discussed in the previous chapter.

Requirements

 THE essential tenet of this thesis is to improve grid computing by introducing network-awareness into scheduling practices. Before going any further, this proposition needs investigation and elaboration. This is the scope of this chapter. Section 4.1 commences by commenting on the variety of grid applications there is, section 4.2 discusses their network characteristics, and section 4.3 addresses how these characteristics are catered for by current technologies. We contend that there is room for improvement in this regard by acknowledging the unpredictability of underlying networks. Finally, section 4.4 puts forward a set of stipulations to be fulfilled in order to gain better grid scheduling.

4.1 Taxonomy of Grid Applications

In this subsection we outline a classification of grid applications. Classifying factors used in the literature include application purpose, scale, level of parallelism and machine organization. In our classification, we will use the support offered by the application as the primary classifying factor and the purpose as the secondary one. This provides a tiered taxonomy that clearly identifies the differences and commonalities between a large number of grid applications.

4.1.1 Computational Grids

Computational grids are those built mainly to offer high aggregated computational power. This is perhaps the class of grid applications closest to Foster and Kesselman's

original metacomputing model and, thus, embodies the prevalent perception of what grid applications are.

The computational capacity offered by such distributed supercomputers could be used in different ways. **High Performance Computing (HPC)** aims to deliver a large amount of computational power over a relatively short period of time in order to perform complex computations as quickly as possible. Here, execution time is critical and, to that end, HPC applications run in parallel on different machines¹.

On the other hand, **High Throughput Computing (HTC)** is more concerned with the total number of jobs that could be completed in the long-term than with the magnitude of computational power that could be provided for one job in the short term. In other words, completion time is not as important as the number of jobs that can be taken by the system. To serve this purpose, HTC applications are designed to run as loosely coupled processes. The distinction between HPC and HTC is made clearer by the metrics used for each computing concept: HPC uses FLOPS while HTC uses the number of jobs per (relatively long) period of time, e.g. jobs/month.

Many-task computing (MTC) [287] is a middle-of-the-road concept that aims to achieve high throughput of interdependent jobs over short periods of time. It does so by allowing embarrassingly parallel tasks that do not necessarily conform to an inter-process communication standard such as Message Passing Interface (MPI) [310] to run over parallel computers.

4.1.2 Data Grids

Data grids provide support for data-intensive applications by implementing mechanisms to manage large distributed data repositories. This enables applications to produce and consume vast amounts of information without worrying about data ownership, acquisition, distribution, replication, consistency, etc. Data grids are distinct from distributed databases; the latter has more stringent requirements to govern transactions.

One way the authors of [352] used to classify data grids according to purpose is the level at which the data is manipulated. This could range from the fine-grained **process level**, through **task level** manipulation, to **workflow level** at the coarse-grained end.

¹Unsurprisingly, HPC applications are the main adopters of parallel programming languages and Application Programming Interfaces (APIs).

4.1.3 Service Grids

The third class of grid systems are those that tend to focus on what they can offer in terms of functionality as opposed to magnitude of computational or storage resources. These grids are assembled to provide services that cannot be provided by any one machine. Subclasses of service grids include **Collaborative Grids** that provide a virtual workspace to enable real-time interaction between users (e.g. scientists, market analysts, etc.); **Multimedia Grids** that provide an infrastructure for managing and delivering real-time multimedia content; and **On-Demand (or Utility) Grids** that use virtualisation to engineer applications to interact between different services in order to solve highly complex problems. The dynamic construction offered by the latter model was the original vision for grid computing and its realisation and uptake in recent years was the precursor to cloud computing [160, 348].

4.2 Grid Application Network Characteristics

Similar to many other concepts, the early days of grid computing came with a great deal of enthusiasm. Many people, especially from academia, were keen on the idea and invested heavily into developing, using and promoting the use of grid applications. Early examples ranging from search for extraterrestrial life to high energy physics provide a vivid illustration of the huge welcome grid computing received in scientific research circles. This investment has eventually forged what most people consider to be grid applications.

The grid paradigm has expanded to many applications in different fields. The technologies have matured to encompass these proliferations. This has all altered the definition of grid applications, but did not cause any confusion over the term. Debate was later sparked by associating the ‘grid’ term to applications that are not necessarily grid applications in the sense defined in the beginning of this chapter. This caused ambiguity around the ‘grid application’ term.

We adhere to the definition introduced in subsection 1.1.1 which is the formula most commonly agreed upon, as introduced by Foster and Kesselman [154] and redefined in [155]. In light of this, we now look at the characteristics of grid applications from a network perspective. To many, this is all about large data transfers. Although data transfers are indeed a predominant characteristic [144], other characteristics too have an impact on the network requirements of grid applications.

4.2.1 Data Transfers

Grid applications are typically described as ones that handle huge amounts of data. This is because the early and main adopters of the grid computing paradigm were e-scientists, and in particular particle physicists, who more often than not need to transfer, process and store extremely large datasets. This fact lies behind the common belief that grid transfers are ‘elephants and mice’ [340], i.e. flows are either huge bulks of data or relatively tiny control signals.

Although such dichotomic assumption could be used to interpret certain characteristics of generic Web traffic (as in [78]), it presents a misconstrued description of grid traffic. We conducted a survey of different grid applications to ascertain this. The survey included e-science as well as e-commerce grid applications. Initial findings of the survey were published in [136, 135, 134]. More data was gathered later, the results of which were documented in [187, 168, 245]. Here we include some of the final findings.

Figure 4.1 depicts the dataset size logistic distribution and corresponding sigmoid curve. The graph clearly discredits the dichotomic assumption as most datasets seem to be in the mid-range (around 10 MB) rather than at either extremes. Almost 10% of the datasets of all applications are in bulks smaller than 100 kB in size, 55% are in bulks of 1-100 MB, and 17% are in bulks of 10 GB or more. These results illustrate the diverse nature of grid datasets, refuting the dichotomy assumption. The results also show how different grid traffic is in comparison with generic IP traffic, such as Web traffic, in terms of connection activity, data granularity and throughput.

4.2.2 Control Messaging

Two aspects of grid applications influence its control signalling behaviour. The first is the level of parallelism. Tightly coupled applications require strict governance and regulated inter-process synchronisation. Thus, each process exchanges a large volume of messages with other processes. On the other hand, loosely coupled applications require far less control messaging per process. The number of control messages per process could, therefore, serve as an indication of the level of parallelism, an aspect otherwise difficult to quantify. The second factor that affects the signalling behaviour of a grid application is the number of processes. As this number increases, more control messages are required.

The inter-process coordination overhead of an application is essentially the prod-

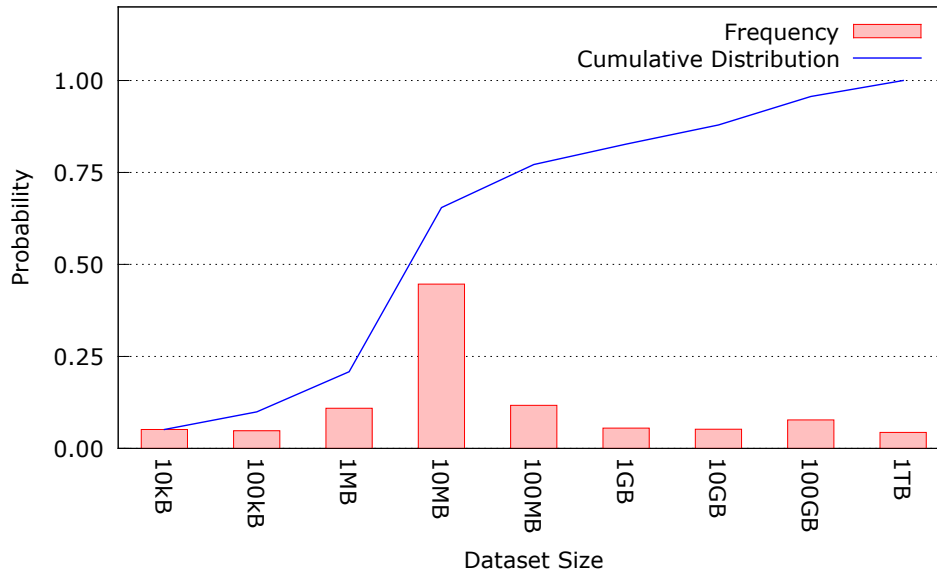


Figure 4.1: Histogram and Cumulative Distribution Function of Dataset Sizes

uct of the size and the frequency of its control messages, as reflected by these two factors. Figure 4.2 shows how the grid application space is divided using the above two factors. Applications with a small number of highly-parallel processes are otherwise known as HPC applications. Examples include weather forecasting and geodynamo modelling. Embarrassingly parallel applications with a large number of processes, such as Monte Carlo simulations and molecular dynamics applications, are also known as HTC applications. MTC applications are those with a large number of interdependent processes. These applications (e.g. Sawzall [280] and Dryad [201]) are usually made up of complex workflows that process shared datasets. The control messaging overhead is thus high. Lastly is the group of applications with a small number of almost independent processes. We refer to these as *traverse* applications. Few applications fall in this set as most grid applications would lean towards either the HPC or HTC model. Understandably, traverse applications have a lot more in common with other distributed applications than with what is the common perception of a grid application. Examples of traverse applications include some data mining applications, such as [93], autonomous mobile grid applications, such as [172], and many black box e-business web services.

In addition to the two factors outlined above, other aspects of grid applications affect their signalling behaviour. These include the number of VOs and their policies, the number of users, the employed middleware, and the underlying technologies such as XML. Such characteristics require careful profiling and benchmarking in order

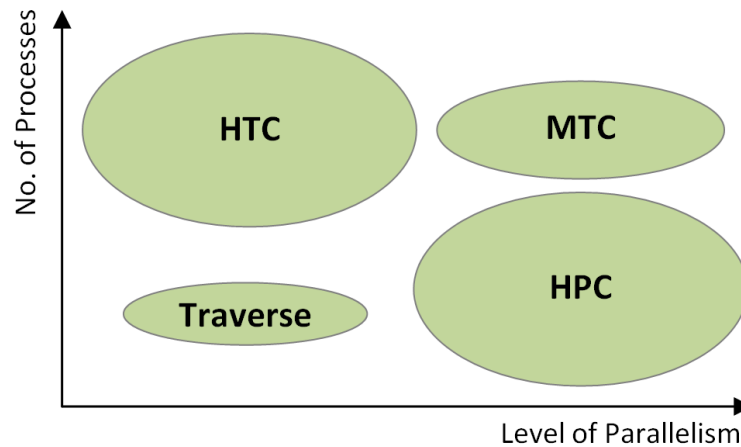


Figure 4.2: *Grid Application Types in Respect to the Level of Parallelism and the Number of Processes*

to develop an understanding of their impact on signalling behaviour in particular and application behaviour in general. Some work has been done in this area, e.g. [325, 370, 161].

4.2.3 Time Urgency

Certain applications need to enforce deadlines on the delivery of their data. Datasets that arrive later than the deadlines are discarded as they are considered of no more use. HPC, MTC, collaborative and multimedia grid applications are examples of such time-critical applications. Embarrassingly parallel applications, on the other hand, do not implement such policies.

In addition to deadlines set by the application, there are strategic policies put in place in order to meet certain goals such as scalability, expressiveness, dynamicity, security and uniformity [147]. For example, an operational goal of a grid application could be to schedule all jobs submitted by users within 30 minutes of submission, or to successfully execute 99% of all jobs within the deadline set by users. This has an indirect impact on the urgency of grid applications.

4.2.4 Distribution

The *raison d'être* of a grid system is to share resources regardless of their geographical locations. By definition, thus, the network that interconnects these resources and their users spans across different administrative boundaries. This commonly involves traversing one or more shared networks along an end-to-end path. In such networks,

network resources are limited and of uncertain performance due to contention by other users.

This is in stark comparison to legacy HPC systems where privately owned networks were used. These offered abundant network resources, enough to exceed traffic delivery requirements. They, hence, were of predictable performance and behaviour which cast the network as an assured resource.

In grid's cross-domain habitats, however, the network introduces a certain degree of unpredictability. Sometimes, this effect is attenuated by overprovisioning the network. *Overprovisioning* refers to the act of improving the capacities of network infrastructures in the hope of diminishing any degradation of service that can result from unforeseeable increases in contention. This measure is evident in many e-science and e-business grid deployments. However, a large number of other grid applications, such as collaborative computing, do not possess the necessary funds to secure overprovisioned network resources and default to using best-effort networks as is. Overprovisioning will be discussed again in the following section.

4.3 Suitability of Current Grid Monitoring Solutions

The networks used for HPC applications and for early heterogeneous distributed systems were owned and managed local networks and hence posed no threat to the performance of these systems. However, the demands of such heterogeneous distributed systems grew more and more over time as discussed in section 4.2. This became a problem once these applications migrated to run over shared networks such as the Internet. In this environment, only best-effort delivery service is available, contention is high and unpredictable, resources are shared and hence cannot be managed, and quality of service is extremely difficult to guarantee. Despite this being a problem, little has been done to remedy it in an appropriate fashion. This is the argument of this section (and is indeed the first research goal of the thesis).

There are three main approaches to dealing with the unpredictable performance of shared networks: *overprovisioning*, *advance resource reservation*, and *network-aware scheduling*. These are introduced in due course within the following paragraphs.

Motivated by the success of the I-WAY project [123], Foster and Kesselman laid down their vision for a global computing infrastructure in [154]. This seminal paper introduced Globus as the technology to enable the assembly of such virtual super-

computer. It was clear from the paper that coordinating heterogeneous resources that are connected by unreliable networks is a central matter that deserves attention.

However, this attention soon dissipated and was never substantiated until a few years later when grid applications prospered. Huge interest from scientific research communities caused them to multiply in number and their demands to grow. Once the network was identified as a performance bottleneck, a number of solutions were proposed to remedy this. Examples include GISs (such as gloperf, NWS, Wren, etc.) and other approaches (such as caching, c.f. [337]).

Despite their drawbacks that were discussed in section 3.2.3, these solutions offered means of obtaining network performance measurements. However, these efforts were not advanced much further in the following years because of the resurgence of the ‘guaranteed network resources’ belief within the grid community. We attribute this to two reasons. The first reason behind this revival is the sheer amount of money invested in e-science grid applications, and the second being the dramatic technological (as well as commercial) advances in physical network infrastructures [288, 264, 308], routing equipment [272, 223, 262], routing protocols [346, 284], transport protocols [316, 61, 148] and standards [83, 292]. These coinciding phenomena ultimately lead to overprovisioned grid network infrastructures.

A common argument is that **overprovisioning** is enough to dissolve all concerns about non-guaranteed network performance. Overprovisioning does indeed allow networks to accommodate more traffic. However, it is not always appropriate, sufficient and viable. First, the desired effect of overprovisioning is severely undercut in cases where the core network, the default recipient of overprovisioning procedures, is not the source of poor network performance. End-to-end performance deterioration is more commonly attributed to problems in the last-mile connection such as poor hardware performance, faulty networking device configurations, poor link quality, long local queues, unoptimised TCP stacks, etc. Second, overprovisioning is only a transitory remedy considering the rate at which application requirements grow. Third, overprovisioning does not offer any guarantees. Its optimistic outlook that increased capacity would be enough to provide for network needs fails at times of, for example, great contention surges. In such situations, overprovisioning offers no solution as it retains the best-effort approach. Fourth, the option of overprovisioning is not always an available one. This could be due to the huge costs associated with it. It could also be due to other reasons; for example, overprovisioning is not practically possible in many wireless environments.

In situations where overprovisioning is not possible, **advance resource reservation** [265] is sometimes used to ensure that grid jobs receive adequate network performance. Examples of solutions of such approach include GARA [293], NRSE [82], G-QoS [56], GNRB [53], NRM [271] and GNB [92]. This approach is only possible though in networks where there is a certain level of control over all network resources. This might be available in bespoke infrastructures but not for the majority of grid applications which are deployed across public shared networks. Moreover, advance resource reservation requires prior knowledge of job requirements and workflow plans. Such a priori knowledge is rarely available.

Since then, grid technologies and standards have evolved with little or no concern about the network as a system resource. In other words, grid technologies became network-oblivious. Grid schedulers do not incorporate network state into their decision process (e.g. Condor-G). GISs provide all sorts of information about computational resources but nothing about network performance (e.g. MDS), and network-specific GISs were not developed further (e.g. NWS and NGO). Similarly, languages and schemas devised to describe grid jobs and resources do not accommodate network resource description (e.g. GLUE). Measurement tools that were developed for the grid are sometimes used but only for troubleshooting and SLA verification purposes. Such manual and ad hoc use provides no integration with grid schedulers and does not facilitate adaptation to changes in the network.

We argue that there is potential to improve grid performance by using **network-aware scheduling** where grid scheduling is integrated with network performance measurement. As introduced in subsection 3.2.3, there have been a number of works on network measurement for grids. We argue that neither these nor other GISs offer suitable forms of facilitating network-aware scheduling. We now turn our attention to this argument by reviewing work done in this field.

GISs can also be broken down to distinct but dependent functions: *a*) information collection; *b*) information management; and *c*) information dissemination. We categorise the drawbacks of current GISs using these three functions as follows.

4.3.1 Information Collection

The collection of network metrics sometimes proves to be a complicated matter as certain trade-offs come into play. In light of the overview provided by the previous two chapters, various measurement systems are deemed unfitting for grids.

Accuracy

First and foremost is the issue of measurement accuracy. The success of network-aware scheduling hinges on the correctness of the network information supplied to the scheduler. Various measurement tools disregard such importance by using estimated metrics.

Several tools, such as *gloperf*, *NWS*, Shavitt et al.'s work and *DIANA*, employ tomography. Although metrics calculated using this technique could be quite close to the real values, they remain merely estimates. Erroneous projections endanger the potential gain of using network-aware scheduling. Building on top of that, e.g. for prediction purposes, increases the magnitude of the error between the estimates and the real performance [350, 257].

Furthermore, measurements calculated using tomography are only edge-to-edge. This information is insufficient. Degraded network performance is often due to hardware or software problems in the last-mile connection. This is one reason why true end-to-end performance is important, i.e. that measured from end-system to end-system and not PoP^{II}-to-PoP.

Another source of erroneous measurements is the use of ICMP. Numerous tools utilise this technique, for example *PingER*, *Surveyor* and *iPlane*. As previously discussed in section 2.3 and subsection 2.5.2, ICMP rate-limiting is not uncommon (c.f. [275, 356]) and hence could result in metrics that deviate from the performance that is actually delivered to the transport layer. Wenwei et al. report this deviation to be as high as 30% [356].

Reliability

ICMP-based tools suffer from more than just inaccurate measurements; they suffer from poor reliability as well. This is because ICMP traffic is filtered by some edge routers [167, 356] in order to prioritise TCP/UDP traffic at times of congestion or to avert vulnerabilities such as resource consumption attacks using ICMP floods. For instance, the National Grid Service (NGS) is a UK grid initiative to provide an infrastructure that enables grid applications for research purposes. At the time of writing, we have found that more than 50% of all NGS sites drop ICMP traffic.

^{II}Point-of-Presence

Overhead

Active measurement techniques are widely used in grids and other distributed systems. Examples include PingER (and solutions that employ it such as DIANA and EGEE's^{III} NPM [267]), Surveyor, gloperf, NWS and e2emonit. Such techniques are mainly employed due to their high reliability as well as the potential of high accuracy. However, there are a number of problems associated with such techniques.

The first and most obvious problem associated with active measurement is the high network overhead they incur. Measurement tools that use such techniques have an intrusive nature by definition, creating a strain on the network [303]. In May 2004, Les Cottrell reported that “25% of the traffic on Abilene (an Internet2 network operations center) is iperf and ping type traffic” [238]. This artificial traffic fills communication channels and increases router queue lengths. Competition for network resources increases which ultimately results in degraded network performance in the form of inflated RTT and packet loss rate.

Second, the level of accuracy achieved by active tools is diminished because of the attempts to reduce the mentioned network overhead. Some tools resolve to using small probes. Such probes are not enough to challenge the network which compromises their accuracy. For instance, the NWS mechanism does not measure the amount of available bandwidth but rather an “artifact of the TCP behaviour” [364] since TCP does not progress beyond the ‘Slow Start’ phase. Vazhkudai et al. [350] empirically show the size of such error to be almost an order of magnitude.

Many tools reduce the volume of probing traffic by using tomography where only a selection of nodes generate artificial probing traffic. The drawbacks of such a solution have already been highlighted in subsection 2.5.3 and earlier in this subsection.

The third problem with active measurement is what we term the *preludial effect*. Metrics retrieved using active measurement techniques are commonly perceived as an accurate reflection of the sort of performance a real flow similar to the probing flow would receive. However, this is not always true [278, 186]. During times of increased load, network metrics computed using a probe that preludes a real flow could vary considerably from those experienced by the real flow itself.

Fourth, active measurement traditionally requires peer coordination which is yet another overhead. This operational task is generally quite challenging to carry out on production sites due to fear of service disruption, security concerns or plain politics.

^{III}Enabling Grid for E-science project [10].

Finally, some measurement tools require a level of intervention outside the network. This compensates for the lack of invasive network traffic with interference at end-host or en-route nodes. Wren, for example, requires end-host kernel modification in order to collect flow metrics. Other tools, including NWS, perfSONAR and others, depend on deploying specific nodes that carry out the measurement role. MRTG, NeTraMet and Wren exploit SNMP measurements gathered from networking devices. Such approaches induce an added overhead in order to collect network metrics. Regardless of the correctness of metrics collected in such a manner, this overhead significantly adds to the cost of deploying such instrumentations.

4.3.2 Information Management

The choice of GIS architecture is a crucial one as it affects the manner by which collected information is managed, which in turn defines some of the key properties of the overall information system. GIS architectures fall into one of two categories: centralised and distributed.

Centralised information systems hold information in one location. Examples include GridRM, GNRB, Hawkeye, iPlane and JAMM. Such systems suffer from two main drawbacks. First, sustainability and scalability are restricted as a direct consequence of the limitations of a centralised model [372]. Second, having a single point of failure poses a great threat to system availability.

Decentralised or distributed information systems deviate from the centralised model in order to maintain scalability and high availability. These often have a distributed information dissemination or publishing configuration. Examples include GlueDomains, GridRM, MDS, and R-GMA. Such a multi-publisher setup results in conflicting, out-of-date and duplicated status information. Thus, significant maintenance overhead is required. Moreover, the dispersion of information across disjoint sources presents an impediment to long-term performance analysis.

4.3.3 Information Dissemination

Beside the problem of data discrepancy that could arise from a decentralised information management architecture, the task of network-aware scheduling is also complicated using multi-publisher models, such as that of GMA which is implemented by many GISs e.g. GlueDomains, GNB, GridRM, JAMM, MDS, Mercury and Vazhkudai et al.'s work. Consider for instance the architecture of Globus MDS, which comprises of

information producers, collectors and consumers in addition to a directory service. For a scheduler to obtain grid-wide status information using such model, it needs to query the directory service to learn about the available information collectors and then proceed to query each of them about the information they manage. This is a lengthy and taxing procedure considering the high performance demands of grid systems. The situation is worsened as the number of registered collectors increases [372] which, ironically, limits the scalability of such decentralised GISs. As a matter of fact, MDS has been quite poorly adopted despite being the most popular implementation of the GMA, an open de facto standard. This is due to its poor scalability and high maintenance costs [145, 122, 165, 372, 343].

4.3.4 Information Consumption

Another issue is that of information consumption. This relates to the intended users of GISs. Many of these systems are developed for human consumption and not for interoperation with other grid solutions, let alone schedulers. This is of benefit to users and administrators to observe load status and investigate anomalies on a case-by-case basis. It does not, however, allow schedulers to obtain network state information. For example, Load Monitor [18] is a Web-based monitoring service for the NGS. It provides information in a graphical representation for site administrators and end-users about resource capacity and utilisation levels. However, it is not suitable for schedulers. The only scheduler-friendly alternative is to query a central LDAP-based BDII database for a list of resources and then again for the load state of each resource. Other human-user-oriented GISs include eTOP and Ganglia.

4.4 Requirements for Network-Aware Grid Scheduling

This chapter has thus far presented the characteristics and requirements of different classes of grid applications, and elaborated on the suitability of current systems to cater for these needs. We argue that these are not sufficient to attain network-aware scheduling and that there is room for improvement in this regard.

In [104], Chen and Hu present a review of network performance techniques. They conclude their study with a view on unanswered research questions in the area. After implying that ISPs would never share their performance data, they state:

“a common measurement infrastructure might ensure that performance

measurements will be end-to-end, consistent, statistically accurate, fair, secure [...], and safe.”

Drawing from these specifications, we conclude this chapter by laying out the following requirements for a GIS that need to be satisfied in order to achieve network-aware scheduling.

Requirement I: Accurate & End-to-End

One of the main requirements is to collect information that is of a high level of accuracy. For this to be possible, two main techniques need to be avoided. The first is tomography where domain-to-domain or subnet-to-subnet metrics are used to deduce network performance for other members of the same domain/subnet. Such estimations unavoidably involve some level of error which is highly undesirable as the main object of providing network measurements to grid schedulers is to enable them to adapt to the true state of the network. Second, ICMP-based measurement techniques need to be avoided as they can too result in erroneous metrics which misrepresent the state of the network.

Requirement II: Non-Intrusive

Active network probing should be avoided whenever possible in order to minimise the amount of artificial traffic in the network. Such traffic obligates the network to accommodate it as well as real traffic, treating both with equal priority, and thus decreasing the overall delivered performance. Moreover, it is not uncommon to confuse such probes with attacks, as we have experienced during this research.

Requirement III: Reliable

Grid schedulers needs to be able to depend on the GIS for measurements. This involves taking care of two main aspects. First, the measurements should be collected in a manner that does not impede the user with administrative restrictions. This is only achievable if the system is trusted in how it works and what it collects, and thus could be easily integrated into the grid software stack. Second, the system should avoid unreliable measurement methods, such as ICMP-based techniques, and make sure that any defects in the reliability of measurement collection is appropriately compensated for.

Requirement IV: Low Operation Overhead

The collection of information should not warrant intervention from the user. Instead, this should be an automated process that perpetuates in the background as grid applications run. This is important in order to attain full, seamless integration with schedulers where they are able to decide on-the-fly which routes are better than others and incorporate this information into their resource allocation process.

Requirement V: Low Deployment Overhead

It is of great importance for the GIS to offer itself to independent and low-impact deployment. As such, the solution should refrain from depending on peer coordination, system-wide deployment or sole-role devices. Moreover, instrumentations that require significant deployment costs, such as kernel modifications, necessitate too much of a deployment overhead.

Requirement VI: Decentralised

Several GISs rely on a centralised model to collect and/or disseminate information. We see this as a major drawback in their design as they cater for high-performance distributed systems. Employing a centralised architecture introduces a single point of failure and a performance bottleneck. Decentralisation, on the other hand, allows the system to easily scale, handle high load and implement redundancy measures.

Requirement VII: Single Point-of-Contact

Despite the need for a decentralised information system, we realise the importance of having a single point-of-contact for information-related transactions. In essence, this implies an architecture with a single aggregator to which information is submitted and a single publisher from which information is then disseminated. Such architecture is important as it eliminates problems related to multi-point transactions. This means that producers need to communicate the data they gather with only one element of the system (i.e. the aggregator) and consumers need to query only one element as well (i.e. the publisher).

In order to realise this, there needs to be a disjunction between the logical architecture and physical implementation whereby the former is centralised whilst the latter is distributed. Such architecture also enables the implementation of retro-

spective analysis procedures to be carried out on the collected information in order, for instance, to predict performance, distinguish patterns or detect anomalies.

Requirement VIII: Selective

The system should maintain a certain degree of measurement detail in order to deliver fine-grain information. Nonetheless, this should not warrant a solution where detailed logs are accumulated about all network activities. Instead, the information should be just enough to fulfil the needs of network-aware scheduling. The system, hence, has to be fairly selective about what information it collects and manages. This should be a conscious decision in order to reduce the load of the system and to allow it to scale easily.

Requirement IX: Interoperable

We argue that measurement and information systems should be readily interoperable with other grid technologies, namely schedulers and other information mechanisms. The primary aim of this is to enable schedulers to retrieve this information when required as part of the resource discovery and allocation processes. Conformity to current standard interfaces and protocols facilitates such interoperability. Hence, the system needs to adopt these standards in order to appertain to other grid technologies, especially schedulers.

4.5 Summary

This chapter has served to inspect and explain the impetus behind this thesis. This was done by defining grid application classes in section 4.1, reflecting upon the range of characteristics and requirements they exhibit in section 4.2, and reviewing current GISs in section 4.3. By doing this, the chapter has put forward the two following main arguments.

- *Consciousness of the state of the network is important to achieve efficient grid scheduling.* Scheduling in heterogeneous environments such as grids must incorporate reliable information about system resources in order to adapt to changes in the state of these resources. The importance of this guideline has been highlighted numerous times in the literature (c.f. [154, 80, 362, 77]). However, awareness about the state of the network is absent from many grid

technologies including schedulers. The early part of section 4.3 highlighted some of the reasons behind this.

- *Current solutions suffer from several drawbacks that prevent effective network-aware scheduling.* In order to enable network-aware grid scheduling, accurate and reliable network performance information needs to be supplied to schedulers in an efficient and scalable manner. Section 4.3 identified how current grid information systems fall short in delivering such information.

This provides motivation to act on solving this problem. In order to enable efficient network-aware scheduling, a series of 9 conditions for grid network monitoring solutions were put forward in section 4.4.

Part III

Proposed Solution

Simplicity is a great virtue, but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.

EDSGER W. DIJKSTRA

Design



GRIDMAP is a solution devised to provide measurements and predictions of network performance to grid schedulers in a way that meets the requirements set in section 4.4. This chapter covers GridMAP's architectural design and features that were developed to meet this aim. This includes the means of collecting, managing and disseminating performance information as well as the benefits of the solution's architectural design.

5.1 Overview of the GridMAP Solution

GridMAP (Grid Monitoring, Analysis and Prediction) is a solution that collects, analyses and disseminates network measurements. It is specifically designed to aid grid schedulers in adapting to contention over end-to-end network resources, thereupon facilitating network-aware scheduling.

The GridMAP solution essentially carries out 3 main tasks: collecting, managing and disseminating measurements. GridMAP implements these tasks using a twofold design. The first part, called *Pasinemda*, carries out the measurements and hands them over to the second part, called the *GridMAP service*, which stores and manages the measurements and uses them to provide predictions of network performance on request to grid schedulers. Hence, the solution is of a dualistic albeit disproportionate nature.

By carrying out these tasks, the introduction of GridMAP creates a closed-loop system to guide the process of resource selection based on the observed network

measurements. This is illustrated in Figure 5.1. The collection of measurements also allows for more sophistication in the information that is fed back to the scheduler.

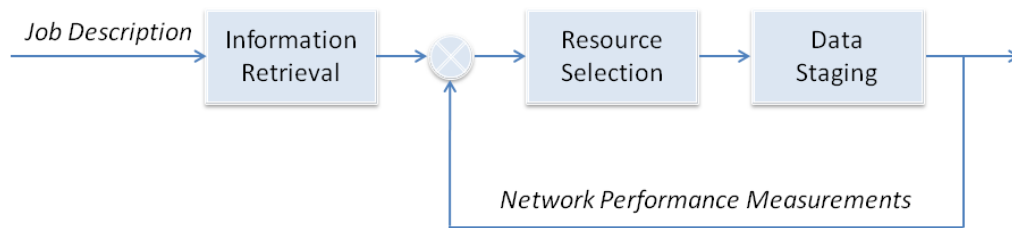


Figure 5.1: *Closed-Loop Network-Aware Scheduling System*

The two main elements of the GridMAP solution are introduced in the following two sections.

5.2 Pasinemda

The *PASsive Network Measurement DAemon* (Pasinemda) is an element of GridMAP responsible for creating the information that will flow through the rest of the system. It calculates two fundamental network metrics: RTT and achieved throughput. The aim is to do so as unobtrusively as possible in order to keep added overheads to a minimum, while at the same time refrain from any compromise on accuracy. Pasinemda accomplishes this by exploiting one of the intrinsic properties of grid applications, namely the abundance of connection-oriented flows.

This section explains the design of Pasinemda. First, the rationale behind the measurement technique and its relevance to grids is discussed. The details of the measurement technique are then presented. This is followed by some comments about the advantages of this technique.

5.2.1 Rationale

Grid nodes constantly exchange data sets, job state, result sets, and control signals during operation. One of the findings of the survey partially discussed in section 4.2.1 (and published in [136] and [135]) is that this virtually continuous communication in the grid, whether a few kilobytes or hundreds of gigabytes in size, is carried out using connection-oriented protocols; chiefly TCP, but in some cases UDT¹ [178].

Pasinemda exploits such frequent connection-oriented interactions to measure network performance. A number of basic network metrics could be extracted simply

¹UDP-based Data Transfer Protocol

by listening to ongoing traffic; RTT can be measured by observing handshakes, packet loss by recording retransmission instances, one-way jitter by examining variance in inter-ACK delays, and achieved throughput by keeping track of transmission size and session duration. Pasinemda monitors handshakes and sessions to measure RTT and achieved throughput, the latter henceforth referred to as *throughput*. This is detailed in the following subsection.

As introduced in subsection 2.4.4, TCP-ping, i.e. the technique of observing TCP handshakes to deduct RTT, is not entirely a new one. In fact, Pasinemda's RTT measurement technique is the same as the first method proposed by Jiang and Dovoloris in [210]. The novelty of Pasinemda lies in the context in which it is applied. Grids provide an abundance of TCP connections that can be exploited using this technique to provide greatly accurate measurements without the need for generating additional traffic. It is not feasible to implement this technique in other systems that do not share the same reliance on and abundance of TCP transfers. When TCP-ping is implemented in such systems, it needs to be supplemented with active probes which prompts overhead, accuracy and security concerns.

5.2.2 Measurement Technique

A typical TCP session is made up of three phases: connection establishment or *handshake*, data exchange, and connection termination or *teardown*. The process of connection establishment, as portrayed in Figure 5.2, is initiated by an empty packet with the SYN flag raised in the TCP header. This is called a *SYN message*. The destination replies with a similar packet with two TCP flags set: SYN and ACK. The process is completed with an ACK message from the peer that initiated the handshake process. Through this three-way handshake process, the nodes establish a connection and agree on the sequence numbers to be used during subsequent exchanges within the session.

The duration between sending a SYN message at time t_0 and receiving the corresponding SYN-ACK message at time t_1 is denoted by t_{syn} .

$$t_{syn} = t_1 - t_0 \quad (5.1)$$

This delay comprises a number of different delays incurred when sending a packet through the network, and is given by:

$$t_{syn} = (\text{Propagation delays}) + (\text{Processing delay}) + (\text{Queuing delays}) + (\text{Latency}) \quad (5.2)$$

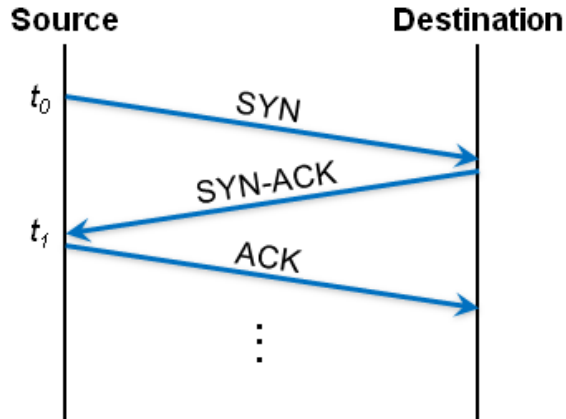


Figure 5.2: TCP Three-Way Handshake Process

This could be translated to:

$$t_{syn} = \left(\frac{d_0}{h_0} + \frac{d_1}{h_1} \right) + p + q + r \quad (5.3)$$

where d_0 and d_1 are the sizes of the SYN and SYN-ACK messages, respectively; h_0 and h_1 are the throughput levels achieved whilst transferring the SYN and SYN-ACK messages, respectively; p is the delay incurred at the destination node whilst processing the request to establish a TCP connection; q is the aggregate queuing delays encountered by the two packets en route; and r is the RTT of the path between the source and destination nodes.

As both SYN and SYN-ACK messages are merely used for signalling during the handshake process, they are both without any payload. This renders their size to the sum of the minimum TCP and IP headers, i.e. a total of 40 bytes each. Propagation delay for such small packets is infinitesimal for most network paths. Eliminating them from Equation 5.3 reduces it to:

$$t_{syn} = p + q + r \quad (5.4)$$

Pasinemda captures packet headers to observe TCP handshakes and record the observed values of t_{syn} . These values are used to represent the RTT of the path, r . Such values are of course polluted as, according to Equation 5.4, the values of p and q are neglected. Under normal circumstances, p is negligible compared to r . In cases where high end-host load creates a relatively significant value of p , the measured t_{syn} would inflate and deviate from the real value of r . Similarly, q is insignificant for healthy routes and only becomes considerable in situations of network congestion or networking device failure.

In that sense, both p and q are to be considered as impurities if t_{syn} is presented as an equivalent of r . Separating the different time periods that make up t_{syn} is of interest for purposes of system characterisation and diagnosis. For example, Martin et al. [253] are interested in dissecting overall RTT in order to isolate the cause of Internet delays. However, this is not the case in the GridMAP outlook. GridMAP is a solution designed to provide operational assistance to grid schedulers in managing the network as a resource. Therefore, GridMAP needs to be pragmatic, not abstruse. GridMAP provides schedulers with a truthful sense of what is expected from the network. In the case of the delay grid applications should anticipate from a path, this should include all constituent delays. Such end-to-end delay, i.e. t_{syn} (Equation 5.4), is more useful to a grid scheduler than the absolute latency of the path or any other component of t_{syn} . A high t_{syn} value effectively indicates to the scheduler that the used network path is not in a good state, which affects the network performance received by the application. Hence, the resource on the other end of the path should be avoided in order to maintain performance. Therefore, the value of t_{syn} is relevant to the scheduler regardless of how the values of p , q or r vary.

Once a FIN flag is encountered signalling the commencement of connection teardown, throughput h is calculated using:

$$h = \frac{d}{t_{flow}} \quad (5.5)$$

where d is the total number of data bytes sent and t_{flow} is the total duration of the flow from handshake acknowledgement till the time of the first FIN signal.

Pasinemda's passive technique, however, has a shortcoming which is that it will only measure paths that are used. The starting position is that of unmeasured paths, while the optimised case is where all paths have been measured at some point in the past. To resolve this issue, Pasinemda needs to be augmented with a mechanism to move nodes from the unmeasured to the measured state in order to include all possible paths into the network performance comparison. We opted to use a non-intrusive single-use probing mechanism to obtain an initial measurement for unmeasured paths. This accessory is discussed further in subsection 5.3.3.

Furthermore, Pasinemda will only gather network measurements at times of traffic. This creates measurement histories that are irregularly spaced which limits the numerical analysis methods that can be applied to the measurements. This shortcoming could be avoided by using periodic probes. However, this would violate Requirements I and II of section 4.4 and hence we refrain from such solution

and instead overcome this obstacle using interpolation, which will be discussed in subsection 6.4.2.

5.2.3 Potential Advantages

Pasinemda's measurements are collected at the nodes that generate the traffic. Collecting measurements in this manner provides a powerful viewpoint and yields metrics that truly represent the experienced network performance.

By using real application data, no artificial traffic is injected into the network and hence no disruption is caused to traffic already traversing the network. In addition, this prevents measurements from being mistaken for threats such as TCP-SYN floods or Denial-of-Service attacks. Furthermore, this eliminates the possibility of measurement traffic following a different route than data, receiving different prioritization, or not going through at all as can be the case with ICMP probes. It also avoids the potential of prelude distortions associated with active measurement.

Moreover, Pasinemda works independently with no need for peer coordination or access to networking devices' accounting information (such as SNMP and IPFIX). This independence along with the simplicity of the Pasinemda measurement technique makes it easily deployable on any grid node either as a stand-alone daemon or alternatively as part of the grid software stack.

5.3 The GridMAP Service

This section introduces the second part of the GridMAP solution: the GridMAP service. It describes the service's role in the GridMAP solution, how it operates, what it provides, and how that is useful to grid schedulers.

5.3.1 Overview

The GridMAP service is an application that runs as a grid service, conforming to the WSRF and the OGSF specifications. The service could be deployed on one or more computers, in which case one of them gives the GridMAP service its Uniform Resource Identifier (URI). This computer is referred to as the *GridMAP service front*.

The service provides a set of standard grid service interfaces that allow convenient access for schedulers, enabling them to receive performance information about relevant nodes and connections. Schedulers can incorporate this information into

their job and data management, and resource allocation processes to automatically adapt to perceived and foreseeable network performance.

5.3.2 Architecture

The GridMAP service is made up of 4 main components: *Measurement Receiver*, *Measurement Database*, *Predictor*, and *Query Handler*. These, along with the service stub, are depicted in Figure 5.3 and then introduced in the remainder of this subsection.

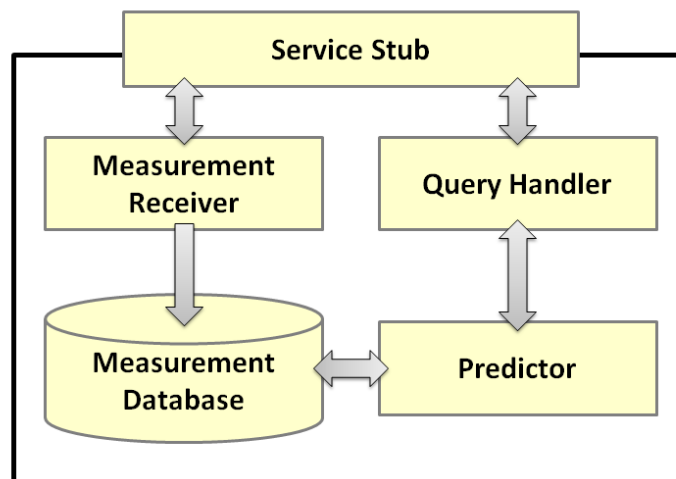


Figure 5.3: *GridMAP Architecture*

Service Stub

This is the intrinsic gatekeeping component of any grid or Web service. Using WSDL, the stub provides a definition of the functionality the service offers and what information is needed in order to invoke such functionality. This includes two communication types: by Pasinemda instances, and by grid schedulers.

Beside providing an interface to the service, the stub is responsible for translating communications between local and remote components. Hence, the stub converts remote method invocations (which are sent using SOAP) into local method calls, and converts serialised objects (also sent using XML over SOAP) into local data structures. The stub also carries out similar duties for communications in the reverse direction. This greatly simplifies the task of developing the service, as it removes the need for undertaking SOAP generation and interpretation. The rest of the modules can be developed as if they were part of a stand-alone application.

The GridMAP service is OGSA-compliant and hence is required to support stateful interaction. This means that state information is cached for each resource that invoked the service. The resource identifier is used as a reference for future communications if stateful invocation is required. Management of state is also carried out by the service stub.

Measurement Receiver

The Measurement Receiver component is part of the service's front-end and is invoked when Pasiemda instances communicate with the service. The Measurement Receiver component essentially provides a translation of measurements from the way they are received from Pasiemda to SQL insert commands.

Measurement Database

The Measurement Database is a repository that holds all the performance measurements collected by Pasiemda. The repository is distributed and replicated across a number of different nodes that host the GridMAP service. Measurements are indexed by the identifier of the reporting node to enable easy retrieval of measurements related to a particular node.

Predictor

This component comprises of numerical analysis algorithms that are invoked by the Query Handler component to provide such services as network performance prediction, pattern recognition, and anomaly detection. This functionality uses measurement history from the Measurement Database which could either be period- or observation-based. In period-based history, a fixed time duration is used as a window so that only measurements obtained during the past duration are used for prediction. On the other hand, observation-based history is bound by the number of measurements rather than by time. The former method is more suitable for production grids where frequent network activity and hence measurements is anticipated. The latter method is better suited for grids that would generate sparse measurements, such as volunteer computing grids.

Query Handler

Like the Measurement Receiver component, the Query Handler component is invoked by the GridMAP service stub. This occurs when a grid scheduler contacts the service enquiring about the network performance of one or more nodes. Such an event results in triggering a series of data analysis procedures (that are part of the Predictor component) to forecast the performance of the network. Once this is complete, the Query Handler then replies to the grid scheduler with a list of nodes ordered according to the expected network performance.

Additionally, the Query Handler serves to distribute analysis and prediction load across the computers that host the GridMAP service. New service invocations are handled in a round-robin fashion between the computers hosting the GridMAP service.

5.3.3 Integration

The following subsections describe the synergy between the GridMAP service and Pasiemda as well as that between the GridMAP service and grid schedulers. These interactions are depicted in Figure 5.4, omitting the service stub to simplify the portrayal of service invocations.

Service & Pasiemda Interaction

Measurements of RTT and throughput collected by Pasiemda are time-stamped and cached locally on the grid nodes where they were collected. On a periodical basis, Pasiemda submits any cached measurements through the service stub. This communication of network measurements, depicted in Figure 5.4, is carried out using the NM-WG XML schema over HTTP. This not only observes current standards, but also allows Pasiemda's reports to traverse communication restrictions such as firewalls. Once received by the Measurement Receiver component of the service, the measurements are indexed and stored in the Measurement Database.

Service & Scheduler Interaction

When a scheduler wishes to consult GridMAP about the network state of one or more grid resources before scheduling a job, it contacts the GridMAP service through its stub with a query. The query includes a list of the candidate resources that the scheduler could allocate for the job. This is received by the Query Handler

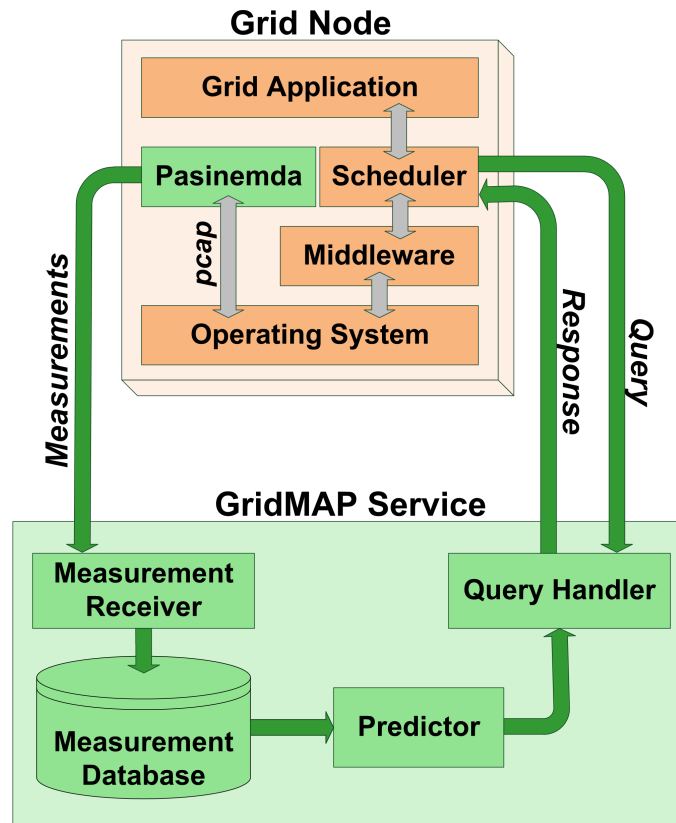


Figure 5.4: Interactions of GridMAP Service with Pasinemda and Grid Scheduler

component which invokes the Predictor component a number of times with the different combinations between the node of the enquiring scheduler and the nodes of the candidate resources. The forecasts produced by the Predictor are then combined to reorder the list of candidate resources and assign weights that correspond to their respective expected network performance. This interaction is also done using the NM-WG schema and is depicted in Figure 5.4 while an example of the exchanged information is given in Figure 5.5.

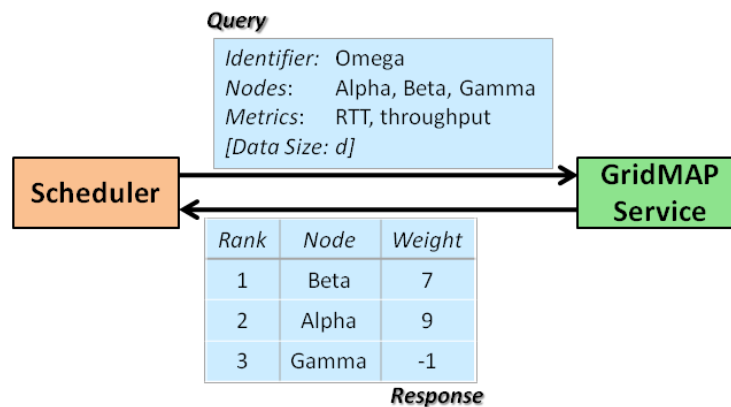


Figure 5.5: Query and Response Exchange between Scheduler and GridMAP Service

In this example, a scheduler running on node Omega is about to allocate resources to execute a job. The scheduler invokes the GridMAP service by identifying itself, the set of nodes that satisfy the job's computational requirements (i.e. Alpha, Beta, Gamma), and the network metrics that are of interest to this type of job (i.e. RTT and throughput). The total size of the files to be transferred (data files, parameter files, executables, etc.) should also be provided if throughput is specified as a metric of interest. The GridMAP service seeks metrics for those candidate nodes and uses them to project short-term network performance based on the requested metrics.

The candidate nodes are then ordered into a list weighted in terms of predicted network performance. Lower weight indicates better predicted network performance. In the example of Figure 5.5, the performance of the network path to node Beta is anticipated to be better than that to node Alpha by a ratio of 7:9. Negative weight signifies lack of measurements for the node pair. In the previous example, the GridMAP service did not find any stored measurements of the path from Omega to Gamma. The scheduler that consumes GridMAP's response could use such negative weight to trigger a single probe in order to obtain an initial measurement for the unmeasured Omega–Gamma path. The choice of probe, if any, is an implementation decision which will be discussed in section 6.5.

5.3.4 Potential Advantages

The GridMAP service supplies a sending host with information about end-to-end connections to remote resources. This reduces the maintenance cost for applications and allows them to respond to changes in network contention by adjusting resource allocation. This process is performed without the cooperation of intermediate network elements (e.g. routers).

Interaction with GridMAP via an OGSA-defined service interface brings great usability and interoperability. Similar grid and Web service-based solutions include DWDM-RAM, perfSONAR, NRM and SONoMA.

Furthermore, the architecture of the GridMAP service allows the collective data archives to be logically available from one source through the service stub. This simplifies data submission and retrieval, and allows advanced analysis (e.g. prediction and pattern recognition) to be performed on the metrics accumulated from different grid nodes. As the GridMAP service is a grid service, the data it stores is easily distributed and replicated. This decentralisation property avoids scenarios of a single

point of failure, ensuring high resilience and availability. Additionally, it enables easy scaling to afford the computational costs of storing and analysing large amounts of measurement data.

The measurements collected by GridMAP can also serve further purposes. For example, they act as ‘health records’ for resources which provide a better insight for troubleshooting, QoS accounting, and SLA verification. They are also helpful for researchers wishing to evaluate grid technologies.

5.4 Functions & Features

This section reviews the key features of GridMAP, partitioned under 5 main functions. The subsection discussing each feature will also mention which requirement from section 4.4 it corresponds to, if any.

5.4.1 Performance Measurement

GridMAP is specifically designed to provide highly accurate end-to-end measurements in a manner that is reliable and readily usable. This sets GridMAP apart from other monitoring solutions developed for grids and for distributed systems in general. The criteria set for GridMAP’s approach to network measurement are as follows.

Non-Intrusive (Requirement II)

First, the collected measurement are to be predominantly passive triggered by generated network traffic, restricting the use of active measurement techniques as much as possible. This is a fundamental feature of the GridMAP monitoring system to avoid the added overhead and distortions caused by active probing.

Accurate (Requirement I)

GridMAP is designed to provide information about the performance of the network. Such information allows schedulers to become network-aware and thus be able to manage the network as a system resource. It is thus very important to provide information that scrupulously reflects the state of the network, regardless of the measurement approach taken.

Therefore, GridMAP does not rely on any estimated measurements of performance. No tomographic techniques are used. In other words, no nodes are assigned the roles of ‘monitors’ or ‘tracers’, nor will any domain-to-domain measurements

be generalised to other members of the same domain. Instead, GridMAP takes a ubiquitous approach where all nodes submit their own end-to-end measurements reflecting the exact network performance they experience.

Reliable (Requirement III)

The measurement is done by listening to ongoing traffic and, hence, is not triggered by the user, application or a component of the grid software stack. Such passive means prevents the measurement from being controlled or influenced by any part of the used grid stack, and is thus deployment-independent and ‘always on’. Moreover, the measurement technique does not employ any unreliable means, such as ICMP measurement. In effect, GridMAP measurements are carried out in a consistent and dependable manner.

Scheduler-Oriented (Requirements I and V)

Network measurements are collected from the vantage point of the sending node. This perspective provides a true representation of the network quality delivered to the grid application and is hence valuable to schedulers evaluating network paths to different candidate nodes.

True Asymmetric End-to-End (Requirement I)

Measurements collected by GridMAP represent the end-host to end-host connection. This eliminates space, and need, for estimations and is thus representative of the end-host to end-host network experience the grid application is likely to get. Furthermore, the lack of dependence on tomographic performance estimations also eliminates, from a grid application’s viewpoint, the need for measuring the performance of individual hops in a path.

Measuring the characteristics of paths in different directions rather than using one-way measurements for both directions is important as asymmetric routing is proving more prevalent than before [373, 273]. It also makes it easily deployable.

Independently and Easily Deployed (Requirement V)

Grid environments are made up of diverse, heterogeneous resources that are typically controlled by a collection of organisations or individuals. This is true of computational, data and service grids. Hence, deploying a new solution in a grid infrastructure,

whether a high-end production grid or an ad hoc collaborative one, promises to be quite a challenge. It is for this reason that the measurement technique used by GridMAP is self-reliant, uncommitted to the cooperation of other peers in the grid. This enables the measurement to be carried out autonomously and easily.

Low Overhead (Requirement VIII)

The passive measurement approach adopted by GridMAP sets out to reduce the overhead and inaccuracy of generating artificial traffic. In the same vein, the overhead of the collected measurements should not grow to an extent that burdens the GridMAP solution or any other grid solution that relies on it. Therefore, GridMAP is selective about the measurements it collects and avoids creating high measurement hindrance.

5.4.2 Data Accretion

Measurements collected by Pasinemda are aggregated via the GridMAP service stub. The advantages to this form of aggregation are as follows.

Single Point-of-Contact (Requirement VII)

The GridMAP service acts as a unique aggregator of collected measurements. Such a feature is an important one in order to simplify the process of submitting and managing information, as highlighted in subsection 4.3.2 and section 4.4.

Extensible (Requirement IX)

The measurements are sent in XML format using the NM-WG schema. This allows for integration with other grid solutions and GISs if needed. It also enables the solution to be expanded later to include a wider range of metrics or more detailed measurements.

Permeative (Requirement IX)

A lot of users of grid applications are members of large institutions. The boundaries of such institutions is more often than not safeguarded with firewalls, NATs, and other boundary enforcement techniques. This is a source of difficulty for cross-organisational grids (c.f. [323, 246, 326]). Pasinemda avoids such difficulties with communication through organisational boundaries by submitting measurements via a standard WSDL-defined grid service stub over HTTP.

5.4.3 Data Management

The information management model of GridMAP is logically centralised but in fact the system is physically distributed to gain the advantage of both arrangements.

Distributed (Requirement VI)

Distribution of the GridMAP solution allows it to scale to handle more amounts and types of measurements, facilitate data redundancy, and avoid scenarios of single point of failure and performance bottlenecks. Therefore, the distributed nature of GridMAP enables scalability, resilience and high availability. It also makes it feasible to afford the computational costs of managing large amounts of measurements.

Logically Consolidated (Requirement VII)

Having a logically centralised structure provides access to all measurements as though they were physically in one location. There are two main advantages to this. First, it provides the valuable benefits of having a single publisher model, which will be discussed in subsection 5.4.4. Second, it grants the chance of performing different sorts of analysis of the collected measurements. For instance, prediction algorithms could be implemented to offer short-term forecasts of performance. Pattern recognition algorithms could also be put in place to provide long-term forecasts and to detect performance anomalies. The outcome of such numerical analysis of the measurements could be used to evaluate the performance of grid applications from a networking point of view, verify SLAs, and detect and troubleshoot inconsistencies.

Expandable

The measurements collected by GridMAP are auto-reported ones as they are obtained by grid nodes about their perceived network performance. This vantage point results in realistic and accurate measurements. It also has the advantage of allowing the GridMAP system to expand both horizontally and vertically to collect more information about the grid. For instance, the nodes could report on more well-defined network metrics. They could also report on the availability of their local computational resources. Hence, the design of the GridMAP system allows it to be taken further to offer more information that could be useful for schedulers.

5.4.4 Information Publishing

A scheduler which requires information from GridMAP would request so through its service stub and the information would be sent back in XML format. Two main advantages of this arrangement are as follows.

Single Point-of-Contact (Requirement VII)

The model adopted is based on a unique publisher of information, i.e. the GridMAP service. The presence of such a single point-of-contact is a highly valuable feature of the GridMAP solution. It provides a clear and identifiable source of information and guarantees data consistency.

Extensible (Requirement IX)

Like measurements, information is communicated in a structured XML format, providing potential for easy extension of published information.

5.4.5 Information Consumption

In order to receive information from the service, a scheduler has to initiate a request. This is a query typically tailored by the scheduler to gain information about certain metrics of specific nodes. The response of the GridMAP system is devised to offer the following advantages.

Configurable (Requirement IV)

The GridMAP service has the ability to respond to queries about different metrics of candidate nodes and from different source nodes. The resulting response is thus highly customised to assist the querying scheduler in a very specific scheduling decision.

Interoperable (Requirements IV and IX)

The GridMAP response is devised to be consumed by schedulers and is thus in machine-readable form, i.e. XML. This facilitates automated information processing without user intervention. Moreover, the fact that the NM-WG schema is an established standard of representing network metrics enables the same data retrieved from the GridMAP service to be consumed by different clients, let alone schedulers,

regardless of their language or platform. For instance, GridMAP could be used by portals to provide graphical representations of monitoring information.

5.5 Summary

This chapter introduced the design of GridMAP, a solution with a broad objective of providing means by which grid schedulers can obtain knowledge of and adapt to changes in the network paths to remote resources. The GridMAP solution is applicable to a wide range of distributed applications, but is particularly important to grids where the required high performance can be hindered by the unpredictability of public best-effort networks.

GridMAP consists of two elements. The first is Pasinemda, which is responsible for measuring the network passively, accurately and reliably. The Pasinemda measurement technique exploits TCP exchanges in the grid: connection setup is used to calculate RTT as the delay between sending a SYN message and receiving its corresponding SYN-ACK response. This RTT measurement includes the real latency of the path as well as the processing delay at the remote end (which serves as an indicator of the node's responsiveness) and the path's queuing delays (which attests to the level of congestion on the path). Pasinemda is light-weight and independent which makes it easy to deploy. Additionally, due to Pasinemda's unobtrusive nature, grid traffic monitoring becomes an involuntary process that requires no intervention but is sustained as traffic naturally passes through a grid node.

The other part of the GridMAP solution (i.e. the GridMAP service) is a grid service that consolidates and makes available all the measurements collected by Pasinemda instances. The service is intrinsically distributed yet operates as one system, which presents a number of advantages. It ensures scalability, resilience and high availability. It enables advanced processing on the collective measurements. It also provides potential to maintain high levels of performance by allowing distribution of load. Moreover, the architecture of the service presents a single point-of-contact scenario. This is a vital advantage as Pasinemda instances and, more importantly, grid schedulers need to contact only one body to report and retrieve network measurements, respectively.

Implementation



GRIDMAP's modular design readily lends itself to a modular implementation based on function. The implementation of these functions is described in this chapter. Section 6.1 details how Pasinemda is implemented. Sections 6.2, 6.3 and 6.4 present the three main modules of the GridMAP service. These are implemented using Java in a GT service container. Section 6.5 reviews how information provided by GridMAP is consumed by schedulers.

6.1 Performance Measurement

Pasinemda, the measuring element of GridMAP, is implemented as a UNIX daemon using C. The daemon makes use of the packet capture library *libpcap* [204] which enables it to listen to ongoing network traffic. On the spawn of new outgoing TCP flows, the daemon creates a dedicated data structure to follow the flow. The daemon logs the times when different TCP flags (e.g. SYN, SYN-ACK, FIN) are encountered. This is used to keep track of the flow's fundamental information, namely the following:

- IP address of destination node,
- time of handshake,
- time of SYN-ACK reply,
- time of teardown request, and
- total number of transferred data bytes.

This information is then used to compute the RTT and throughput of the connection, through the process illustrated by the flowchart in Figure 6.1.

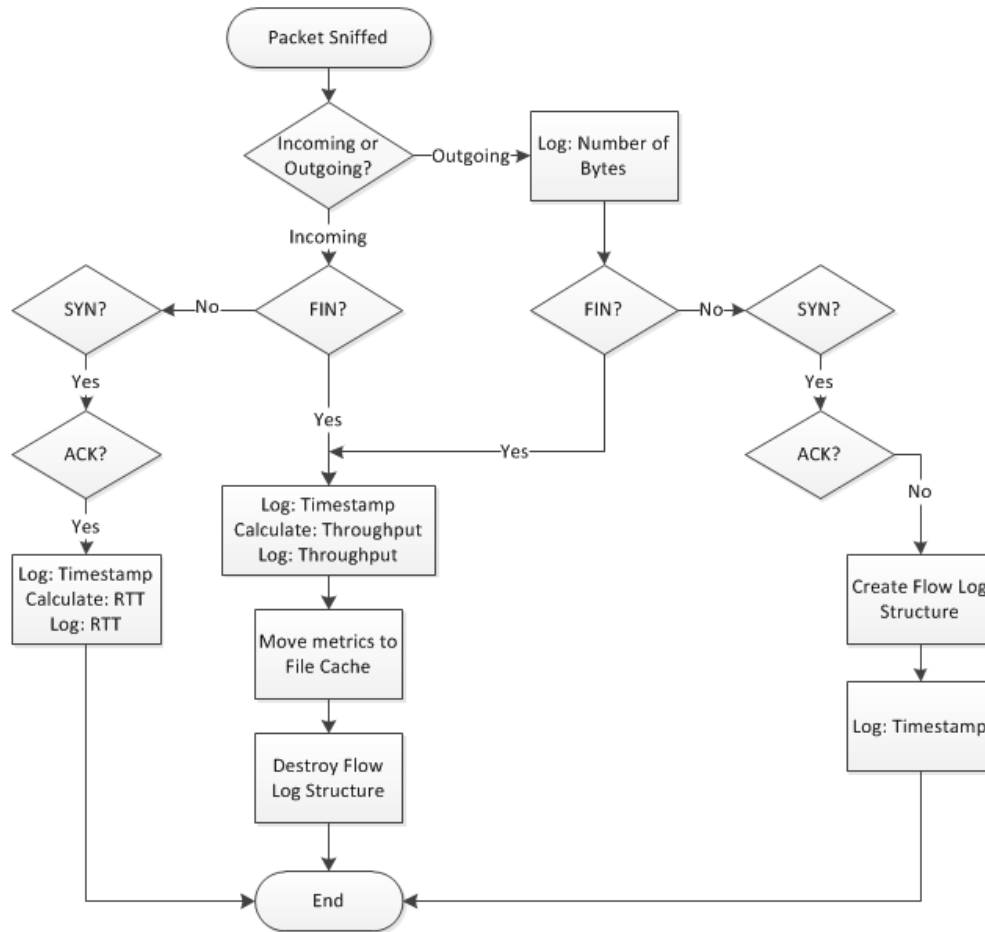


Figure 6.1: *Pasinemda's Measurement Process*

This measurement process runs in the background on the node and hence captures any traffic it generates into the network. The only requirement for this to take place is execution with root privileges in order to allow libpcap to access the network interface. This, however, is only needed on nodes running Pasinemda, i.e. where users are submitting jobs to be scheduled, and not on any other grid nodes. In other words, a grid user who wishes to use GridMAP just needs to run Pasinemda in privileged mode on his local node from which he submits jobs. In the case of e-science grid applications, for example, this is the researcher's desktop or laptop machine. On shared computers, this could be made easier by appropriately setting the access mode of the Pasinemda executable. The same is done with the ping executable, for example, which also requires root privileges to use raw sockets for sending ICMP packets. No peer collaboration is required for Pasinemda to operate and hence there is no need for the user to install or run anything on any other node in the grid.

6.2 Data Accretion

This function relates to the process of amalgamating measurements collected at various grid nodes running Pasinemda into one logical location. The process is initiated by the Pasinemda instances themselves: Pasinemda parses any cached measurements into XML using the NM-WG schema and submits them to the service through its stub. This is carried out every s minutes. The default value of s is 15 minutes, but it could be changed through a command line argument to suit the needs of different applications. Algorithm 6.2.1 summarises this process.

In addition to reporting measurements, each Pasinemda daemon goes through a bootstrapping phase where it contacts the GridMAP service to register the hostname of the node where it resides and the IP addresses of all associated external network interfaces. The Pasinemda instance also generates a client stub to match the service's endpoint. The client stub is regenerated whenever the service interface changes from the last time it was invoked.

Algorithm 6.2.1: REPORTMEASUREMENTS(*FileCache*)

```

Read FileCache
if !(isEmpty(FileCache))
  while sizeof(FileCache) > 0
    do {
      Create new XML record:
        (DestinationIP, TimeOfFlow, TotalBytes,
         RTT, Throughput)
      Parse first flow digest to XML
      Save XML record
      Reduce flow digest from FileCache
      Connect to GridMAP Service (SourceIP)
      Submit XML records
    }
  Sleep for  $s$ minutes

```

On the service side, measurements are received by invoking the Measurement Receiver module. This module inserts the received data into the Measurement Database, as summarised by Algorithm 6.2.2.

```

Algorithm 6.2.2: RECEIVEMEASUREMENTS(XMLRecords)

procedure CHECKHOST(IP)
  SQL ← SELECT unique host identifier with IP address IP
  ResultSet ← Execute SQL
  if !(isEmpty(ResultSet))
    then Host ← Unique Host Identifier
    else {
      comment: Host not running Pasinemda
      Host ← Resolve IP to hostname using host
      if Host not found
        then Host ← IP
      SQL ← CREATE new host: (Host, IP)
      Execute SQL
    }
  return (Host)

main
  SrcHost ← CHECKHOST(SourceIP)
  for each Record ∈ XMLRecords
    do {
      DestHost ← CHECKHOST(DestinationIP)
      SQL ← INSERT metrics using SrcHost and DestHost
    }

```

6.3 Data Management

The entity-relationship model of the Measurement Database is depicted in Figure 6.2 using Crow’s Foot notation. The database is managed using MySQL and connected to using JDBC.

The representation of data using the Measurement Database takes into account that one node might have more than one IP address. It also allows for fast queries for metrics between a combination of source and destination nodes using unique foreign key pairs. The database model is also easily expandable to include, for example, more network performance metrics or local computational resource availability. The latter extension could easily be realised using an additional table as shown in Figure 6.3.

Replication of stored data is achieved using MySQL Replication [369]. In this setup, one copy of the database is specified as ‘master’ while the others as ‘slaves’. Changes to the master database are automatically mimicked in an asynchronous fashion on slave copies to maintain conformity. In our implementation, the GridMAP service

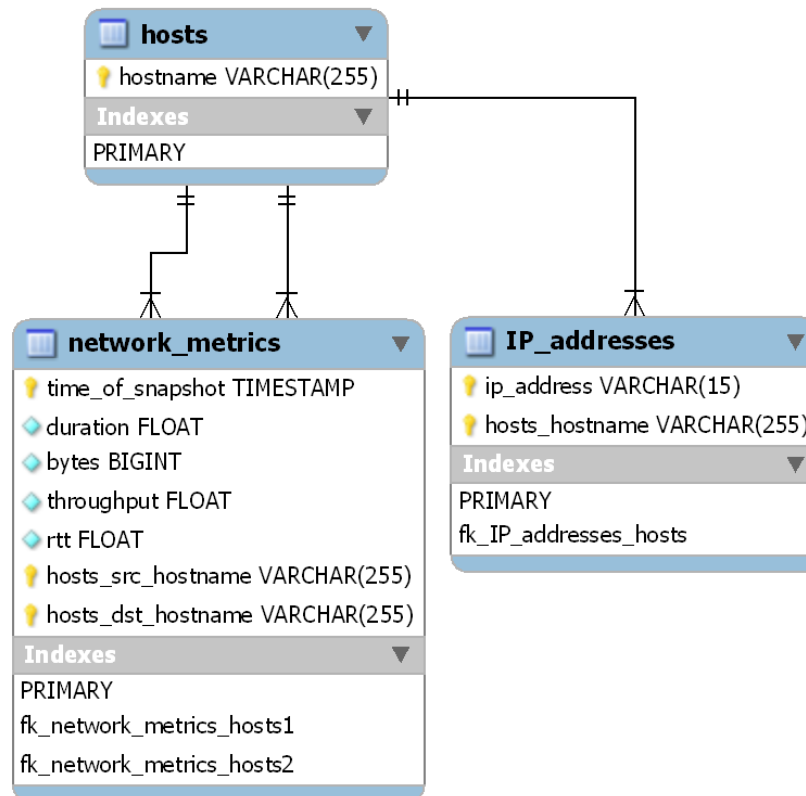


Figure 6.2: *Measurement Database Entity-Relationship Model*

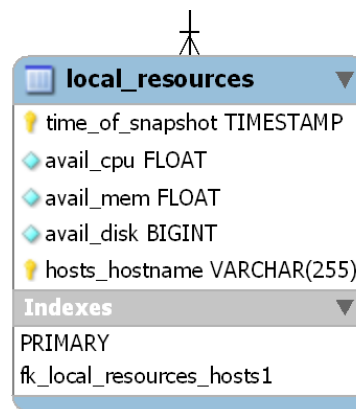


Figure 6.3: *Possible Extension to the Measurement Database Model*

front (i.e. where the service URI points) is specified as master while additional nodes are specified as slave. MySQL provides two modes of replication: statement-based and row-based. Our implementation employs the former as most changes made to the database are via multi-transaction INSERT statements that affect a number of rows in two or three tables. Such replication scheme achieves the target of being logically centralised while being physically distributed.

MySQL Replication was preferred to grid data replication solutions, such as Grid Datafarm (Gfarm) [330] and Globus Data Replication Service (DRS) [106], as these

solutions are better suited for replication of large, fairly static files. In contrast, MySQL transaction-based replication is much more fine-grained and is thus suitable for non-readonly relational databases. Other replication management solutions, such as Storage Resource Broker (SRB) [75], were disregarded due to their large maintenance overhead and compatibility issues with MySQL.

6.4 Information Publishing

So far, the functions presented were quite straight forward as each had a quite distinct assignment. The function of publishing information through the GridMAP service, however, is more complex. This function is broadly split into 3 sub-functions: *service invocation*, *data analysis* and *dissemination*. This section reviews how these sub-functions are implemented.

6.4.1 Service Invocation

A grid scheduler that is about to allocate resources to a job approaches the GridMAP service by method calls to the client stub local to its machine. This stub is generated based on the WSDL description of the GridMAP service, mirroring its interface. The client stub serialises the method invocation and its associated parameters into SOAP and communicates it to the server stub, which performs a deserialisation operation to convert from SOAP back to local method calls.

The remote invocation includes four parameters (mentioned in subsection 5.3.3). These are as follows:

1. *Identifier*: This is the IP address or hostname of the scheduler's node. It is used in SQL queries and to identify service resource states.
2. *Nodes*: This is a list of nodes with resources that satisfy the computational requirements of the job.
3. *Metrics*: The network metrics that are of relevance to this job.
4. *Data Size*: The total number of bytes to be transferred for the job to start executing. This is only specified if throughput is of interest.

6.4.2 Data Analysis

Data analysis is carried out by the Predictor module. When provided with a pair of source and destination nodes and metrics of interest, it retrieves the corresponding measurement history from the Measurement Database and uses it to predict future metrics. We use 48 hours period-based history, and advise not to exceed this value for production grids as long periods can cause severe fluctuation which has a detrimental effect on prediction quality. This, however, is adjustable and is left to the discretion of the user.

This subsection describes the numerical analysis that is carried out by the Predictor module, starting with the requirements then detailing the technique and finally some implementation details.

Requirements

Using the (*SrcIP*, *DestIP*) pair used by the Query Handler module to invoke the Predictor module, the latter retrieves all metrics reported by *SrcIP* about connections to *DestIP*. The returned data set holds a series of Pasinemda-obtained measurements that is discrete, quite possibly irregularly spaced, and potentially sparse and bursty (based on ‘pulses’ of activity). The observations are also expected to be quite precise with very low error margins, based on feature 5.4.1, *Performance Measurement*¹. For the Predictor module to work efficiently, it needs an algorithm that works on such data series to provide accurate predictions. It also needs to do so rapidly in order to minimise overhead and to enable the system to scale. Hence, there is an emphasis on pragmatism in the process of choosing the right algorithm.

Technique

The measurements collected by one Pasinemda instance is a sequence of observations at different times. Such a sequence might thus appear to be a time series as they are potentially dependent observations of certain variables (i.e. RTT and throughput). However, the observations are not obtained at regular intervals but at times when network traffic is generated which might seem, for the purposes of this thesis, random. Hence, the collected measurements do not necessarily form a time series. Consequently, common time series analysis methods (c.f. spectral analysis [209], Box-Jenkins [85], etc.) are not directly applicable here. Even traditional

¹The accuracy of these measurements will be scrutinised in chapter 7.

methods, such as discrete-time Fourier transform, would not be applicable due to the lost samples.

Several techniques have been put forward to analyse and forecast series of irregularly spaced observations, such as weighted wavelet Z-transform [153], parametric clustering algorithms [371], and iterative singular spectrum analysis [226]. Alternatively, irregularity in time series could be dealt with using *resampling* [260]. Here, interpolation is employed to compensate for inconsistent periodicity by approximating the value of the missing samples. This then qualifies irregular series to be used by traditional forecasting methods. We prefer this latter approach for application here as the former alternatives are computationally intensive.

Numerous resampling methods were implemented in the statistical analysis literature to solve different spatial and temporal sample irregularities. Examples include linear, polynomial and spline interpolation methods. Of these, GridMAP requires an interpolation technique that is of low error, has minimum requirements and assumptions about the data, and is computationally feasible. The Akima spline fitting method [55] fulfils these requirements. This nonlinear deterministic procedure is used both for interpolation and curve fitting. It produces a smooth interpolatory function made up of polynomials of order three or less. Such a function is very stable to outliers, much more so than those produced by other spline interpolation methods, such as cubic splines. It also produces very low error and is hence appropriately suited for data with very low noise, such as Pasinemda-obtained network measurements. Moreover, being a parsimonious method, its implementation is of low complexity and requires a single linear application per series, which makes it very computationally efficient.

Applying the Akima spline technique alone provides a full set of regularly spaced values. It allows extraction of accurate growth trends. It cannot, however, be used to produce accurate forecasts through extrapolation. This is because its deterministic nature stresses the global trend which might not be representative of all observations, especially in a potentially bursty series like Pasinemda's. Therefore, we couple the Akima procedure with an ad hoc forecasting procedure, namely weighted moving averages. This is a linear stochastic filter that provides a forecast of network performance based only on past RTT and throughput measurements, with more weight given to recent measurements than older ones. Simple moving average is applied for series with 3 or less samples, and exponential smoothing is applied thereafter. This procedure continuously adapts to new measurements as they become available. It is

also easy to implement and fast to execute.

There are of course alternatives to using Akima splines coupled with weighted moving averages as a prediction approach. The Kalman filter [215], for instance, might seem an obvious method to produce forecasts of discrete irregular data. However, it is suited for data with high levels of noise which is not the case with Pasinemda's measurements. Also, the Kalman filter is implemented recursively and is thus less computationally efficient than Akima splines [185, 354]. Other alternatives include regression analysis (c.f. [100]) and maximum likelihood estimation (c.f. [211]), but these demand a priori information about the observations which is not available and would have to be arbitrated, skewing the analysis process.

Implementation

Let x_i denote a Pasinemda measurement value taken at time t_i , and X represent a series of N distinct such measurements, either RTT or throughput. Hence:

$$X = x_0, x_1, x_2, \dots, x_{N-1}$$

These samples are more often than not irregularly spaced, meaning that:

$$P(t_i - t_{i-1} = t_{i+1} - t_i) < P(t_i - t_{i-1} \neq t_{i+1} - t_i) \quad (6.1)$$

This series is then interpolated using version 0.5-4 of the `akima` package [170] of R [286] to obtain an equispaced series X' :

$$X' = x'_0, x'_1, x'_2, \dots, x'_{N-1}$$

Notice that x_i might be but is not necessarily the same as x'_i .

The forecast for the coming period, \hat{x}_N , is computed as follows. If $N \leq 3$ then simple moving average is applied:

$$\hat{x}_N = \frac{\sum_{i=0}^{N-1} x'_i}{N} \quad (6.2)$$

However, if more than 3 observations are available, exponential smoothing is used:

$$\hat{x}_N = \hat{x}_{N-1} + \alpha(x'_{N-1} - \hat{x}_{N-1}) \quad (6.3)$$

where α is the *smoothing constant* in the range $0 < \alpha < 1$. Alternatively:

$$\hat{x}_N = \alpha x'_{N-1} + (1 - \alpha)\hat{x}_{N-1} \quad (6.4)$$

This is a recursive process where the weights are decreasing at an exponential rate: α , then $\alpha(1 - \alpha)$, then $\alpha(1 - \alpha)^2, \dots$ etc. By continuously replacing previous forecasts into equation 6.4, it is reduced to the non-recursive form:

$$\hat{x}_N = \alpha \sum_{i=1}^{N-1} (1 - \alpha)^{i-1} x'_{N-i} + (1 - \alpha)^{N-1} x'_0 \quad (6.5)$$

The constant α determines the degree by which old observations are dampened and, inversely, the degree by which recent ones are pronounced. This is illustrated by a few examples in Table 6.1.

α	Weight		
	$\alpha(1 - \alpha)$	$\alpha(1 - \alpha)^2$	$\alpha(1 - \alpha)^3$
0.9	0.16	0.032	0.0064
0.6	0.25	0.125	0.0625
0.3	0.16	0.128	0.1024
\leftarrow Recent Samples		Older Samples \rightarrow	

Table 6.1: Examples of Sample Weights used by the Exponential Smoothing Model

The closer the value of α is to 1, the more emphasis there is on the local trend. This is more suitable for series where new values are more dependent on recent changes than the overall average. On the other hand, the closer α is to 0, the more input the model takes from older observations, which is suitable for series where the global trend has a dominant effect on the value of subsequent measurements.

The conventional technique to find the most suitable value of α is trial and error. In our implementation, α is initially 0.3. After the exponential smoothing procedure is carried out, an evaluator of the Mean Square Error (MSE) is stored in state. The value of this metric reflects the accuracy of the forecast and is used to improve the following forecast for the same series.

Finally, the predictions are used to calculate weights that will be used to order the list of candidate nodes. The weights are analogous to the predicted amount of time it will take to transfer all files required by the job to be ready to execute. For jobs interested in RTT only, the weight, w_r is calculated using:

$$w_r = 3\hat{r} \quad (6.6)$$

where \hat{r} is the predicted RTT in milliseconds. The constant 3 refers to the average number of round trips needed to *stage in* the job, i.e. transfer all the files it requires for execution. w_h , the weight corresponding to predicted throughput \hat{h} , is calculated

using:

$$w_h = \frac{d}{\bar{h}} \quad (6.7)$$

where d is the total job transfer size including data, executables and parameters. If both RTT and throughput are specified as metrics of interest, then the weight, w , is calculated using:

$$w = w_r + w_h \quad (6.8)$$

6.4.3 Dissemination

After carrying out the data analysis for all source and candidate pairs, the Predictor compiles the predictions and publishes them back to the scheduler from which the query originated. This is structured as an ordered weighted list of the candidate nodes and the predicted metrics for each. It is parsed in XML format and sent through the stub over SOAP.

By this point, the service has responded to the inquiring scheduler. However, the data created by the invocation is not entirely purged; as grid services are stateful, some state information is retained for use in case of future invocations from the same scheduler. The cached information includes invocation parameters as well as Predictor return values and thread state (such as the last value of α). These remain cached under the scheduler's identifier for a *lease-based lifecycle* [313], i.e. the time for which state information is held. In our implementation, the lease time is set to 60 minutes to reduce caching overhead whilst allowing enough room for schedulers to reuse and build on previous queries. If a new invocation is received from a scheduler with cached state, the Query Handler first checks if any new measurements were reported since the last invocation. If not, the outcome of the last invocation is sent back and the Predictor module is not called. Otherwise, the Predictor module is called with the previous series along with the measurements since the last invocation and the cached value of α to be used as the smoothing constant. Calls to the Measurement Receiver module trigger a reevaluation of MSE and a subsequent modification of α .

6.5 Information Consumption

The function of consuming GridMAP predictions is broken down to the following tasks:

- **Interrupt** the matchmaking process at the appropriate time in order to have enough information to formulate a GridMAP service query.
- **Digest** the information provided by the GridMAP service and use it to influence the matchmaking process. This would be done according to application requirements. So, the weights returned by GridMAP could be used as they are, or alternatively they could be squared or even cubed for applications that are more susceptible to network performance such as replica management applications (e.g. [109]).
- **React** to lack of measurements indicated by negative weights.

The manner in which these tasks are handled is strictly a scheduler-specific issue. This is because for a scheduler to make use of such information it first has to be modified to query GridMAP prior to committing scheduling decisions. It is thus a decision to be made by the developer depending on the available grid software stack. Therefore, this section does not present our specific implementation. Instead, it reviews three different ways in which such an implementation can be achieved. The particular implementation we developed is introduced at the end of this section.

6.5.1 Source Code Modification

The first method of integrating GridMAP into the scheduling process is to modify the source code of the scheduler to invoke the GridMAP service. This includes altering matchmaking methods and forging a client to generate and receive SOAP requests and responses.

This task assumes the availability of the scheduler's source code. Further, it entails a considerable level of familiarity with the code in order to know which parts of the code to change and to avoid tampering with other core functionality. Thus, such effort has the potential to become quite time-consuming. It is also disruptive as the code would need re-compiling after the modifications are made. However, it provides maximum flexibility as the developer can exactly specify what information to provide to the GridMAP service and when, as well as what to do with the returned information.

6.5.2 Runtime Modification

An alternative method is to change the scheduler behaviour at runtime, also known as *hooking*. This task relies on identifying and intercepting method invocations at

runtime and injecting modified calls and data structures.

Such effort requires less knowledge of the intricate inner workings of the scheduler compared to the previous method. Nonetheless, hooking entails a great deal of reverse engineering and sometimes profiling which could make it just as time-consuming as the previous method (c.f. [325]). There is also the possibility of failing to *hook* certain applications (i.e. schedulers in this case) due to the way they were written.

6.5.3 Wrapping

The third method to integrate GridMAP into schedulers is to use a wrapper to veer the scheduling process into consulting GridMAP. This could be done in different ways depending on the scheduler in question but essentially the wrapper manipulates the scheduler by altering job descriptions. For instance, the wrapper could use submitted job descriptions (for example, using ClassAd) to obtain feedback from GridMAP and then submit the modified job description to the scheduler (which would be Condor in the example of ClassAd).

Of the three discussed methods, this third one is the easiest. It requires very little, if any, knowledge of how the scheduler works and it demands a minimal amount of coding. The drawback, however, is less flexibility. Using a wrapper binds the programmer to the constraints of the scheduler interface and thus there is less control on GridMAP invocations and their role in influencing the scheduling.

For our implementation, we prefer the wrapping method for its ease of development and applicability to a wide range of schedulers. In our implementation, the wrapper reacts to negative weights by opening a half-connection to the remote node in question and then immediately closing it. This acquires one initial RTT measurement per candidate without resembling a Denial-of-Service attack. We implement our wrapper, called gSched, for the particular case of the gLite WMS meta-scheduler. This will be used in the performance evaluation presented in section 8.3.3 where the impact of using GridMAP is assessed on a production grid infrastructure.

6.6 Summary

This chapter reviewed the implementation of the five functions of the GridMAP system. Pasinemda was first discussed, identifying how this easily deployable libpcap-based daemon monitors network flows and extracts RTT and throughput accord-

ingly. Second, the function of periodically reporting collected measurements to the GridMAP service was explained. Third, the chapter discussed how the collected measurements are stored, organised, and replicated. Fourth, the function of information publishing was explored. This included the sub-functions of information request through service invocation, performance prediction through data analysis, and finally information dissemination. To produce performance predictions, period-based measurement history is put through a procedure of Akima splines coupled with exponential smoothing. This technique is simple, accurate and robust against outliers, allowing it to work contiguously without the need for recursive computations. Finally, the chapter highlighted the topic of consuming the information provided by the GridMAP service by describing three different methods of achieving this. The flexibility of having these different options is a consequence of GridMAP's service-based model which allows language-independent, platform-independent interaction. Of these three options, we decided to choose the wrapping method for its ease and wide applicability, and implement it in the evaluation of GridMAP's impact on scheduling performance in chapter 8.

Part IV

Evaluation

A problem worthy of attack proves its worth by fighting back.

PIET HEIN

Evaluating Pasinemda

ONE of the main contributions of this thesis is the use of passive monitoring techniques to measure the performance of grid networks. As discussed in section 5.2, this approach offers reliable end-to-end measurements without the need for peer collaboration. It also evades the overheads associated with alternative measurement approaches such as intrusiveness and estimation.

However, it is important to assess the precision of the measurements obtained using the passive approach adopted by Pasinemda, especially considering that many passive measurement techniques are renowned for low accuracy. In this chapter, we describe a number of tests that were carried out in order to establish the accuracy of the Pasinemda measurements. An abridged discussion of the results presented here was published in [137].

The objectives of the experiments are first discussed in section 7.1, followed by a presentation of the experimental setup in section 7.2. Next, section 7.3 presents the findings from the experiments. The overall outcomes are summarised in section 7.4 by revisiting the research questions. Finally, section 7.5 offers concluding remarks.

7.1 Evaluation Questions

The objective of the experiments described in this chapter is to establish the accuracy of the network measurements obtained by Pasinemda. Effectively, this is about scrutinising the accuracy of the RTT measurements in particular. This is because Pasinemda measures RTT as the two-way delay experienced during a TCP handshake.

This, of course, is not purely the inherent path latency but is in fact a gross delay which includes any processing delays introduced at the remote end plus queuing delays experienced en route (recall Equation 5.4 on page 77). Therefore, it is important to establish the accuracy of such a method. Moreover, Pasinemda measures throughput as a variable that is dependent on RTT. This makes the evaluation of RTT accuracy even more important.

The experiments presented here, thus, aim to answer the following questions:

- *How accurate is Pasinemda at measuring RTT and throughput?*
- *Does the accuracy vary for different network connections?*

The outcome of these questions is important as it impacts the quality of the information that flows through the rest of the GridMAP system and is eventually fed to grid schedulers.

7.2 Methodology

A series of tests were conducted in which Pasinemda was used to measure connections of varying distances and quality. Table 7.1 summarises these connections.

In the first test, the source and destination hosts reside in the same building and are connected using Ethernet. In the second test, the destination is a home computer connected to the Internet using commercial DSL. The third test is carried out on a connection that terminates in the Oxford e-Research Centre (OeRC). The fourth and fifth tests involve two intracontinental connections: one to an EMANICS [11] testbed node in Munich, and another in the reverse direction from the University of Innsbruck.

These different connection distances and types are common examples of a wide range of grid environments. For example, collaborative grid applications, such

Name	Source	Destination	Number of Hops	Capacity (Mbps)	RTT (ms)
Ethernet	InfoLab21	InfoLab21	1	100	0.6
DSL	InfoLab21	Lancaster	4	16	8
OeRC	InfoLab21	Oxford	12	100	10
EMANICS	InfoLab21	Munich	15	16	29
Innsbruck	Innsbruck	InfoLab21	17	16	48

Table 7.1: Summary of the Connections Used to Evaluate the Accuracy of Pasinemda

as SETI@home [66], usually run over residential connections like DSL. Other grid applications, however, need to run over networks of much better quality. For example, POV-RAY [34] is an e-science application that typically runs on high speed campus-to-campus connections. The OeRC connection in our experiments is provided by JANET [16] and is the sort of connection that a substantial number of UK-based e-scientists use on a day-to-day basis. For regional multi-institution grid projects, such as LHC ATLAS [4], intracontinental WAN connections are employed. Finally, Ethernet connections are included in the tests due to their widespread use. Grid resources are by definition geographically distributed, but in many circumstances these resources are divided into ‘islands’ where a collection of resources and users are grouped at one site in which case Ethernet is the connective tissue.

The setup in each test is identical: we generate TCP traffic using iperf [338] for 34 different transmission durations (ranging from 1 to 500 seconds). In every test, the Pasinemda daemon sits on the sending node while the destination, running iperf’s discard server, simply acknowledges received packets. We compare Pasinemda’s RTT measurements to those of ping, and throughput measurements to those of iperf.

7.3 Results

7.3.1 Round Trip Time

We set up these experiments such that five ping repetitions are triggered with each iperf probe. At the same time, we exploit the three-way TCP handshake of the iperf probe to measure RTT. We then compare our results to the minimum, mean, and maximum ping measurements. Minimum values are the most accurate as they are a result of minimum ICMP discrimination (see sections 2.3 and 2.5). Mean values are generally accepted as the representative ping measurement and hence are important to include in the results. Maximum values are included to illustrate the range of RTT measurements obtained by ping. Figures 7.1-7.5 illustrate these comparisons.

Several observations stand out from the graphs. First, ping measurements are noticeably unreliable. This is apparent from the great variance between the measurements of the ping repetitions taken at any one probe. Such variance is mainly due to ICMP discrimination. At times of long router queues, ICMP packets are treated as low priority traffic and thus remain enqueued longer than other packets. They can also be rate-limited. Therefore, the minimum RTT value measured by ping is always

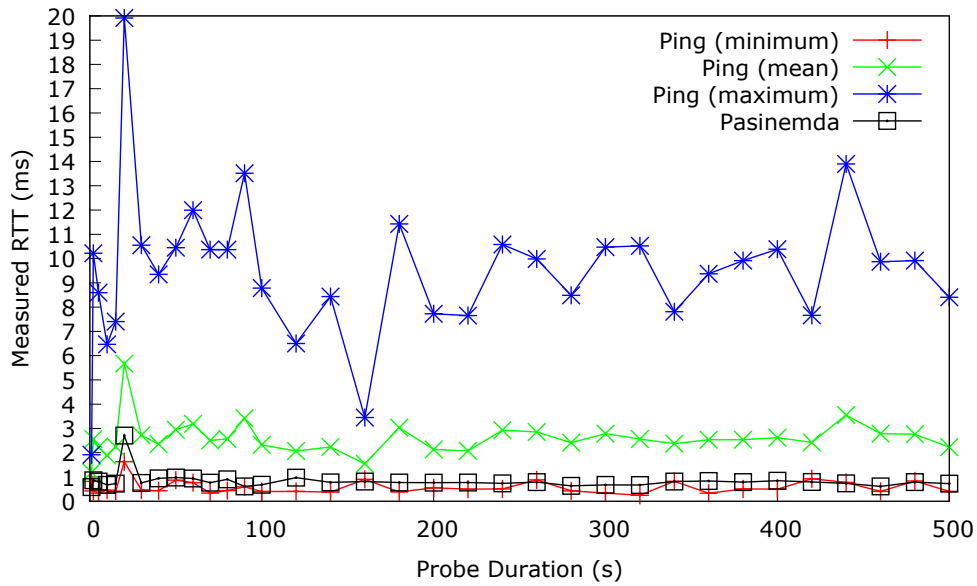


Figure 7.1: *RTT Measured by Ping and Pasinmda on the Ethernet Connection*

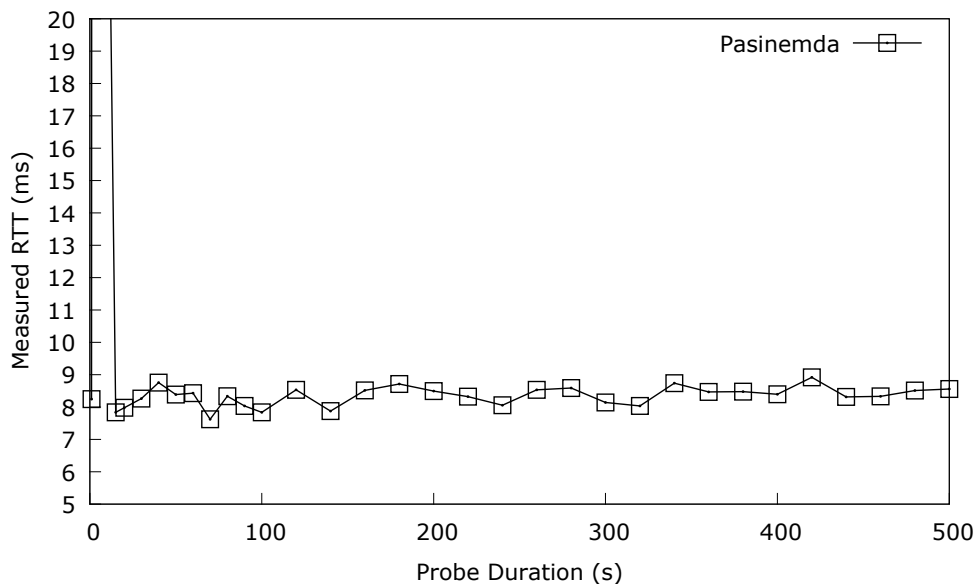


Figure 7.2: *RTT Measured by Pasinmda on the DSL Connection*

the one with the least amount of queue-caused error. On the other end of the scale, the maximum value represents the worst case scenario where the measurement is severely skewed.

Second, despite the fact that Pasinmda relies on only one packet exchange to calculate RTT (as opposed to ping's 5, in these experiments), Pasinmda's measurements are almost always within the range of ping's minimum and mean measurements. Table 7.2 outlines the overall difference between Pasinmda and ping in each of the tests. Pasinmda's measurements tend to be close to the minimum ping values. This

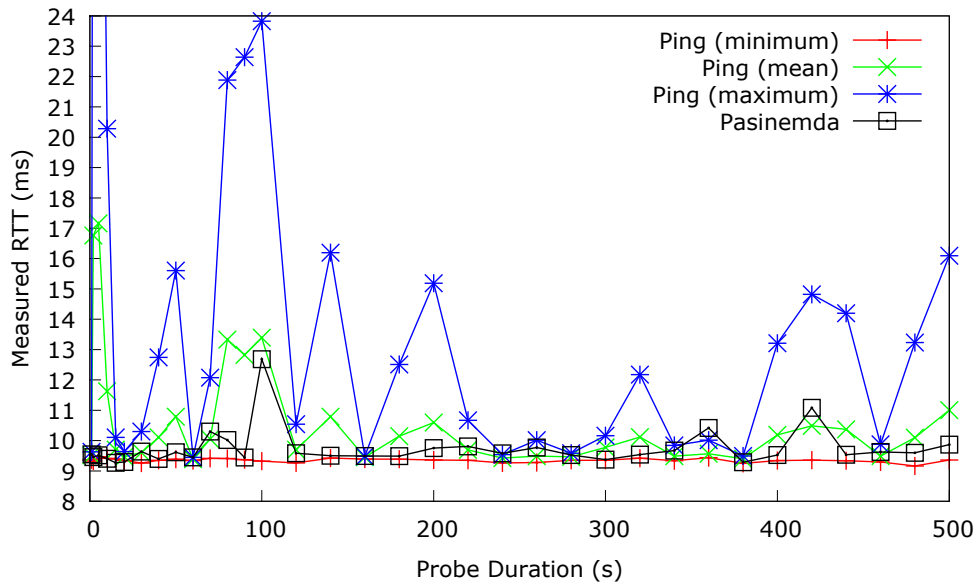


Figure 7.3: *RTT Measured by Ping and Pasinmda on the OeRC Connection*

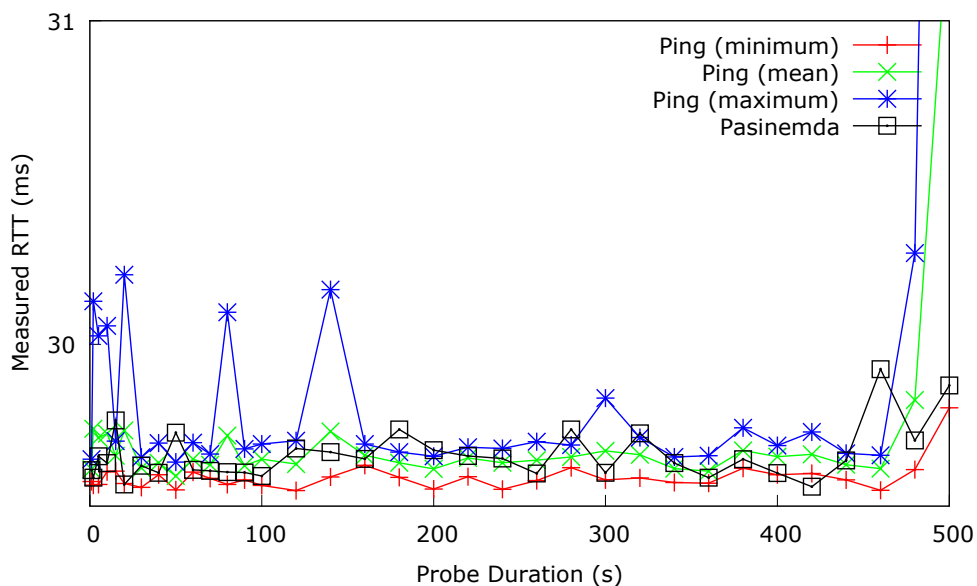


Figure 7.4: *RTT Measured by Ping and Pasinmda on the EMANICS Connection*

effectively means that computing RTT by looking at TCP handshake packets is almost as accurate as the minimum of a series of ping-obtained RTT measurements.

Third, the percentage error is reduced as the connections get longer. This trend is expected as the effect of remote processing delays, which is included in Pasinmda's RTT measurements, is relatively reduced as path latency increases. In order to normalise these differences between the RTT measured by ping and Pasinmda, we plot the ratio between the measurements in Figures 7.6-7.9.

Fourth, the RTT measured by ping fluctuates from one probe to another. This

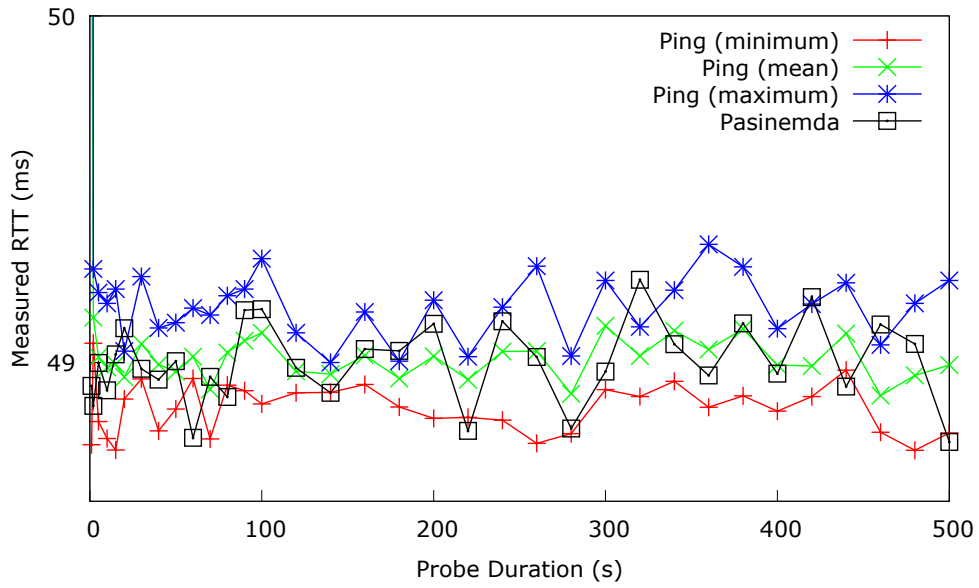


Figure 7.5: *RTT Measured by Ping and Pasinemda on the Innsbruck Connection*

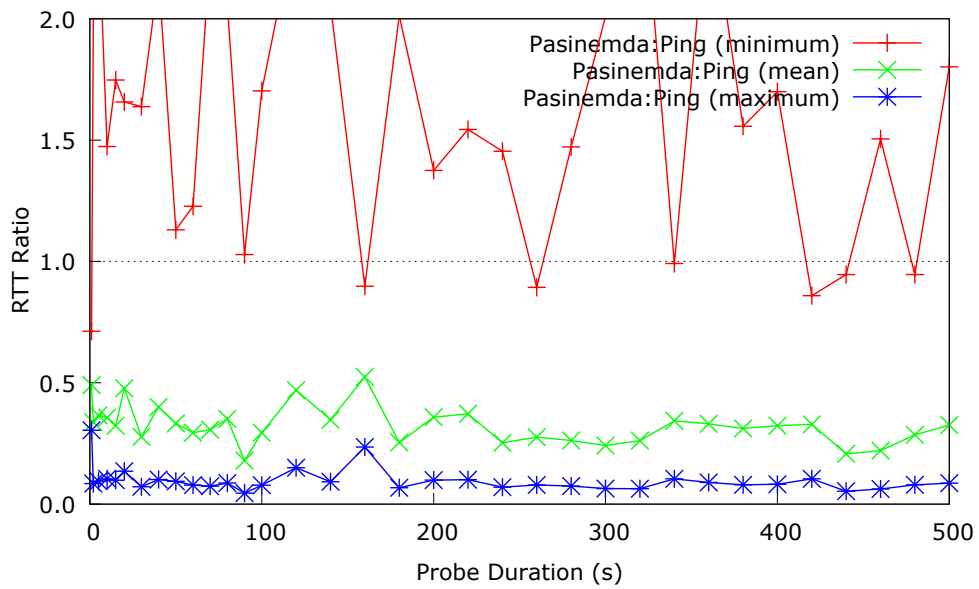


Figure 7.6: *Ratio of the RTT Measurements of Ping and Pasinemda on the Ethernet Connection*

Connection	Percentage Difference		
	Ping (min)	Ping (mean)	Ping (max)
Ethernet	-30.02%	224.15%	1083.70%
DSL	<i>n/a</i>	<i>n/a</i>	<i>n/a</i>
OeRC	-3.65%	9.68%	53.66%
EMANICS	-0.23%	0.18%	1.04%
Innsbruck	-0.27%	0.12%	0.82%

Table 7.2: *Percentage Difference Between Pasinemda and Ping Measurements*

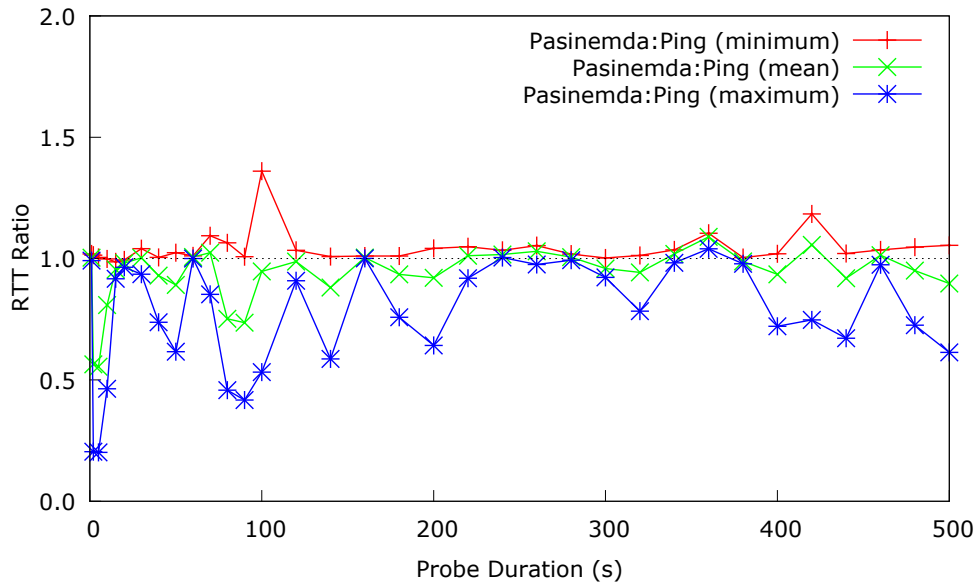


Figure 7.7: Ratio of the RTT Measurements of Ping and Pasinemda on the OeRC Connection

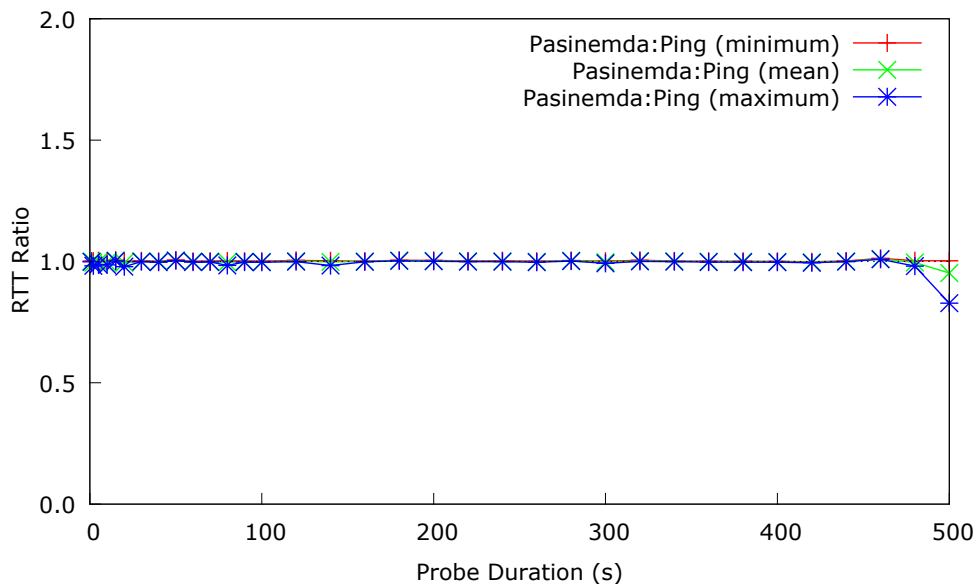


Figure 7.8: Ratio of the RTT Measurements of Ping and Pasinemda on the EMANICS Connection

unsteady behaviour is also attributed to the unreliability of ICMP as a probe carrier. This is made even more evident by the fact that the fluctuations are far more pronounced for the maximum ping values than they are for minimum values. This trend could be observed in Figures 7.1, 7.3-7.9 as well as in Table 7.3 which describes the statistical dispersion of the Pasinemda and ping RTT measurements using their standard deviations.

Other important trends could also be deduced from Table 7.3. The first is the relative stability of the Pasinemda RTT measurements, apart from the DSL test (due

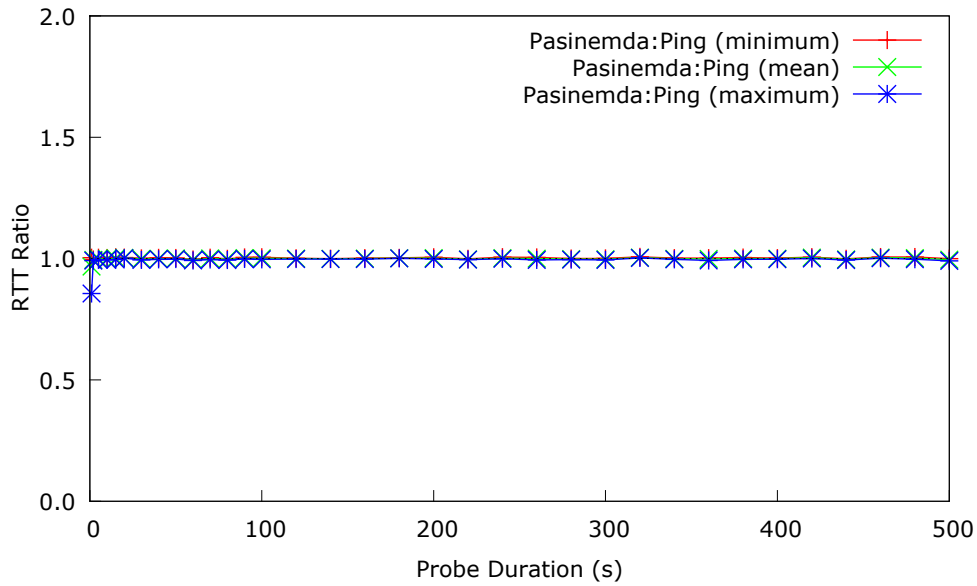


Figure 7.9: Ratio of the RTT Measurements of Ping and Pasinemda on the Innsbruck Connection

Connection	Standard Deviation (ms)			
	Pasinemda	Ping (min)	Ping (mean)	Ping (max)
Ethernet	0.35	0.27	0.72	3.00
DSL	61.06	<i>n/a</i>	<i>n/a</i>	<i>n/a</i>
OeRC	0.63	0.06	1.92	9.03
EMANICS	0.08	0.04	0.30	1.10
Innsbruck	0.12	0.07	0.28	1.37

Table 7.3: Standard Deviations of Pasinemda and Ping Measurements

to high contention). This is reflected by the closeness of the standard deviation of the Pasinemda RTT measurements to that of the minimum ping. This implies that using a natural TCP flow to measure RTT is just as reliable as carrying out a series of ping tests and picking the minimum value. Second, the burstiness of the Pasinemda RTT measurements diminishes on long paths. This is due to the previously discussed effect of processing delay in proportion to path latency.

Taking this one step further requires us to look at the *coefficient of variation* (CV). CV is given by the ratio of the standard deviation to the mean. It provides a way of measuring statistical dispersion of a sample without being attached to its unit, which in this case is milliseconds. Lower CV values signify little fluctuation around the mean value, and vice versa. Table 7.4 displays the CVs for the RTT measurements. These values are quite useful because CV is a normalised statistic and hence allows us to compare the variation of the different samples around their respective mean

Connection	Coefficient of Variation			
	Pasinemda	Ping (min)	Ping (mean)	Ping (max)
Ethernet	0.416	0.479	0.277	0.316
DSL	2.634	<i>n/a</i>	<i>n/a</i>	<i>n/a</i>
OeRC	0.065	0.007	0.180	0.603
EMANICS	0.003	0.001	0.010	0.037
Innsbruck	0.002	0.002	0.006	0.028

Table 7.4: *Coefficients of Variation of Pasinemda and Ping Measurements*

values regardless of the absolute value of the individual measurements. What is acquired from this table is the verification of the aforementioned trends, namely that Pasinemda is just as reliable and accurate as the minimum of a series of ping tests and that Pasinemda's accuracy is improved as inherent path latency increases.

The final observation is again related to the unreliability of ICMP. ICMP packets are not just treated as low priority traffic but are in some cases dropped all together. This is evident from the lack of ping results for the DSL test, as portrayed in Figure 7.2 and consequently in Tables 7.2-7.4, where the commercial network provider's routers blocked all ICMP traffic. Pasinemda, on the other hand, does not rely on ICMP and hence was able to obtain RTT measurements for this connection.

7.3.2 Throughput

In Figures 7.10-7.14 we compare the throughput calculated by Pasinemda against that returned by the iperf client. Because different connections achieve significantly different throughputs, the figures also plot the ratio between the measurements obtained by the two tools.

From the graphs, we establish that Pasinemda's throughput measurements are consistently accurate compared to iperf for all TCP transfers on the five different connections. Overall, Pasinemda's throughput measurements are within 2.20% of the measurements obtained by iperf. Since Pasinemda calculates throughput as a function of RTT, these results are further testimony to the accuracy of Pasinemda's RTT measurements.

7.4 Outcomes

The experiments discussed in this chapter were aimed at answering two main questions. They are now revisited in light of the aforementioned results.

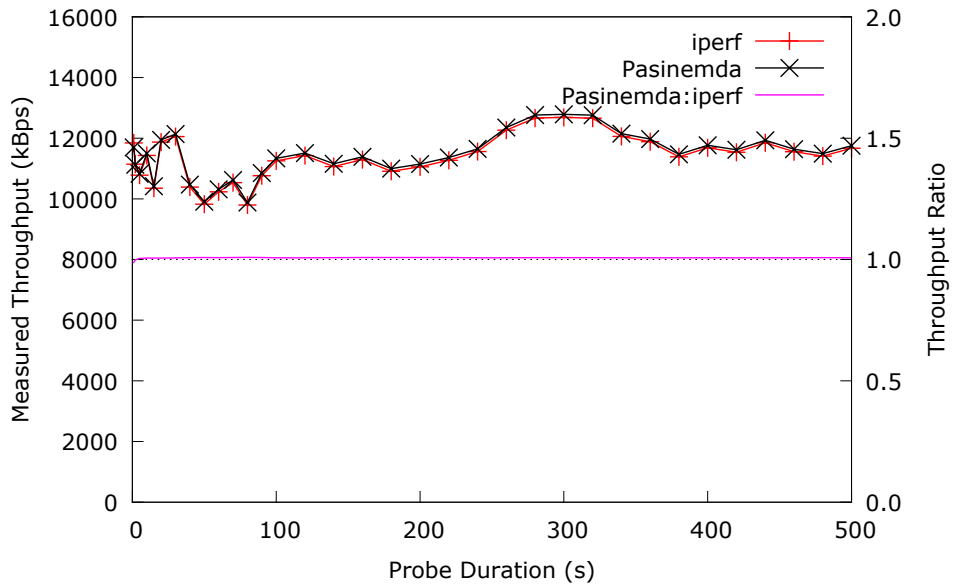


Figure 7.10: Values and Ratio of Throughput Measured by iperf and Pasinemda on the Ethernet Connection

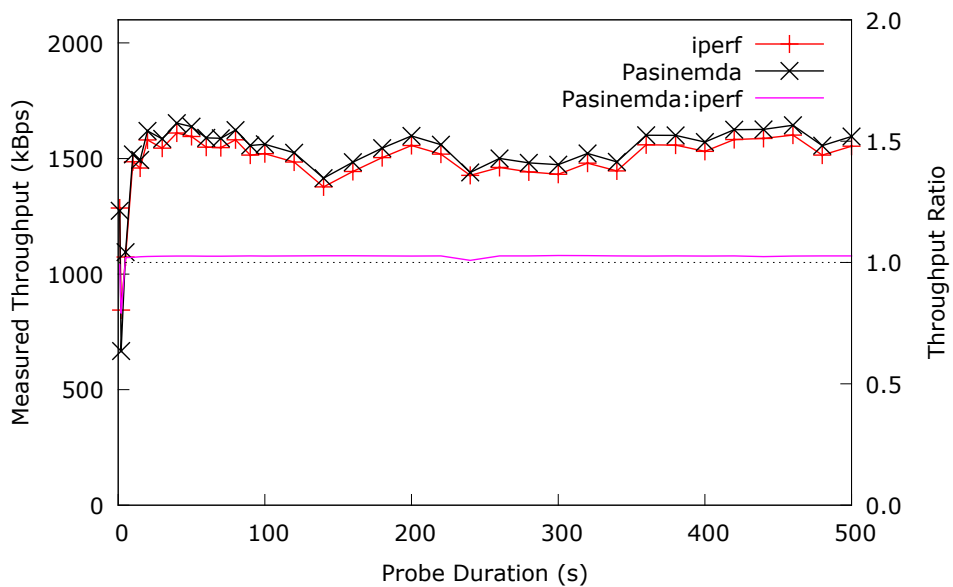


Figure 7.11: Values and Ratio of Throughput Measured by iperf and Pasinemda on the DSL Connection

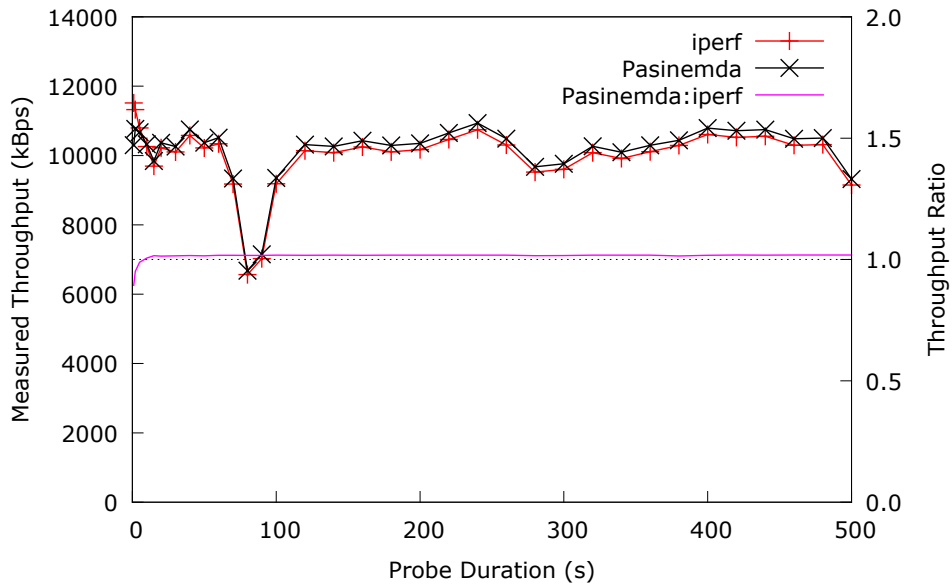


Figure 7.12: Values and Ratio of Throughput Measured by iperf and Pasinemda on the OeRC Connection

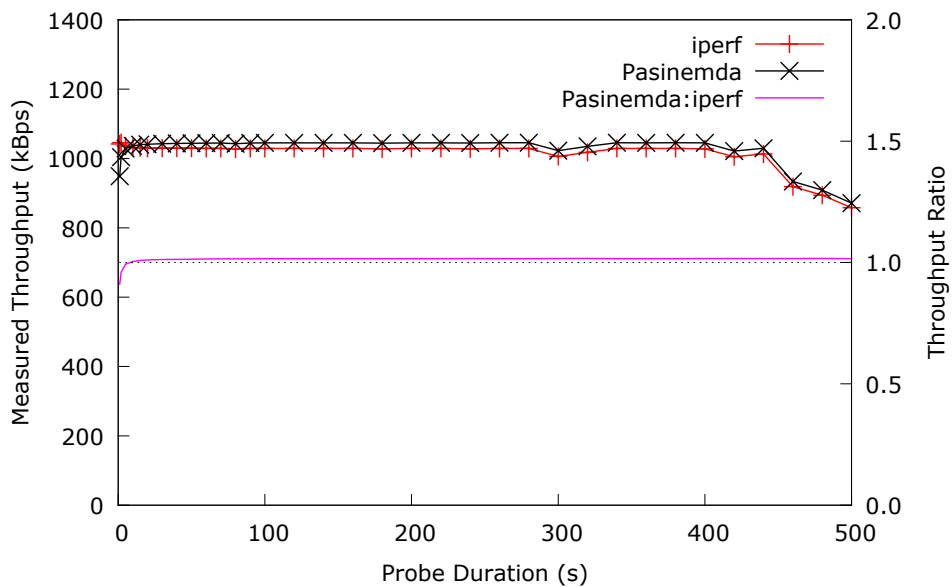


Figure 7.13: Values and Ratio of Throughput Measured by iperf and Pasinemda on the EMANICS Connection

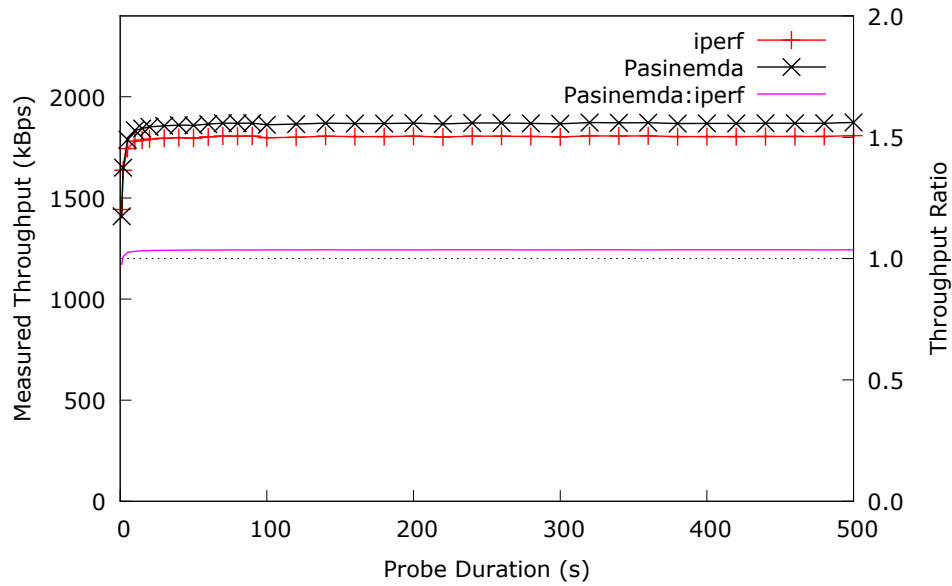


Figure 7.14: Values and Ratio of Throughput Measured by iperf and Pasinemda on the Innsbruck Connection

- *How accurate is Pasinemda at measuring RTT and throughput?*

The experiments established that the Pasinemda technique is accurate for measuring RTT. By merely observing a single packet exchange through TCP handshake, Pasinemda was able to attain almost the same accuracy as the minimum value of five ping probes: within a range of 0.23–0.27% for intra-continental paths, 3.65% for a national high-speed path, and 30.02% for an Ethernet connection. Accordingly, Pasinemda’s throughput measurements were also accurate, within 2.20% on average of iperf’s measurements.

Moreover, Pasinemda displayed reliability as it measured RTT for all connections with minimal variation: the coefficient of measured RTT variation was 0.002–0.003 for intra-continental paths, 0.065 for a national high-speed path, 2.634 for DSL, and 0.416 for Ethernet. This is a significant improvement over ICMP-based techniques which are unreliable and prone to error.

- *Does the accuracy vary for different network connections?*

Pasinemda’s accuracy seems to improve as connection latency increases. This is because as this happens, other constituent delays (i.e. remote processing and en route queuing delays) decrease in proportion to the inherent path latency. This implies that Pasinemda is most accurate with long distance connections such as transatlantic paths and least accurate with low-latency connections such as Ethernet.

7.5 Summary

The core objective of this chapter was to ascertain the accuracy of the RTT measurements obtained by GridMAP's Pasinemda. This is of high importance because it is the basis upon which throughput measurements are calculated. The accuracy of the measurements supplied by Pasinemda is also important because these measurements will eventually be used to influence schedulers and is hence instrumental in GridMAP's offering to grid schedulers.

The accuracy and stability of Pasinemda's measurements was validated through a series of tests against ping and iperf on five different real-world connections representative of common grid conditions. Furthermore, due to Pasinemda's independence of ICMP, it presents itself as a highly reliable purely passive RTT measurement technique.

Evaluating GridMAP

HAVING proven the accuracy levels achieved by Pasinemda, attention is now turned to evaluating the impact of using GridMAP on the quality of scheduling. This chapter looks at the benefits and costs of the GridMAP solution by implementing it in a typical grid environment. Section 8.1 describes the objectives of the experiments and puts forward a set of questions to be answered by the evaluation. Section 8.2 reviews how the GridMAP solution was deployed. Section 8.3 covers the methods used to apply GridMAP to different grid scheduling scenarios. The observed consequences are then detailed in the following three sections: sections 8.4, 8.5 and 8.6 look at network utilisation and scheduling performance, while section 8.7 examines different added overheads. Finally, evaluation questions are revisited in section 8.8, and conclusions drawn from the evaluation are outlined in section 8.9.

8.1 Evaluation Questions

The aim of the set of experiments discussed in this chapter is to answer the following questions.

- I. *Can network utilisation be enhanced by using GridMAP?*
- II. *Can grid scheduling be improved by using GridMAP?*
- III. *Does the effect of GridMAP vary for different grid applications?*
- IV. *What is the cost of using GridMAP?*

Questions I and II are closely related as they both attempt to fundamentally assess the usefulness of employing GridMAP's passively-acquired network measurements to influence scheduling decisions. The hypothesis here is that the use of GridMAP will indeed improve the efficiency of scheduling grid applications and the utilisation of the network. If this is proven to be true, it would be compelling to investigate whether this is true with all or a subset of grid applications and hence Question III. The final objective is to inspect another impact of GridMAP by examining the overheads associated with using it.

The experiments that follow in this chapter apply GridMAP to the scheduling procedure of applications of various characteristics. By gauging the profit or forfeit, if any, of using GridMAP on the scheduling efficiency under different circumstances it would be possible to answer the above questions.

8.2 GridMAP Deployment

This section describes how the GridMAP solution was deployed for the purposes of this evaluation.

8.2.1 Pasinemda

During the experiments, one node is responsible for scheduling jobs. This node runs an instance of Pasinemda which allows it to collect network measurements, based on job submissions and file transfers, and then report them to the GridMAP service. The node also interacts with the service prior to scheduling in order to achieve network-aware resource allocation.

This node resided in InfoLab21 on the Lancaster University campus. It was a Dual-Core AMD Opteron 2.2GHz with 2GB RAM running CentOS Linux version 5.3, JDK 6 update 16 and GT version 4.2.1.

8.2.2 GridMAP Service

During the experiments, the GridMAP service was deployed on four computers provided as virtual machines hosted on the EmuLab testbed [357]. Each of these computers was an Intel Xeon 3GHz with 896MB RAM and 3GB disk space running Red Hat Linux release 9, JDK 6 update 16 and GT version 4.2.1. The data managed by the service is distributed and replicated across these four nodes. The database back-

end was managed by MySQL Community Server version 5.1.42. Data analysis was carried out using R version 2.9.2.

During the experiments, the GridMAP service will be invoked from the host located in the Lancaster University campus. EmuLab is located in the University of Utah in Salt Lake City, UT, USA. This makes the service reachable from Lancaster via a 20-hop connection of 180ms RTT on average. The decision to host the service on such a relatively remote node was deliberate in order to provide a robust evaluation of GridMAP's added value by creating realistic network conditions.

8.2.3 Induced Load

The deployment of the GridMAP service was augmented by a dummy deployment put in place to generate network, storage and processing overhead on the service during the experiments.

This dummy deployment included 20 more virtual machines, also hosted on EmuLab, running Pasinemda. At node creation time, every node randomly selects a number s between 5 and 60. This represents the number of minutes before which Pasinemda reports its cached measurements to the GridMAP service. The nodes exchange a dummy data set (10MB in size) using GridFTP in the following manner. Each node issues a query to the GridMAP service, transfers one of the data sets to one of the other 19 nodes, selected at random, and then sleeps for an interval chosen randomly between 0.1 seconds and s minutes. All of the 20 load-inducing machines were of identical configuration: Intel Pentium III 850MHz with 512MB RAM running Red Hat Linux release 9, JDK 6 update 16 and GT version 4.2.1.

This additional arrangement subjects the GridMAP service to relatively significant load levels in the form of metric submissions and queries. The response of the queries are not used to influence the file transfers. They just serve the purpose of triggering a process of data storage, data acquisition and analysis on the GridMAP service. The Pasinemda submissions based on the interactions of the 20 nodes also serve to synthesise a realistic network and processing load on the service.

8.3 Methodology

This section describes the experiments that were devised to measure the impact of using GridMAP on scheduling grid applications. In doing this, the section discusses four main aspects of the experiments:

- I. The choice of grid testbed and software stack to integrate GridMAP into.
- II. The choice of grid applications.
- III. The means of integrating GridMAP into the chosen grid system.
- IV. The principal metrics of the evaluation.

8.3.1 Grid Testbed

This subsection reviews the choice of testbed and the consequences of this choice.

Real or Simulated Testbed

The first principal decision regarding the testbed is whether to use real or simulated networks. There are several advantages to simulation, most important of which is retraceability. The controlled nature of simulation environments also makes it easy to control environment variables and replay scenarios. This is indeed valuable when compared to deploying a solution over strictly governed real infrastructures.

Nevertheless, controlled environments offered by simulation are of little value if they do not properly recreate the ecosystem they represent. Grids are made up of a collection of highly sought after resources interconnected by public IP networks. Both of these aspects are extremely difficult to recreate. Various grid performance simulation models exist, such as [341, 89]. However, realistic and fine-grained simulation of production grids is not yet possible [188]. Similarly, simulating public IP networks is very difficult [274, 151]. Thus, it is found that simulating grid infrastructures would be more troublesome than rewarding.

Furthermore, our goal is to carry out rigorous evaluative experiments to test the value of GridMAP. This is best done under realistic circumstances where computational and network resource availability is dependant on a wide range of system variables and thus potentially volatile.

Choice of Testbed

Committing to real-world testing, the question switches to the choice of network infrastructure. The requirements for an ideal testbed to run the experiments are as follows:

- I. The grid should be a production infrastructure and not an experimental testbed. This is to test GridMAP under real operational conditions.

- II. The grid has to be used by real applications. This is discussed in subsection 8.3.2.
- III. The grid has to have an advanced grid software stack as it is desirable to test GridMAP in a state of the art environment.
- IV. There should be room to use modified schedulers in order to integrate GridMAP into the scheduling process.

Most grids that satisfy condition I will consequently satisfy conditions II and III. However, no one testbed could be found that immediately satisfies all four conditions as all production (and even experimental grids) implement stringent administrative policies to prohibit tampering with the software stack and listening to traffic.

To resolve this, the National Grid Service (NGS) was chosen as an infrastructure that satisfies the most of these conditions.

NGS Configuration

Background: The NGS is a production grid infrastructure put in place to aid academics and researchers around the UK in deploying and running grid applications. The NGS provides access to a large number of computational resources and scientific instruments all interconnected by SuperJANET5, the UK's academic network backbone, operating at 10Gbps. The basic NGS virtual organisation (VO), called *ngs.ac.uk*, encompasses resources at five sites. Resources at 20 more sites are available for members of other VOs.

The NGS fulfils conditions I, II and III. For all practical purposes, the NGS also satisfies condition IV as it allows its users to invoke Globus GRAM to fork jobs on remote sites and in that sense it does allow the use of improvised scheduling solutions. In spite of this, practical implementation of such handmade schedulers is hampered by various authentication and information retrieval difficulties and is thus quite limited.

In addition, the NGS has a firm policy against granting root privileges to users. We therefore turn to wrapping instead of source code modification as a way of bypassing this difficulty and that associated with using modified schedulers. This will be discussed in subsection 8.3.3.

Structure: Like many production grids of such scale, the NGS follows an 'islands of grids' model. Partner and affiliate sites are supercomputers, clusters or campus grids

that are locally governed *Local Resource Managers* (LRMs). The sites are interconnected via the SuperJANET5 backbone. This configuration, portrayed in Figure 8.1, imposes very little restrictions upon intra-site setup and hence offers valuable site self-governance.

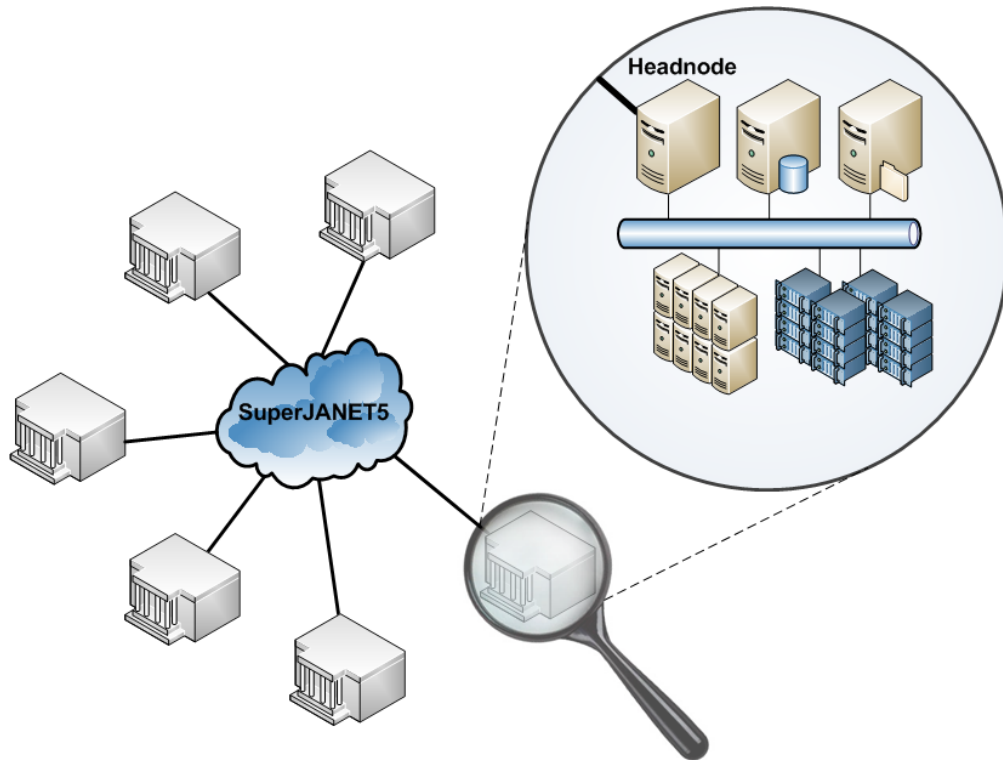


Figure 8.1: *The NGS' Islands of Grids Configuration*

One of the few rules of this islands of grids formula is the allocation of a dedicated host, referred to as the *headnode*, to act as an access point to the site resources. Behind the headnode lies a number of *worker nodes* that are used to execute jobs and store data. On many NGS sites, worker nodes are not directly accessible from outside. Instead, internal resources are made available through interaction with the LRM via the headnode. The islands in this configuration are, therefore, hierarchical clusters (as introduced in subsection 3.1.2).

Software Stack: A grid software stack is a set of solutions put in place in order to deliver the capabilities of the infrastructure to the application level. This includes but is not limited to the OS, middleware, scheduler, directory services, information services, and data management solutions. The NGS software stack comprises primarily of Linux, GT and gLite. Other software components supported by the NGS are presented in appendix B.

Coordination between LRMs of different NGS sites is done by the gLite Workload Management System (WMS) [67]. Based on the requirements of a submitted job, WMS picks the site that potentially best suits the requirements and hands the job to the respective LRM. The LRM then takes responsibility of scheduling the job over local resources. WMS is thus called a *meta-scheduler*. From this time forth, gLite WMS will be referred to simply as ‘the scheduler’ while LRMs will retain their reference.

Limitations: The use of the NGS has two main limitations. First, NGS users access the gLite-managed infrastructure via a dedicated node called UI-WMS. Through UI-WMS, authenticated users could issue CLI commands to submit or cancel jobs, check their status, and acquire output files. Specifics of the gLite WMS operations are detailed in appendix C. However, users are not allowed root privileges on UI-WMS, meaning that running Pasinemda is not possible. Section 8.3.3 will describe how this is resolved.

Second is the restriction of user access beyond site headnodes. This follows from the islands of grids model, which undeniably provides great flexibility to the management of local resources and is thus adopted by almost all production grid infrastructures (including EGEE, GILDA, and various national-level grids e.g. INFN Grid and NAREGI). For the purpose of this evaluation, this means that the measurements that can be carried out are of the connection to the headnode only. This is sufficient to describe the overall connection to the worker nodes that would eventually carry out the job as the last and only unmeasured hop is a Gigabit Ethernet connection the quality of which is better than the rest of the end-to-end network path.

8.3.2 Grid Applications

This subsection highlights case studies chosen for the purposes of this evaluation.

Application Classes

The decision of grid application is an important one due to the paramount effect the application’s characteristics have on its network behaviour and consequently on GridMAP’s potential added value. In order to be able to answer evaluation question II in section 8.1, it was decided to choose a number of grid applications as case studies to represent three of the main classes identified in the taxonomy of grid applications presented in section 4.1. These classes were identified as the most common ones, particularly in the e-science branch of grid applications, and they are as follows.

The **HTC** class includes a large host of applications used in astrophysics, bioinformatics, biomedicine, chemistry, and engineering research. These applications carry out considerably profuse computations based on relatively small input files. Such computations are embarrassingly parallel and more often than not of a strictly linear execution fashion. Examples include AutoDock, BLAST, CHARMM, CosmoMC, Exonerate, GADGET2, GROMACS, and POV-RAY.

HPC applications are quite similar to their HTC counterparts in that extensive computations are performed based on a set of input parameters. Thus, they too are CPU-intensive and not largely data-intensive. However, the execution nature of these computations is not linear, but parallel over a number of nodes with significant interaction between the different processes. Applications such as Amber, DL_POLY, FASTA (parallel mode), Gaussian, mpiBLAST, MrBayes (MPI mode), NAMD, Firefly (previously known as PC-GAMESS), and Siesta are extensively used in bioinformatics, chemistry, and particle physics research.

Data Grid applications are those where large datasets are sent across the grid. Such a scenario is fairly common with all e-science applications where files containing collected data, simulation results, etc. need to be processed or stored at remote sites for later reference. Examples include data management applications, such as OceanStore, as well as data analysis applications, such as R.

Real or Artificial Workloads

Another decision here is whether to use real applications or synthetic workloads, such as [164, 198, 199], to resemble them. In a way, this decision is similar to that of the network testbed. In order to use a synthetic workload, models need to be created to truly represent the characteristics and behaviour of real grid applications. Considerable effort and time is required here in order to assemble a workload and then ascertain that it soundly represents the corresponding real application. In such a case, it is easier to use the real application itself.

Case Studies

One grid application from each of the aforementioned classes was chosen as a case study. These applications had to be ones that are deployed at more than one NGS site. Applications deployed at only one site (such as CosmoMC, Exonerate, FASTA, GADGET2, MrBayes, etc.) have no need for GridMAP as submitted jobs are inevitably

handed to the LRM of that site.

The application selected in each case study represents a number of grid applications of similar characteristics and requirements that have the potential of improving their performance through the use of GridMAP. The chosen applications are now presented.

Case Study A: AutoDock AutoDock [196], developed by The Scripps Research Institute [43], is a suite of open-source grid applications distributed under the GNU General Public License. It is very widely used in biomedicine to explore chemical prospects. AutoDock simulates molecular docking, providing foresight into the binding of molecules to the receptors of three-dimensional structures, such as proteins. For instance, the FightAIDS@Home [197] collaborative grid computing project uses AutoDock to analyse HIV-1 protease binding inhibitors.

Application Characteristics: AutoDock's expected input is a number of files that specify different parameters that control the docking model. These include the atomic structure of the docking ligand substrate, as well as a map of the composition and behaviour of the three-dimensional structure. The parameter files are a few kilobytes in size, but are used by the AutoDock suite to create atom-specific maps needed for remote execution. In total, AutoDock's input files do not usually exceed 10MB in size.

Using this input, AutoDock explores the possibilities of binding the ligand to the target protein. This is achieved by executing a number of CPU-intensive simulations on a single machine. Upon completion of the docking simulations, the application produces a report detailing the different docking scenarios that were undertaken and the resulting molecular interactions. This report is typically under 100kB in size.

Typical Usage: Computational docking is very important for combinatorial chemistry and biomedicine research as it provides a means of battling drug resistance. For instance, the HIV viral protein has an ability to mutate rapidly even in the presence of inhibiting drugs. This process produces mutants that are unaffected by drugs that were designed to target the previous generation of the virus. As a result, the efficiency of drugs against the HIV virus is extremely limited. To produce effective drugs, hence, a very wide range of viral proteins needs to be modelled and a tremendous amount of simulated dockings to the widest array of known potential inhibiting compounds¹ need to be carried out. This extremely hurried trial-and-error process in an attempt

¹For example, the ZINC database [200] contains more than 13 million potential inhibitors.

to surpass the mutation of the viral protein is known as *virtual screening* [306] and typically requires thousands of docking simulations per day.

Similar Applications: The requirements of the AutoDock application represents those of a large number of widely used e-science applications such as other docking tools like CHARMM, DOCK and FlexX, as well as other simulators of molecular dynamics like BLAST, EMBOSS and GROMACS.

Version, Sites and Input: The AutoDock version used in the experiments is version 4.0.1 which is installed at three NGS sites: *ngs.leeds.ac.uk*, *ngs.oerc.ox.ac.uk* and *ngs.rl.ac.uk*. The input files used are provided as examples within the AutoDock suite, and the chosen docking algorithm is the Darwinian genetic algorithm.

Case Study B: Amber PMEMD Like AutoDock, Assisted Model Building with Energy Refinement (Amber) [95] is developed by The Scripps Research Institute. Amber is a commercial suite of about 50 molecular dynamics simulation programs. Perhaps the most common of these programs is Simulated Annealing with NMR-Derived Energy Restraints (Sander) which is a parallel Fortran 90 program used for replica-exchange, thermodynamic integration, and ‘potential of mean force’ calculations. Sander assigns certain atoms to different processors, with inter-processor communication done using MPI. Particle Mesh Ewald Molecular Dynamics (PMEMD) [128], another program in the Amber suite, is essentially most of Sander’s functionality rewritten to improve performance by minimising inter-processor information exchange.

Application Characteristics: Typical input to the PMEMD program is a configuration file that specifies parameters of the molecular simulation along with other files to detail the exact structures and properties of the molecules. Hence, similar to AutoDock, PMEMD input files are fairly lightweight, hardly ever exceeding 10MB. However, the two applications, and their categories, differ greatly from this point onwards. PMEMD employs MPI to execute as a number of tightly-coupled CPU-intensive processes which, according to the simulation configuration and the utilised hardware, could last up to a few hours per process.

PMEMD’s output is a summary of the molecular interactions including the energies, coordinates and velocities of the atoms in the simulated system. The size of these files obviously depends on the properties of the molecular system: the number of atoms, their initial coordinates, velocities, force fields, etc. However, the output is usually small in comparison to the input files.

Typical Usage: Like AutoDock and other simulators of molecular dynamics, the majority of the applications in the Amber suite including PMEMD are *parametric applications* (also called *parameter sweep applications*). This means that they run a number of jobs with different variations of the environment parameters in order to study the different possibilities of, in this case, molecular interactions. Typical use of such parametric applications involves a large number of jobs (1,000s-100,000s) to cover all possible scenarios.

Similar Applications: Many other applications used for simulating molecular dynamics in particular and e-science in general share many of PMEMD's characteristics. Examples include DL_POLY, Firefly, Gaussian, and mpiBLAST.

Version, Sites and Input: In the experiments, we use Amber version 10.0 which is installed at only two NGS sites: *ngs.leeds.ac.uk* and *ngs.rl.ac.uk*.^{II}

Case Study C: LogStore In order to investigate the effect of using GridMAP on data grid applications, a simple application was developed to use GridMAP to archive syslog [242] files. The files contain records of system events and notifications obtained from a group of computers. The files are compressed and sent to a grid node where they are uncompressed, sanity checked, and then indexed in an SQL database. In that regard, LogStore is similar to NetLogger [335] and Rsyslog [35], only much more simplified and adapted for the grid through the use of gSched (presented in subsection 8.3.3).

Application Characteristics: The monitored computers were 32 virtual machines hosted on EmuLab. The computers were used over a period of 11 months to carry out various functions including prototyping, data analysis, and hosting the GridMAP service. During these processes, the virtual machines periodically reported their system logs to a shared repository.

Typical Usage: Data grid applications are very widely used in e-science to process, store and replicate datasets obtained from experiments, simulations, or other processes. Their usage considerably varies according to the size of the datasets and the required processing. An application such as LogStore could be used on a day-to-day basis (for instance, to detect system anomalies [182]), or in an on-demand fashion for troubleshooting purposes. Additionally, it could be used with datasets that are megabytes in size, as with e-sociology participation surveys and e-

^{II}The author would like to thank these sites for providing access to the Amber application suite.

commerce investment portfolios, or hundreds of gigabytes, like the LHC experiments and geodemographic models.

Similar Applications: LogStore is a representative of an array of grid applications that send datasets to be processed and archived. This includes data archiving and replication applications like GDMP, OceanStore, Reptor, and Sprite.

Version, Sites and Input: The LogStore application requires the use of Awk, MySQL and Tar on the remote node. The wide availability of these software packages allows LogStore to be executed on all NGS nodes that are part of the NGS VO, namely those at the following sites: *ngs.leeds.ac.uk*, *ngs.oerc.ox.ac.uk*, *ngs.rl.ac.uk*, *ngs.wmin.ac.uk*, and *vidar.ngs.manchester.ac.uk*.

The size of the individual logs depends on the magnitude of node activity. On average, one day's worth of logs is around 42,800 entries which amounts to just under 4MB of clear text. After compression, this is reduced to approximately 330kB. The total size of the repository collected over 11 months is 110.66MB.

8.3.3 Implementing GridMAP in the Context of the NGS

gSched is a wrapper developed to consume GridMAP predictions. It is specifically implemented to integrate GridMAP into the WMS-based NGS scheduling process. The use of *gSched* indirectly bypasses the prevention of network measurement, which is a common administrative restriction of production infrastructures such as the NGS. This subsection comments on the functionality and properties of *gSched*.

Overview

gSched is implemented to play the role of a user submitting jobs to the NGS. *gSched* resides on the Pasiemda host described in subsection 8.2.1 (which is outside the NGS infrastructure) just as a user of the NGS would. From this location, *gSched* is able to log into UI-WMS and site headnodes. From there, *gSched* issues CLI commands just as a user would to inquire about the state of resources, and manage jobs and data.

Default Scheduling Procedure

To submit a job, *gSched* could take one of two courses of action. One course is to push the job description and input files (containing data, parameters, etc.) directly to *gLite* WMS via the UI-WMS node. This setup is depicted in Figure 8.2. In doing this, *gSched* is involving the full scheduling functionality of WMS by relinquishing all scheduling

control to it. This setup does not allow gSched to modify scheduling decisions to account for the state of the network.

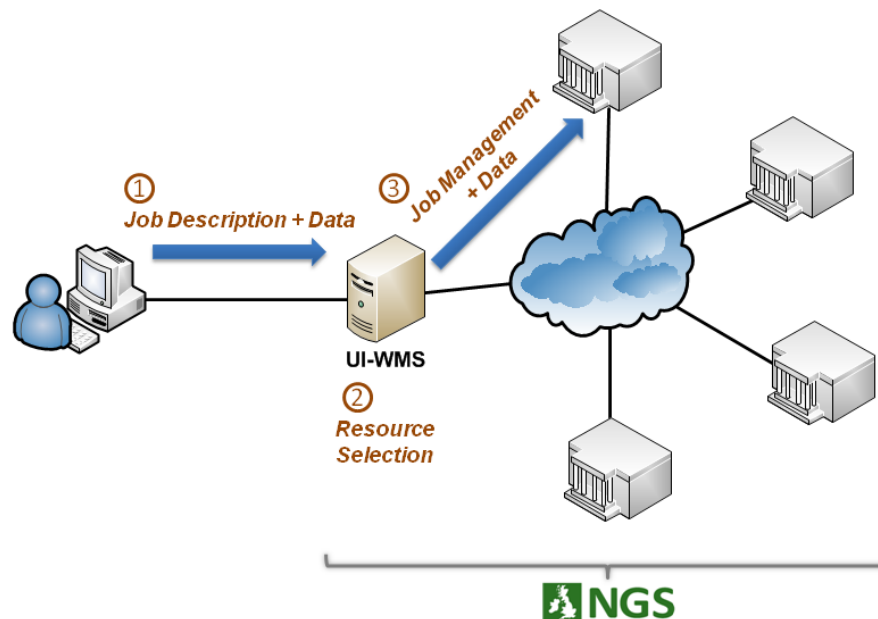


Figure 8.2: Scheduling Process using gLite WMS

The alternative procedure is to question gLite WMS about the best sites to schedule to for a particular job description. gSched would then use WMS' response to deliver input files to the remote site and then signal WMS to tend to the job execution from that point onwards. This alternative course of action is presented in Figure 8.3.

Here, gSched employs the available software stack to obtain information about grid resources. This information could then be used by gSched to create its own scheduling formula. By doing this, gSched devolves some of the scheduling responsibilities from WMS. gSched could thus be described as a *workflow manager* or an *application-level scheduler*.

Network-Aware Scheduling Procedure

The use of gSched facilitates network-aware scheduling by allowing scheduling decisions to accommodate the state of the network as a control variable. This scheme of consuming GridMAP's information is now described.

gSched begins with a template job description that resembles what would be sent to UI-WMS if all scheduling responsibilities would be delegated to gLite WMS. This job description is used to inquire about the resources that potentially best satisfy the

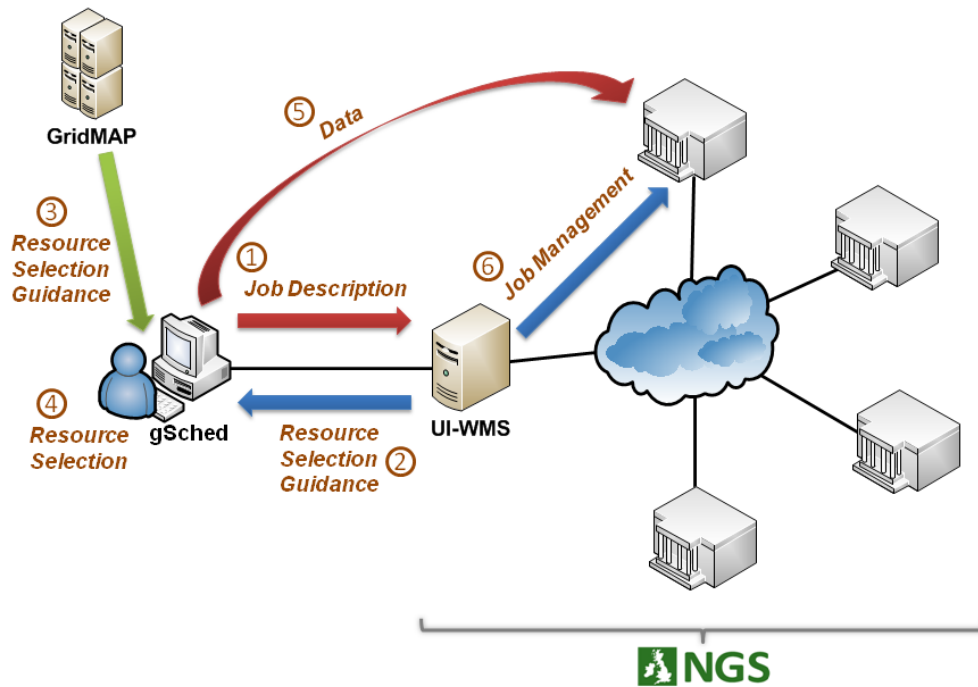


Figure 8.3: Scheduling Process using gSched

job's computational requirements. The response is an ordered list of such sites. The nodes in this list are assigned weights according to their order, then passed to the GridMAP service. The GridMAP service returns a weighted list reordered according to the expected health of the candidate connections. Respective node weights from both lists are then added. The node with the lowest aggregate weight is chosen as the best match. If more than one node share the same aggregate weight, the one ranked higher by GridMAP is favoured. In the case of a negative weight provided by the GridMAP service, gSched starts TCP handshake with the candidate node, obtains an RTT measurement and caches it, and then closes the connection.

An example is given in Figure 8.4, in which gSched concludes that *ngs.leeds.ac.uk* is the best site to schedule to. Based on this outcome, gSched stages the input files directly to the user space on that site and, as a final step, augments the template job description file with that chosen site as a requirement.

8.3.4 Evaluation Metrics

This subsection defines the metrics of the evaluation, which focuses on two main aspects: change in network connectivity, and change in scheduling performance.

The inspection of the utilisation of the network with and without using GridMAP is carried out in order to measure how much GridMAP assists in the management

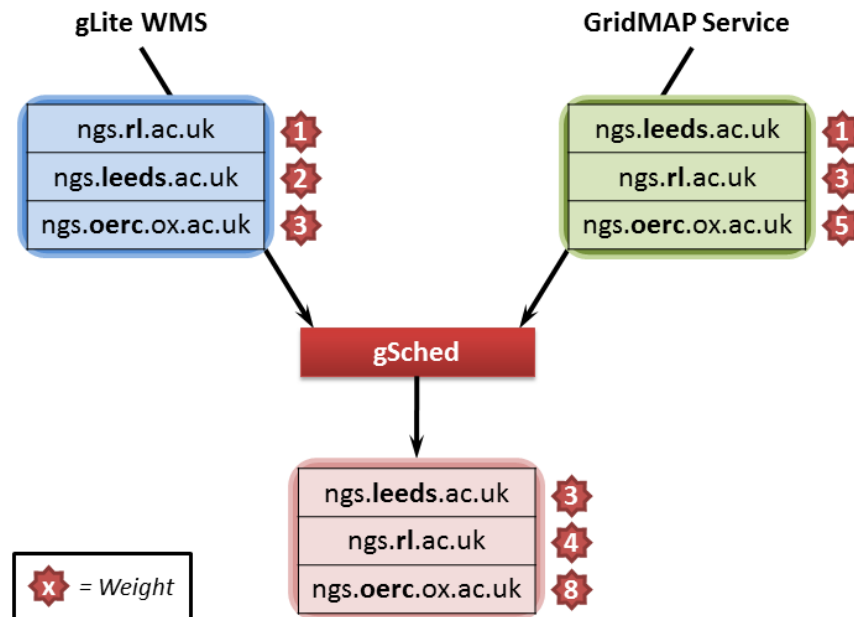


Figure 8.4: An Example of Weighting Compatible Resources Using gSched

of the network as a system resource. This shows whether or not GridMAP actually helps grid software in getting more out of the network. This is done by recording the levels of RTT, r , and achieved throughput, h . This is a fairly straight-forward process as Pasinemda supplies these metrics.

For the aspect of scheduling performance, however, some discussion is necessary to outline the phases that constitute the scheduling process, first from an abstract point of view then from that of gLite WMS. The mapping will assist in pragmatically calculating the scheduling efficiency.

Absolute Metrics

The breakdown of the different scheduling phases identified in subsection 1.1.3 is important to identify three key metrics. The first is the *staging time*, T_s , during which input files are transferred to the remote resource where execution should take place. Reducing this time is the main objective of network-aware scheduling.

Another important metric is what we call *intercession time*, T_i . This period of time is the one during which the job submitted by the user is handled by the scheduler in order to marry it to the best possible resources. This is the time consumed by the first three scheduling phases; i.e. matchmaking, submission, and queuing. Reducing this time indicates that this marriage takes place more quickly and hence has a higher scheduling turnover rate. For interactive jobs, this is highly desirable if not necessary.

For other less time-sensitive jobs, intercession time is relevant because from the user's point of view it signifies the delay before which their request, i.e. job submission, is actually addressed by the system. In any case, intercession time reflects the efficiency of the implemented scheduling mechanism.

The final metric commonly used to measure scheduling performance (especially in HPC scenarios) is job *completion time*, T_c , which is the total time taken by all five phases, i.e. the delay experienced between submitting a job for execution and obtaining its outcome. Completion time is considered a principal metric of scheduling efficiency as it signifies the success of the scheduler in allocating resources for the job that minimise the overall job lifetime.

gLite WMS Metrics

gLite WMS, just like any other scheduler, takes a submitted job through different phases^{III} during which the gLite system carries out a series of processes. However, these phases do not map directly to those of subsection 1.1.3. This is due not only to WMS's operational specifics but also to its disposition. WMS is a meta-scheduler designed to relieve users from interacting with remote LRMs of potentially different types. In order to do so, users submit their job descriptions and input data to WMS for it to act on their behalf. This establishes an additional matchmaking phase during which an extra data transfer step is carried out.

The time taken to stage output files can not be separated from execution time. This is because both processes are signified by only one change in WMS job status. When execution is complete either successfully or not, all output, error and log files specified in the job description are then retrieved to a location within trusted NGS nodes to avoid authorisation denial. Since it is not possible to monitor traffic going over NGS nodes, it is impossible to differentiate between execution and output staging times.

Absolute Metrics Redefined

It is now possible to redefine the absolute scheduling metrics as follows. Staging time, T_s , is the time taken by gSched to query gLite WMS followed by GridMAP about the best resources, to formulate the job description accordingly, and to stage the

^{III}More information about the WMS job cycle is available in appendix C.3.

input files. Reducing T_s indicates improved network management and is the main advantage that could be gained by grid schedulers adapting GridMAP.

The other scheduling metric that will be monitored in the evaluation experiments is the intercession time T_i . This is the time between the user submitting the job and it starting to execute. Recognising T_i allows us to put the change in T_s in context. It also serves as a measure of scheduling performance as it signifies the ability of the scheduler to match a job with resources that will tend to it as quickly as possible.

Job completion time T_c is typically used by HPC applications as a measure of scheduling performance. However, it is not considered as a metric of this evaluation due to the impossibility of distinguishing it using gLite WMS.

A summary of the principal evaluation metrics is presented in Table 8.1.

Aspect	Metric	Description
Network Utilisation	r	RTT
	h	Throughput
Scheduling Performance	T_s	Staging Time
	T_i	Intercession Time

Table 8.1: *Evaluation Metrics*

8.3.5 Methodology Summary

This section illustrated the experiment setup put in place to evaluate GridMAP's impact on grid scheduling. The criteria concerning the choice of the grid testbed were discussed first. It was decided that using a real grid infrastructure, NGS, would provide a realistic environment as opposed to using simulated grids. Testing on an operational production grid of such scale truly presents GridMAP with the most challenging conditions.

This was followed by a discussion of the choice of applications. Three real applications were chosen as case studies to represent the main classes of grid applications. Some background was given about the individual case studies and the array of grid applications they represent. Typical characteristics were also discussed, a summary of which is given in Table 8.2.

The section also presented gSched, a wrapper implemented to achieve network-aware scheduling on the NGS using GridMAP. Finally, the section described staging and intercession times as metrics that evaluate scheduling effectiveness and a means

Case Study	Sites	Input (MB)	Output (MB)	Inter-Process Messaging	Approx. CPU Usage (FLOP / Byte input)
AutoDock	3	2.83	0.08	None	43,000
Amber PMEMD	2	5.20	0.08	Very High	728,000
LogStore	5	110.66	0.00006	None	2,600

Table 8.2: *Summary of the Case Studies' Characteristics*

of computing them using the information obtainable from the NGS software stack. RTT and throughput will be used to evaluate network management.

8.4 Comparing the Performance of gLite WMS and gSched with GridMAP

This section highlights the outcome of experiments devised to test GridMAP in the context of each of the three case studies. In each experiment, 100 iterations of each case study is submitted to the grid using both setups described in subsection 8.3.3.

The experiments proved a significant improvement attained by using GridMAP. However, the results needed to be discarded as the comparison they portray is unfair. This is because a significant part of the improvement may be down to the alteration of the scheduling process caused by using gSched. In the first model which only uses gLite WMS, the UI-WMS node is used as a cache to store job input files before eventually being transferred to the chosen headnode, one step away from the execution theater (see Figure 8.2). This is a store and forward approach. On the other hand, gSched stages input files directly to the headnode of the matched site (see Figure 8.3). Hence, gLite WMS stages files on three separate steps while gSched achieves so in only two.

The results of these experiments are summarised in tables 8.3-8.5 without commentary, nonetheless, for the sake of completeness.

8.5 Analysis of Impact on Network Utilisation & Scheduling Performance

This section introduces and discusses the results obtained from three experiments that were designed in order to answer questions I, II and III of the evaluation questions (section 8.1). Each experiment involves submitting a set of 100 jobs of one of the

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gLite WMS	100/100	μ	6.05	2,068.89	2.46	116.96
		σ	0.06	436.23	0.29	41.88
		CV	0.01	0.21	0.12	0.36
gSched with GridMAP	100/100	μ	4.05	2,302.04	2.31	128.83
		σ	0.97	452.97	0.29	66.00
		CV	0.24	0.20	0.12	0.51

Table 8.3: Summary of the AutoDock Experiments Using gLite WMS and gSched

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gLite WMS	100/100	μ	6.16	2,257.52	3.52	140.82
		σ	0.52	580.52	.59	21.73
		CV	0.09	0.26	0.17	0.15
gSched with GridMAP	100/100	μ	3.80	3,726.89	3.07	137.97
		σ	0.65	1,099.48	0.47	23.04
		CV	0.17	0.30	0.15	0.17

Table 8.4: Summary of Amber PMEMD Experiments Using gLite WMS and gSched

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gLite WMS	100/100	μ	10.06	12,244.62	11.23	138.02
		σ	20.39	3,390.46	3.43	30.83
		CV	2.03	0.28	0.31	0.22
gSched with GridMAP	100/100	μ	4.05	10,632.91	12.68	134.62
		σ	0.48	2,299.77	2.46	28.21
		CV	0.12	0.22	0.19	0.21

Table 8.5: Summary of LogStore Experiments Using gLite WMS and gSched

case studies using gSched with and without GridMAP. Both of the techniques compared in this section use gSched and hence follow the same scheduling procedure. The only difference is the consult of GridMAP prior to final matchmaking. Therefore, the results presented here offer a fair portrayal of the gain obtained when GridMAP is used as a network advisor.

8.5.1 Experiment A: AutoDock

Figure 8.5 depicts the metrics recorded for 100 AutoDock jobs scheduled using gSched and gSched with GridMAP. Table 8.6 presents a statistical summary of the observed metrics. In order to appreciate the operational conditions of the NGS VO during the experiment, the system load was recorded to be between 45% and 51%, with a mean

value of 48.3% and a mode of 47%.

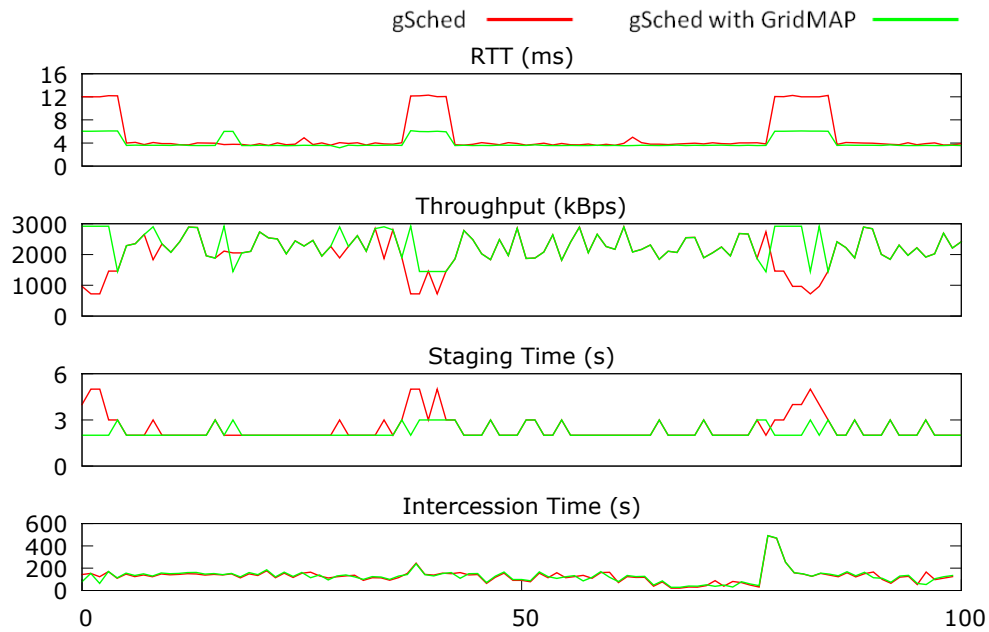


Figure 8.5: AutoDock Results With and Without GridMAP

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gSched	100/100	μ	5.28	2,047.29	2.58	129.21
		σ	3.11	561.19	0.74	65.32
		CV	0.59	0.27	0.29	0.51
gSched with GridMAP	100/100	μ	4.05	2,302.04	2.31	128.83
		σ	0.97	452.97	0.29	66.00
		CV	0.24	0.20	0.12	0.51
Improvement		$\Delta\mu$	23.3%	12.4%	10.5%	0.3%

Table 8.6: Summary of the AutoDock Experiment With and Without GridMAP

The experiment results demonstrate how the use of GridMAP helps improve management of the network. By choosing sites with better network connectivity, RTT was reduced by 23.3% and throughput increased by 12.4%.

The results also show the reduction in staging time that could be brought to such an application with the use of GridMAP. As presented in Table 8.6, this reduction is 10.5% eventually resulting in a 0.3% reduction in intercession time. These results might seem insignificant. However, this use case is of an HTC parameter sweep application that takes about 5 minutes to run each job. The significance of these results shall be further discussed in section 8.5.4.

8.5.2 Experiment B: Amber PMEMD

Metrics recorded running the PMEMD application are plotted in Figure 8.6 and summarised in Table 8.7. During the experiment, the NGS VO load was in the range 73%–83%, with mean of 79.6% and mode of 81%.

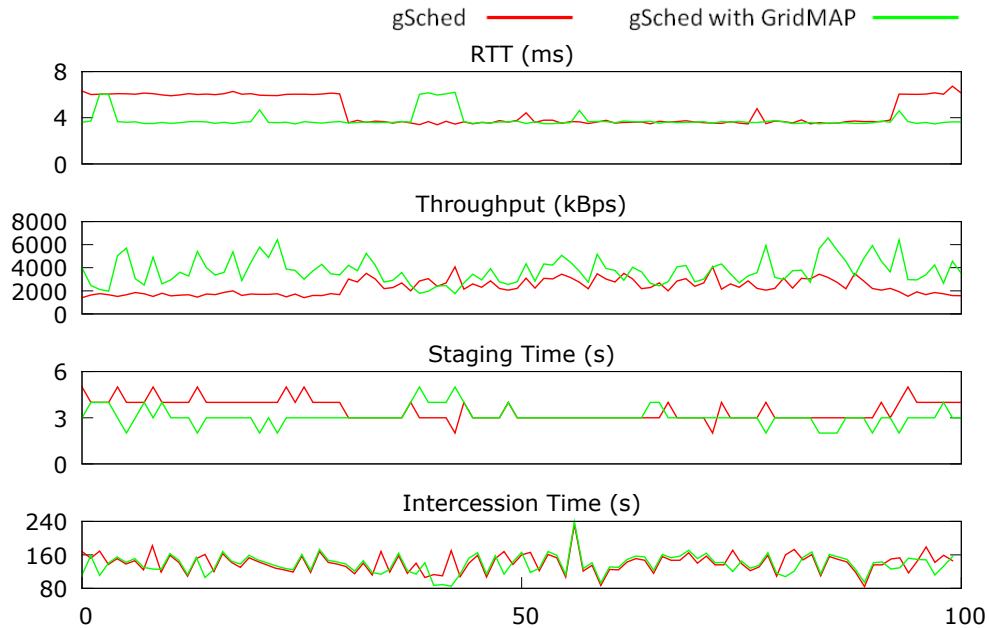


Figure 8.6: Amber PMEMD Results With and Without GridMAP

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gSched	100/100	μ	4.57	2,317.10	3.49	140.89
		σ	1.20	652.81	0.66	22.57
		CV	0.26	0.28	0.19	0.16
gSched with GridMAP	100/100	μ	3.80	3,726.89	3.07	137.97
		σ	0.65	1,099.48	0.47	23.04
		CV	0.17	0.30	0.15	0.17
Improvement		$\Delta\mu$	16.8%	60.8%	12.0%	2.1%

Table 8.7: Summary of Amber PMEMD Experiment With and Without GridMAP

Similar to experiment A, this experiment showed that improved network utilisation leads to reduced staging times, which consequently reflects on better scheduling performance. However, the improvement in throughput and staging time in this experiment is greater than that of the last. This is mainly due to the increase in the size of the input files in this experiment compared to those of the former. The larger the files are, the larger the congestion window is allowed to get and hence more utilisation of the available path, leading to reduced file transfer times.

8.5.3 Experiment C: LogStore

Plotted in Figure 8.7 are the metrics recorded for the LogStore application. A summary of these is presented in Table 8.8. During the experiment, the NGS VO load was between 77% and 88%, with a mean value of 84.4% and mode of 88%.

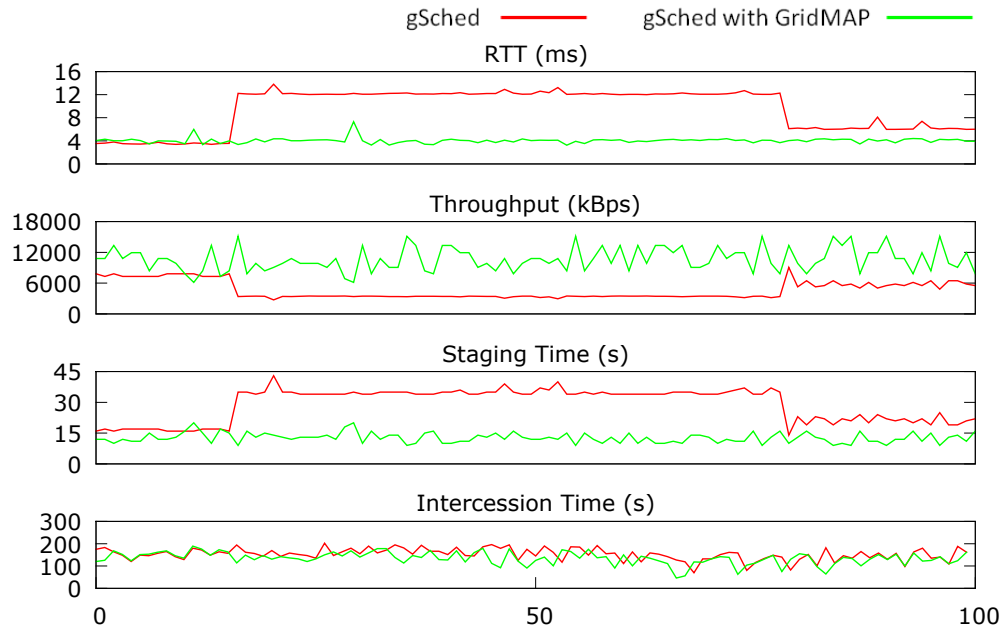


Figure 8.7: LogStore Results With and Without GridMAP

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gSched	100/100	μ	9.58	4,494.99	28.98	152.86
		σ	3.01	1,388.03	8.01	26.72
		CV	0.31	0.31	0.28	0.17
gSched with GridMAP	100/100	μ	4.05	10,632.91	12.68	134.62
		σ	0.48	2,299.77	2.46	28.21
		CV	0.12	0.22	0.19	0.21
Improvement		$\Delta\mu$	57.7%	136.6%	56.2%	11.9%

Table 8.8: Summary of LogStore Experiment With and Without GridMAP

The results of this experiment provide further proof that improved network-awareness results in better network utilisation which in turn yields in reduced staging times and improved scheduling performance. The significantly more pronounced improvement in network utilisation is again due to the even larger input files of this case study. There is, however, another reason. The improvement in experienced levels of RTT and throughput is primarily engendered by the variance between the network connectivity of the candidate sites. In this experiment, there were five candidate

sites of varying network qualities. Such variance served to accentuate the effect of network-aware scheduling.

Furthermore, GridMAP achieves more stable network utilisation. This is evident by comparing the CV values of RTT and throughput. This is a direct effect of the introduction of network-awareness, without which the network is taken for granted and not regarded as a shared system resource of limited capacity. This effect is significantly manifested in this experiment in particular due to the size of the input file which allows the application to pose more of a challenge to the over-provisioned network in comparison to experiments A and B.

8.5.4 Findings

The presented results provided two main findings.

Network Utilisation

The first finding is the conclusive demonstration that a better level of network utilisation is attainable by using GridMAP to introduce network-awareness into scheduling. This has been observed for all case studies representing different types of grid applications, including those where the used application was installed at two sites only.

Based on the outcome of the aforementioned experiments, it is evident that the magnitude of the rise in network utilisation is dependent on two factors. First is the size of the application input files. The larger the input the greater the effect of GridMAP in improving network utilisation [146]. The second factor is the degree of variance in the network connection quality of the sites posed as candidates for scheduling the job to. As this variance increases so does the potential impact of GridMAP on network utilisation. The variance in network quality is liable to be greater as the number of candidate sites increases. This issue of network utilisation and its relationship with the number of candidate sites and their network quality is discussed further in section 8.6.

Scheduling Performance

The second finding is the effect optimised network utilisation had on scheduling efficiency as represented by reductions in both staging and intercession times. This again has been shown across all case studies, albeit the reduction in intercession time

might seem meagre in experiments A and B. Nevertheless, the significance of such gain is only appreciated once what it presents is properly contextualised.

First, the input files used in our tests are relatively small in size: 2.83MB and 5.20MB for case studies A and B, respectively. The reduction in file staging times is proportional to the size of data to be transferred and hence will be more pronounced for jobs with larger input files.

Second, this improvement in scheduling is gained when coupling gSched with GridMAP. This is an added gain on top of the improvement in performance already achieved by staging directly to the execution theater headnode (c.f. section 8.4), the effect of employing a user-oriented application-level scheduler, i.e. gSched, instead of a meta-scheduler, i.e. gLite WMS.

Third, this gain is calculated per job based on the average of one hundred such jobs executed on a production grid infrastructure. The AutoDock suite is a parametric application, i.e. typical usage is running thousands of such docking jobs to subject a protein and its mutations to a very large number of compounds under varying conditions of temperature, pH, etc. For example, WISDOM is a grid deployment of virtual screening tools for malaria-related research [205]. The deployment submitted 31,231 AutoDock jobs consuming 50.5 CPU years over an effective period of 15 days; i.e. more than 2000 AutoDock jobs per day. A similar deployment submitted 77,504 jobs over 76 days [218]. Hence, even a gain of such magnitude is a welcomed advancement in the ongoing battle against drug resistance.

Similar parameter sweep applications use years worth of CPU utilisation to test candidate drugs [142] and to investigate membrane permeability that affects drug absorption [105]. Typical usage of the PMEMD application is very similar. This is also true for grid applications in other fields. For instance, [345] runs thousands of simulations using very large datasets (2GB–2TB) to model UK population demographics, further stressing the importance of rapid job turnaround time.

8.6 Analysis of Impact in Alternative Network Environments

This section presents a projection of the full impact of GridMAP on the NGS if the probabilities of choosing any of the available sites were equal.

8.6.1 Rationale

One of the things observed during the experiments presented thus far is the very rare selection of two of the NGS sites to run the submitted jobs. These were the *ngs.oerc.ox.ac.uk* site at OeRC in the University of Oxford and the *ngs.wmin.ac.uk* site at the University of Westminster. Of the NGS VO which includes five sites in total, these two sites represent roughly 21% of its aggregate computational capacity.

On inspection, it emerged that gLite WMS frequently rates these two sites very low. It was deduced that this is mainly due to low computational resource availability, which in turn is because some of the applications installed at these sites are not available on any other NGS sites. In the case of OeRC, these applications include a number of chemistry and data analysis applications, in addition to a collection of NGS service management modules. In the case of Westminster, this includes some bioinformatics applications and workload managers. The lack of availability of these applications elsewhere results in high demand for resources in the two sites.

Other reasons for low ranking are more site-specific. The OeRC site has undergone a gradual migration to a new site, *gk1.osc.ox.ac.uk*, and the installation of a new computing cluster. This process took a long time which coincided with the majority of the experiment runs. During this migration process, the capacity of the site was continuously being reduced as the original service at OeRC was curtailed whilst the new service was not introduced until very late in the evaluation process. In addition, the computational capacity made available at Westminster for the NGS VO is considerably smaller than that at other sites such as Leeds, Manchester, and RAL^{IV}. All of the above reasons contributed to frequent low ranking by gLite WMS.

When GridMAP was used, the probability of choosing one of these two sites decreased even further. Of the five sites that are part of the basic NGS VO, OeRC and Westminster have the worst connections from Lancaster University where gSched resides. This situated them even lower in the ranking created by gSched after consulting GridMAP.

The unlikely selection of these two sites presented a problem. It had the effect of virtually reducing the number of available sites. For Autodock this is reduced from 3 to 2, and for LogStore this is reduced from 5 to 3. This has a detrimental effect on the gain of using GridMAP as the two sites with the worst connections were hardly ever selected for reasons beyond their network connectivity. Moreover, this

^{IV}Rutherford Appleton Laboratory in Oxfordshire.

has considerably confined the variance in network quality as the remaining sites with a somewhat good chance of being chosen by any scheduler are of quite proximate network metrics.

Therefore, this section attempts to apprehend the full impact of GridMAP on scheduling over the NGS. This is achieved by comparing the results already presented in section 8.5 against gSchedInv with GridMAP. *gSchedInv* is a variant of gSched that inverses the weights indicated by GridMAP. This promotes the use of sites with bad connectivity. This process is illustrated with an example in Figure 8.8.

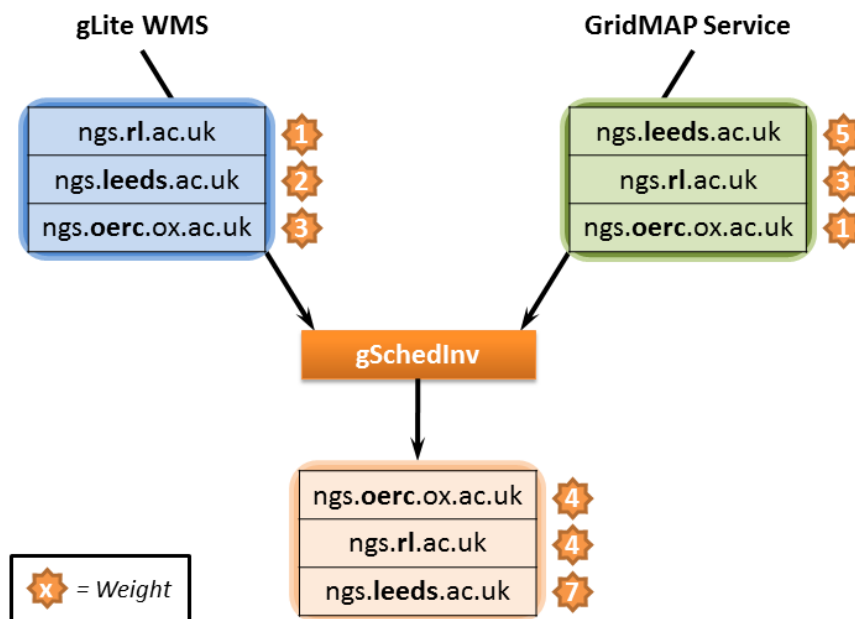


Figure 8.8: An Example of Weighting Compatible Resources Using gSchedInv

The results of this exercise should demonstrate the effect of scheduling to the sites with the worst network connectivity. In other words, the results will compare the worst case of network-oblivious scheduling, represented by gSchedInv with GridMAP, against network-aware scheduling, represented by gSched with GridMAP.

8.6.2 Results

Tables 8.9-8.11 summarise the outcome of the experiments using gSchedInv.

Three main observations stand out from these results. First is the amplification of the gain of using GridMAP. This is expected as the aim of the experiment is to compare the best scheduling selection against the worst from a network perspective. The outcome reveals the magnitude of the effect of GridMAP-induced network-awareness on each application.

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gSchedInv with GridMAP	100/100	μ	11.19	1,285.78	3.70	144.05
		σ	2.33	633.53	1.02	56.98
		CV	0.21	0.49	0.27	0.40
gSched with GridMAP	100/100	μ	4.05	2,302.04	2.31	128.83
		σ	0.97	452.97	0.29	66.00
		CV	0.24	0.20	0.12	0.51
Improvement		$\Delta\mu$	63.8%	79.0%	37.6%	10.6%

Table 8.9: Summary of the AutoDock Experiment Using gSchedInv and gSched

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gSchedInv with GridMAP	100/100	μ	6.21	1,991.87	4.23	141.57
		σ	0.34	225.54	0.30	34.42
		CV	0.06	0.11	0.07	0.24
gSched with GridMAP	100/100	μ	3.80	3,726.89	3.07	137.97
		σ	0.65	1,099.48	0.47	23.04
		CV	0.17	0.30	0.15	0.17
Improvement		$\Delta\mu$	38.8%	87.1%	27.4%	2.5%

Table 8.10: Summary of Amber PMEMD Experiment Using gSchedInv and gSched

Scheduling Technique	Number of Jobs (Successful/Total)		r (ms)	h (kBps)	T_s (s)	T_i (s)
gSchedInv with GridMAP	100/100	μ	12.37	3,475.73	34.37	159.62
		σ	0.68	299.85	2.75	24.36
		CV	0.05	0.09	0.08	0.15
gSched with GridMAP	100/100	μ	4.05	10,632.91	12.68	134.62
		σ	0.48	2,299.77	2.46	28.21
		CV	0.12	0.22	0.19	0.21
Improvement		$\Delta\mu$	67.3%	205.9%	63.1%	15.7%

Table 8.11: Summary of LogStore Experiment Using gSchedInv and gSched

The second significance of the results is the display of the effect of the variance in the number of candidate sites on the GridMAP gain. Take for example experiments A and B where the number of candidate sites were 3 and 2, respectively. The gap between gSchedInv and gSched in experiment A, presented in Table 8.9, is far wider than that presented in Table 8.6. In contrast, the gap between the results in Tables 8.10 and 8.7 is far less severe. This is because the difference between the connectivity of the two sites in experiment B is significantly narrower than the corresponding difference in experiment A.

The final observation is the low CV values in the case of gSchedInv with GridMAP.

This is because the worst site was usually only one in each experiment: OeRC in experiment A , RAL in B, and Westminster in C.

8.7 GridMAP's Overheads

This section examines the different overheads of the GridMAP solution. These overheads make up the expense of using GridMAP, and affect its performance, scalability and applicability.

Based on the the deployment detailed in section 8.2 and summarised in Figure 8.9, this section presents the associated storage, computational and network overheads. The section is concluded with a discussion of the findings.

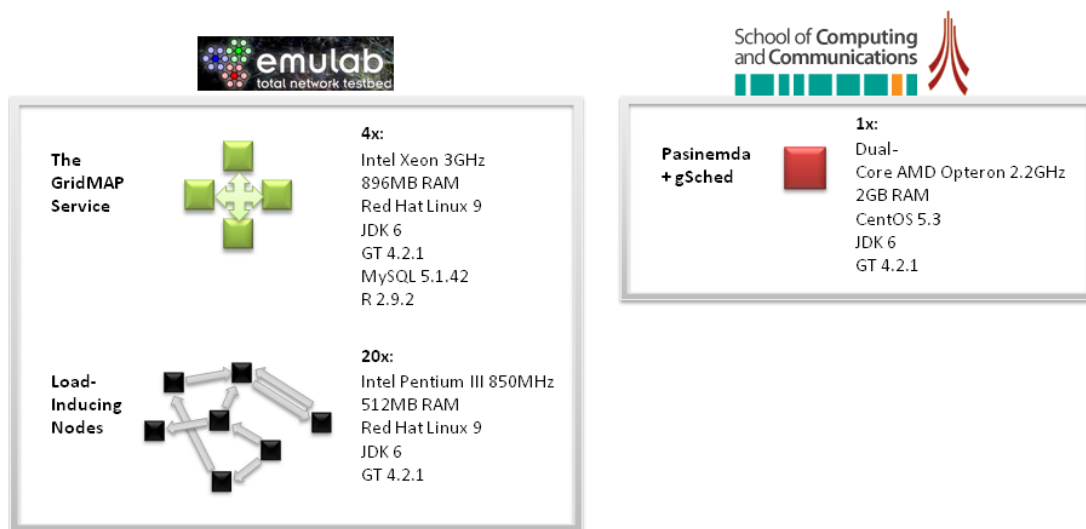


Figure 8.9: Summary of the GridMAP Deployment Configuration

8.7.1 Storage Overhead

Storage overhead refers to the amount of disk space required by the GridMAP solution during its operation. This cost is made up of two elements: the storage overhead of Pasinemda and that of the GridMAP service.

Pasinemda

The storage cost of Pasinemda, S_P , is the maximum amount of space required by the daemon at any one time. The value of S_P depends on the number of measurements kept in cache, which in turn is directly proportional to two factors. The first is the configured submission period, s . The second is the number of measurements

collected during this period. Hence, S_P is inversely proportional to v , the average period between flows generated by the application. Given that each measurement takes 90B to be cached, Pasinemda's storage cost in kilobytes for a certain node N can be calculated using:

$$S_P^N = \frac{90s^N}{1024v^N} \quad (8.1)$$

where s^N and v^N are the submission and average inter-flow period for node N , respectively. Equation 8.1 can be used to estimate the storage overhead of Pasinemda. It can also be used to balance the trade-off between reporting frequency and storage requirements, which can be useful for nodes with restricted network or storage resources.

Using the equation, a highly network-intensive application generating an average of 100 flows every second would require just under 264kB of disk space for a Pasinemda cache which expires every 30 seconds, or 7.72MB for a cache expiring every 15 minutes. Such storage needs are indeed quite limited even for applications with frequent network activity.

During the experiments, mean S_P for the node scheduling onto the NGS from InfoLab21 was recorded as 2.48kB with a standard deviation of 1.16. For the 20 load-generating virtual machines, mean S_P was 58.86kB with a standard deviation of 45.71.

GridMAP Service

The GridMAP service's storage overhead, S_S , corresponds to the total amount of space that is required by the different user processes that constitute the service. These processes handle the database back-end, service front-end, and data analysis elements. The value of S_S , thus, relies on a number of different variables.

First, it depends on the size of the database which in turn depends on variables such as the number of hosts from which measurements have been received, the number of distinct IP addresses they possess, and the number of measurements reported by each host. Second, S_S depends on the amount of storage space required during data analysis. The storage requirements of the implemented Akima splines and exponential smoothing algorithms are usually null, except at time of high load where additional virtual memory might be required. Third, S_S depends on the degree of distribution and data replication exercised in the system. In the used deployment, the service was distributed over a span of 4 computers across which data is dynamically replicated to ensure high availability and fault-tolerance.

The size of an empty database is 33.4KB per replica. The total operative duration of the experiments was about 16 hours, at the end of which the database held over 8,900 measurements. This cost 4.87MB of disk storage per replica, 19.48MB in total.

8.7.2 Computational Overhead

Computational overhead corresponds to the amount of system resources utilised by the GridMAP solution during its operation. This section focuses on the utilisation of two system resources of Pasinemda and the GridMAP service, namely CPU and dynamic RAM. In this section, CPU utilisation is presented as a percentage of the total capacity mentioned in subsections 8.2.1 and 8.13, and summarised in Figure 8.9.

Pasinemda

Table 8.12 presents the average computational overheads observed for Pasinemda across all nodes, where CPU utilisation is presented as a percentage of the total capacity and runtime memory utilisation is presented in kilobytes. It is evident that Pasinemda's processing demands are low.

Utilisation	Cost	
CPU (%)	μ	0.0
	σ	0.1
Memory (kB)	μ	742
	σ	13

Table 8.12: Computational Costs of Pasinemda

GridMAP Service

During the GridMAP service's uptime, the amount of CPU cycles and RAM consumed by the service and its related processes (such as MySQL) on the 4 hosts were recorded every other minute. The average across all 4 machines is presented in Table 8.13, where CPU utilisation is the percentage of total processing capacity and runtime memory utilisation is in megabytes. In the table, *Idle* refers to the average utilisation observed before the GridMAP service is loaded, while *Experiments Running* refers to the average utilisation during the experiments with the GridMAP service loaded.

It is apparent that the computational requirements of the GridMAP service rise considerably as measurement reports and queries are received. Nevertheless, these

Utilisation	Cost		
		Idle	Experiments Running
CPU (%)	μ	0.4	2.3
	σ	0.3	29.1
Memory (MB)	μ	156.3	193.9
	σ	19.2	58.0

Table 8.13: *Computational Costs of the GridMAP Service*

demands are hardly substantial even considering the moderate capability of the computers used to host the GridMAP service.

8.7.3 Network Overhead

Network overhead is the amount of network resources that are used by the GridMAP system to carry out its operations. This can be represented by the volume of generated network traffic.

Pasinemda

Pasinemda contacts the service to report collected measurements. These reports are simply what the daemon has cached during the preceding period. Hence, Pasinemda's network overhead, N_P , is equivalent to its storage overhead plus a constant, i.e.

$$N_P \equiv S_P + w$$

where w is a constant that resembles the identification information sent by Pasinemda at spawn time to register its name and IP addresses. For a node with two public IP addresses, the value of w is around 25–30B (depending on the length of the hostname).

Excluding spawning traffic, the amount of network traffic generated by node N per minute in kB could thus be calculated using the following equation, derived from Equation 8.1.

$$N_P^N = \frac{90}{1024v^N} \quad (8.2)$$

The Pasinemda instance on the scheduling node was observed to create 0.16kB per minute of network traffic, while the instances on the 20 load-generating nodes were found to create an average of 1.78kB per minute.

GridMAP Service

The GridMAP service's network overhead is its maintenance cost, created by the service's internal operations as a distributed system. It includes traffic generated to administer data management, load balancing, data replication and other processes. This was measured at an average of 7.64kB per minute with a standard deviation of 33.26.

8.7.4 Findings

This section has highlighted the different costs associated with operating the GridMAP system. These costs were obtained from monitoring the GridMAP deployment described in section 8.2, where the GridMAP service was deployed on a set of 4 machines with moderate computational capabilities. This was done in order to ensure an accurate representation of the different overheads that can be used to ascertain feasibility.

First, the section demonstrated the remarkably low storage and computational overheads of Pasinemda. It follows that the daemon is not only unobtrusive on the network but also on the grid host where measurement occurs. This adds to the advantages of Pasinemda and makes it applicable even to hosts with limited resources.

Second, the study quantified the storage, computational, and network costs associated with the GridMAP service. These costs have been found to be humble. The simple deployment of GridMAP presented here has been proven to be more than enough to cater for the needs of more than 20 nodes that collectively generate a significant and sustained load. Based on the measured overheads, this deployment of the service could withstand greater loads.

8.8 Outcomes

The aims of this evaluation of the GridMAP solution were summarised in section 8.1 using 4 questions. It is now appropriate to revisit them in light of the outcomes of this evaluation.

- I. *Can network utilisation be enhanced by using GridMAP?, and*
- II. *Can grid scheduling be improved by using GridMAP?*

The answer to both questions is ‘yes’, as demonstrated through a series of experiments on the NGS. The results indicate that RTT was reduced by a factor between 16.8% and 57.7%, and throughput was increased by 12.4% or as much as 136.6%. Staging time was improved by a factor ranging from 10.5% to 56.2%, while intercession time was cut by a factor between 0.3% and 11.9%.

Improvement in network utilisation has been found to depend on two factors, namely the size of the input data and the variance in connection quality. The improvement in network utilisation is directly related to increasing either of these factors. The gain in scheduling performance follows on from the improvement in managing the network by being aware of its state. However, it is more subjective as it relies on the characteristics of the grid application and of the specific job.

The experiments were then modified to investigate how the results would change if more nodes were effectively available as candidates. This exercise showed that GridMAP is able to cause even further improvement in network utilisation (up to 67.3% less RTT and 205.9% more throughput) and scheduling performance (upto 63.1% less staging time and 15.7% less intercession time) in such situations.

III. *Does GridMAP affect different grid applications differently?*

Experimenting using different grid applications has demonstrated that indeed the effect of GridMAP is different for each. This is partly due to the case study’s characteristics, such as input data size, which reflect its network requirements. It was evident from the results that data grid applications stand to gain the most from GridMAP. This is intuitive as data grid applications are the most network-dependent and hence gain more from better network management.

However, the impact also relies on the application’s computational requirements. The HTC case study was observed to benefit more from network-aware scheduling than its HPC counterpart. This is because the HTC approach is set out to maximise long-term scheduling turnover rate, which aligns well with GridMAP’s aim to reduce job staging and intercession times through favouring remote resources reachable via connections of low delay and high throughput. On the other hand, scheduling HPC applications is more concerned with computational considerations, such as the number of free CPUs.

IV. *What is the cost of using GridMAP?*

Different costs of GridMAP have been studied, establishing the unobtrusive nature of Pasinemda as an information collection agent. This adds to the applicability of Pasinemda, allowing it to be deployed to collect measurements in various situations, including those with limited resources such as mobile nodes. Moreover, the overhead study concluded that the GridMAP service is easily deployable with fairly limited disk, CPU, memory, and network footprints. This is even more valid when considering the added value the service offers to schedulers.

8.9 Summary

This chapter presented the instrumentation and results of a series of experiments to investigate the overall impact of the GridMAP system. First, the experiments scheduled three different grid applications with and without GridMAP on the NGS, a production grid infrastructure with an advanced software stack and overprovisioned network. The results attest to the gains that could be achieved both in terms of network utilisation (as indicated by RTT and achieved throughput) and scheduling performance (as signified by staging and intercession times). Such finding is key in proving the importance of considering the network as a grid system resource. It also testifies to the capabilities of GridMAP as a solution that enables grid schedulers to achieve network-awareness. The chapter also included an analytical and empirical study of the different GridMAP overheads. This study illustrated the low impact of both Pasinemda and the GridMAP service on their hosts and on the network. This demonstrates the feasibility and deployability of the GridMAP system.


Part V

Conclusion

Knowledge is the conformity of the object and the intellect.

IBN RUSHD

Conclusion

 disparity has been identified by the thesis between the network requirements of grid applications and grid technologies. To address this, the thesis put forward an argument for network-aware scheduling. It investigated current solutions, formed a list of requirements deemed necessary to actualise efficient network-aware scheduling, and proposed and evaluated a mechanism that satisfies these requirements. This chapter concludes the thesis by looking back at the contributions of each individual chapter and those of the overall thesis. This chapter also presents directions of future work and general comments on the future of grid computing.

9.1 Thesis Summary

Chapter 1 headed the thesis by providing acquaintance with grid computing in general and the topics of resource management and scheduling in particular, briefly critiquing current GISs, stating the research goals of the thesis and introducing the thesis structure.

Chapter 2 explored the field of network measurement. It defined different network metrics, described alternative measurement approaches, and portrayed a portion of the great number of measurement techniques that have been proposed in the literature. Of these techniques, TCP messaging was recognised as one of minimal overhead, easy deployment and reasonable accuracy. However, previous works revealed that the use of artificial TCP probes should be restricted as much as possible as it diminishes the reliability of the technique and could cause security concerns.

Chapter 3 started by highlighting the evolution of resource management in early distributed systems such as parallel and cluster computers. This illustrated the difference between the challenges faced by such systems and those faced by grids, giving impetus for network-aware grid scheduling. The chapter also surveyed a wide array of GISs and touched upon the different shortcomings perceived with them. These were mainly concerned with accuracy, measurement overhead, maintenance overhead, dependability, and the architecture that dictates information collection and dissemination.

Chapter 4 discussed why network-awareness is a pragmatic solution to the inappropriately addressed problem of network unpredictability in grids. It also presented a number of reasons why current GISs fail to facilitate effective network-aware scheduling. In order to accomplish this, the chapter discussed different classes of grid applications, their characteristics and network requirements. The chapter concluded by stipulating 9 requirements for an efficient, low-overhead GIS that avoids the pitfalls of existing GISs.

Chapter 5 introduced the twofold architecture of the GridMAP system. Measurements are collected by the Pasinemda daemon using a purely passive, low-overhead TCP-ping technique. This avoids the use of artificial probes and pre-defined node lists, two factors that have restrained previous TCP-ping techniques. The other part of the system is the GridMAP service. It is a grid service that consolidates and analyses Pasinemda's measurements in order to provide grid schedulers with network performance predictions. The service is intrinsically decentralised but retains a logically centralised model. This enables scalability, resilience, and high availability whilst also offering a single point-of-contact which greatly simplifies information collection and dissemination processes.

Chapter 6 presented the implementation of the five functions of the GridMAP system: Performance Measurement (measurement of RTT and achieved throughput), Data Accretion (periodic submission of metrics to the GridMAP service using NM-WG XML schema), Data Management (the GridMAP service's database back end), Information Publishing (service invocation by schedulers, data analysis to produce predictions, and dissemination of this information), and Information Consumption (three different ways of integrating GridMAP and schedulers).

Chapter 7 ascertained the accuracy of Pasinemda's measurement technique by testing it against two well known and widely used measurement tools over five different connections. It was demonstrated that Pasinemda produces accurate network

performance measurements in a consistent and reliable manner except in the case of networks with extremely short RTTs, e.g. Ethernet. Nonetheless, the magnitude of error in such circumstances was shown to be relatively very small and does not skew overall network performance assessment.

Chapter 8 examined the overall impact of using GridMAP as a solution that enables network-aware grid scheduling. In this chapter, a number of experiments were carried out using gLite WMS to schedule three different case studies over the NGS. To this end, a wrapper was implemented to enable gLite WMS to consult GridMAP and consume the information it returns. There were a number of key findings from the experiments. First, they illustrated that better network utilisation, as reflected by lower RTT and higher throughput values, is achieved using GridMAP. Second, scheduling performance was shown to be affected by better network management, by observing job staging and intercession times, with a different degree of benefit for each case study. However, even applications of relatively low network reliance benefited from GridMAP as in some cases small improvements could have a significant impact due to the application scenario. Some examples of such situations were provided in the chapter. Finally, the chapter investigated the overheads of Pasinemda and the GridMAP service, and concluded that the overheads are small in respect to the potential gain.

9.2 Revisiting the Requirements for Network-Aware Scheduling

Based on an examination of grid application characteristics and contemporary grid technologies, section 4.4 put forward 9 recommendations for solutions that enable network-aware grid scheduling. These were indirectly addressed in section 5.4 after introducing the design of the GridMAP system. These requirements are now revisited, with references to how they were further addressed throughout the thesis.

- I. *Accurate & End-to-End*
- II. *Non-Intrusive*
- III. *Reliable*

These three requirements directly relate to the network measurement technique. Pasinemda is a purely passive network measurement tool used by GridMAP. It was introduced in sections 5.2 and 6.1. Chapter 7 specifically evaluated and discussed Pasinemda's accuracy and reliability.

IV. *Low Operation Overhead*

Requirement IV ensures that the information gathered about the network is utilised by grid schedulers in a seamless fashion without explicit user intervention in order to improve performance, increase usability and reduce errors. This aspect depends on how the consumption of GridMAP information is integrated with the scheduler. Section 6.5 examined three different approaches to attaining such integration and subsection 8.3.3 detailed how we chose to implement it in the context of the NGS.

V. *Low Deployment Overhead*

This requirement aims at minimising deployment overhead and increasing applicability even in grids where strict administrative policies are in place. GridMAP possesses such property due to its independent measurement technique that requires no peer coordination (discussed in sections 5.2 and 6.1) and its low computational footprints (discussed in section 8.7).

VI. *Decentralised*

VII. *Single Point-of-Contact*

Requirements VI and VII attempt to avoid the disadvantages associated with centralised systems while simultaneously gaining their advantages. Both these aspects are features of GridMAP as described in sections 5.3 and 6.2–6.4.

VIII. *Selective*

Requirement VIII is prescribed to reduce operational overhead and promote scalability. This is reflected in Pasinemda's measurement technique (discussed in sections 5.2 and 6.1) and GridMAP's overheads (evaluated in section 8.7).

IX. *Interoperable*

Requirement IX is stipulated to increase compatibility with other grid technologies. This was addressed in GridMAP's design and implementation, and specifically described in subsections 5.3.3 and 6.4.1.

9.3 Contributions & Implications

The introduction of the GridMAP system expands the existing body of grid network measurement to new boundaries with its unique passive measurement approach and

decentralised service-oriented architecture. The main contributions of the thesis are reviewed in the following, first by revisiting the research goals set in section 1.4 and then by commenting on the implications of the work.

9.3.1 Revisiting the Research Goals

- I. *There is room to improve grid scheduling by incorporating network performance into the resource allocation process.*

The arguments and experiments put forward in this thesis emphasise the imperativeness of network-awareness, i.e. acknowledging the network as a system resource that needs monitoring and optimised usage. This was demonstrated first by surveying the grid technologies state of the art, and arguing that indeed the network is not appropriately addressed as a resource of unpredictable nature. The thesis then presented GridMAP as a GIS that enables schedulers to obtain predictions of network performance, and demonstrated that scheduling efficiency is improved by using GridMAP in conjunction with an advanced scheduler over a production grid infrastructure.

- II. *Network performance information could and should be accurately and reliably collected using passive techniques.*

By surveying different network measurement approaches and discussing their accuracy, overhead and deployability shortcomings, the thesis identified passive measurement as the best means of collecting information about the network. Furthermore, through the introduction of Pasinemda, it was demonstrated that passive measurement is indeed an effective and reliable means of accurately measuring networks in grid environments.

Pasinemda is a passive measurement daemon that extracts RTT and achieved throughput from observed TCP traffic. Thus, it requires no peer collaboration, is of no added network overhead, avoids the problems of ICMP-based measurement, and requires no user intervention and is hence always on. It has also been proven to require low computational resources.

Pasinemda is thus a reliable stand-alone tool for situations where frequent TCP communication is prevalent. This is of particular relevance to grids where TCP is heavily relied upon, and because Pasinemda is easy to deploy in infrastructures with stringent administrative policies.

- III. *Network performance information needs to be made available in a manner that avoids the shortcomings of existing GISs.*

The thesis examined various GISs and discussed different drawbacks observed in them. By drawing on this understanding, the thesis recommended a number of requirements for future GISs.

The thesis also proposed GridMAP as a GIS that takes these requirements into account and, hence, avoids problems witnessed with other GISs. GridMAP is a distributed solution that uses end-to-end network measurements obtained passively in order to minimise overhead and simplify deployment without compromising accuracy. Despite being distributed, GridMAP retains a logically centralised structure which has two significant advantages; it enables analysis of the collected data to produce performance predictions, and provides a single point of contact for collecting and disseminating information.

9.3.2 Implications

The thesis has demonstrated that GridMAP is an effective way of providing network state to schedulers, enabling them to adapt their resource allocation decisions to changes in the underlying network. The benefits of this have been demonstrated on an overprovisioned network infrastructure.

However, the implications of this work reach beyond the design, implementation and evaluation of a grid-specific passive network measurement system. The thesis highlights an aspect of grid computing that is not properly addressed by current systems. On inspection, one finds that the problem exists in both directions of the ‘information food system’: schedulers are not designed to take network status into consideration; and there are no proper means to provide such information. More work is needed to push network awareness to be a requirement of grid technologies and standards (as previously discussed at the end of section 1.2) from schedulers, such as gLite WMS and Condor-G, to schemas, such as GLUE.

Furthermore, the GridMAP system could in theory be used with other distributed systems that rely on frequent TCP transfers. This includes scenarios where a decision is sought for selecting between peers in a peer-to-peer swarm (c.f. [175]), edges in a content delivery network (c.f. [173]), content providers in content-centric networks (c.f. [347]), or data centres in cloud communications (c.f. [173]). As the uptake of such

distributed paradigms grows, the contention for network resources increases; and so does the applicability of GridMAP.

Finally, GridMAP is designed and presented from an application's perspective, rather than that of an administrator or a grid user. It is thus an operational solution proposed to improve the performance of grids, and not a diagnostic or user-centric one. Nevertheless, GridMAP's interoperable and extensible design allows it to be employed to serve such other purposes if required. For instance, GridMAP is able to provide subjective per-application measurements that could be used to detect network abnormalities (c.f. [252]) or to create graphical utilisation and performance monitors (c.f. [294]).

9.4 Future Work

Working on this thesis has opened up a number of avenues for future work.

9.4.1 Horizontal Expansion

There is great potential for horizontal development of the GridMAP service.

Measurement Analysis

The information offered by GridMAP could be improved by horizontal development of the service. Such efforts would entail the implementation of further numerical analysis procedures in order to provide more advanced prediction of network performance.

For instance, the interpolation method used to resample the measurement series could be unsuitable for series of very high fluctuation; It has been documented in [301] that some forms of interpolation create large margins of error through bias towards low-frequency components. Although the Akima spline method currently employed by GridMAP is less prone than others to outliers, it is still vulnerable to severe fluctuations.

Advanced numerical analysis could also enable more accurate identification of trend and irregular components. The Holt-Winters exponential model [189, 358], for example, allows for seasonal components. This aids in recognising daily, weekly, monthly and even seasonal patterns if they exist in the sample, the determination of which could significantly improve prediction accuracy. Pattern identification is of special significance in production infrastructures and other networks with narrow

differences between RTTs of candidate nodes [149]. Additionally, identifying such patterns would be beneficial for dynamic anomaly detection (c.f. [91]) and generic troubleshooting purposes.

Finally, a number of different numerical analysis methods could be employed and their accuracy evaluated (through a metric such as MSE or Thiel's inequality coefficient [332]) similar to that implemented in the NWS. Such range of analysis methods would increase GridMAP's ability to adapt to different measurement series and reduce the overall prediction error margin.

Registry

Another possible extension to the system is the inclusion of a Universal Description, Discovery and Integration (UDDI) registry. This registry, implemented as a Web service, would be used to keep track of the address of the GridMAP service front, allowing the latter to change location without affecting the operation of the GridMAP system. Spawning Pasinemda instances would contact the UDDI registry through its fixed URI to retrieve the location of the GridMAP service front. If for any reason the GridMAP service relocates, it would update its address in the UDDI registry and Pasinemda would query the UDDI registry again.

Moreover, the UDDI registry facilitates graceful change of the GridMAP service front which allows operation to seamlessly continue in the case of GridMAP service front failure.

Beside the advantages of geographically liberating the GridMAP service, the location transparency brought by a UDDI registry enables a multi-front GridMAP service. This could be used to add another level of load-balancing where a collection of grid nodes would get assigned a different front.

Database

Another potential improvement is the use of an XML-based database back end instead of an SQL-based one. Such change is worth investigating in order to ascertain whether reducing the amount of conversions between XML and SQL would improve system performance.

9.4.2 Vertical Expansion

Another direction of development is vertically, where there is also plenty of room. Currently, Pasinemda measures RTT and throughput, which are essential network metrics for the majority of grid applications. However, this formula might not be enough for some applications. Collecting a wider range of network metrics (such as jitter, loss, etc.) would provide more utility.

Furthermore, as already mentioned in section 6.3, Pasinemda is capable of providing information about the availability of computational resources, such as CPU cycles and memory space, along with network metrics. This has indeed been implemented in the early stages of this work [137] but not sufficiently developed as more focus was given to the development of network-aware scheduling. One argument against such development is that the grid already has enough GISs that cater for the management of computational resources. However, these systems suffer from the architectural drawbacks highlighted in section 4.3 that adversely affect information dissemination and management.

9.4.3 Autonomy

Another possible direction for development is the implementation of the GridMAP system in a dispersed form where a grid node manages the data it collects about the network without needing to refer to a grid service. Such a solution would need to be as simple as possible in order to keep computational overheads to a minimum, but it would offer complete autonomy and even greater ease of deployment. Certain grids would gain from such a devise, for example collaborative computing and mobile grids.

9.5 Future Trends

While carrying out this research, there has been a recurring theme of uncovering a large array of quite similar solutions to the same problem. Such wealth in number of grid technologies is a result of loosely defined borders as demonstrated by various grid definitions and unbinding standards. Such an aspect could be interpreted as both a liberating strength and a fatal weakness. On the one hand, it is a testament to the huge size of potential beneficiaries that includes numerous fields of scientific research in addition to some commercial applications. On the other hand, it displays

an almost haphazard application of solutions. Despite standardisation efforts, these solutions have large overlap and are non-interoperable in many cases. Ironically, such conflicting landscape completely goes against the ethos of grid computing.

In the end, grid computing provided many advancements in distributed computing that enables collaborative projects on a massive scale. However, the grid also failed to address a number of issues. Some of these shortcomings led to the rapid rise of cloud computing [348] with the perception of it being a replacement, competing technology rather than a subsequent one. This issue of grids versus clouds has recently been well debated (c.f. [160]); Will grids continue to be used, or does the introduction of cloud computing negate the need for grids?

Undoubtedly, cloud computing has managed to resolve many of the problems of its predecessor, the grid. The most important of these is the simple, scalable economic model offered by cloud computing, a factor that should not be underestimated. Many organisations would prefer moving their computations to the cloud rather than building, configuring and running their own computing infrastructure which entails covering upfront hardware costs (servers, network, storage, firewalls, cooling, etc.) and running costs (electricity, part replacement, physical security, and, perhaps more importantly, human support). The former option, i.e. cloud computing, is not just more cost-effective than the latter, but also has less associated risks and offers great flexibility to match the organisation's change in computing demands. Such a clear-cut on-demand economic model was lacking from the grid.

Cloud computing has recently received a lot of interest, especially from mid-sized businesses [297]. There are also signs of rising demands within research communities for cloud computing capabilities. For instance, the NGS has started providing such services [1], as have several UK higher education institutions (c.f. [31]). Despite this, rightly or wrongly, there are still many security and privacy concerns about programs and information associated with cloud computing [254, 69].

There are three reasons why grids are likely to remain on the scene for the foreseeable future. First, the e-science community has invested large sums of money into grid infrastructures. Hundreds of grid-based research projects have been made possible by these technologies, and thousands of researchers and scientists still rely on it. For example, the European Commission invested around €130m in grid research between 2002 and 2006, resulting in 42 national grid initiatives. The €100m European Grid Infrastructure (EGI) project [12] commenced with an aim of assembling a meta-infrastructure to provide interoperation between all these independent grids. Second,

while the cloud is designed to offer flexibility and usability, the grid is designed to facilitate resource sharing. This still makes the grid more suitable for collaborative projects, providing further anchor for the grid in the e-science community. Third, the grid still provides a yet unmatched model for collaborative computing.

In light of this, it is important to promote the role that GridMAP will play both in today's grid and cloud environments. The thesis has already elaborated on GridMAP's role in grids. In fact, the evaluation part of the thesis established the level of improvement that can be brought by GridMAP even in a high-end overprovisioned grid infrastructure. For clouds, GridMAP's role is not yet entirely clear and would only become clear after carrying out a corresponding requirements analysis. However, based on our understanding of cloud application scenarios, there seems to be a role for GridMAP to play in optimising inter-data centre communications. For example, some cloud users employ more than one cloud infrastructure (from one or more cloud service providers) at a time and choose between them dynamically according to job requirements and cost (c.f. the inter-cloud API [184] developed at the Belfast e-Science Centre). This is essentially comparable to the 'islands of grids' model (introduced in subsection 8.3.1) already adopted by many production grids. Indeed, there is already interest in optimising the network for cloud computing (c.f. [69, 76]), something that GridMAP is designed to enable.

Bibliography

- [1] Accessing the NGS Cloud Service. <http://www.ngs.ac.uk/accessing-the-ngs-cloud-service>. [cited on p. 163]
- [2] APAN: Asia-Pacific Advanced Network. <http://www.apan.net/>. [cited on p. 50]
- [3] ARGUS- Auditing Network Activity. <http://www.qosient.com/argus/>. [cited on p. 27]
- [4] ATLAS - A Toroidal LHC Apparatus. <http://atlas.web.cern.ch/>. [cited on p. 108]
- [5] CAIDA: Internet Measurement Tool Taxonomy. <http://www.caida.org/tools/taxonomy/measurement/>. [cited on p. 27]
- [6] Cisco IOS NetFlow. <http://www.cisco.com/web/go/netflow>. [cited on p. 27]
- [7] Configuring J-Flow Statistics. <http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ip-services/html/ip-jflow-stats-config.html>. [cited on p. 27]
- [8] DataTAG Home Page. <http://datatag.web.cern.ch/datatag/>. [cited on p. 42]
- [9] distributed.net. <http://distributed.net/>. [cited on p. 200]
- [10] EGEE Portal: Enabling Grids for E-science. <http://www.eu-egee.org/>. [cited on p. 66, 206]
- [11] EMANICS - European Network of Excellence for the Management of Internet Technologies and Complex Services. <http://www.emanics.org/>. [cited on p. 107]
- [12] European Grid Infrastructure. <http://www.egi.eu/>. [cited on p. 163, 204]
- [13] GeoBoy (discontinued). <http://www.ndgsoftware.com/~ndgprod/page11.html>. [cited on p. 21]
- [14] Grid Scheduling Architecture RG (GSA-RG). <http://forge.gridforum.org/sf/projects/gsa-rg>. [cited on p. 42]
- [15] Hawkeye. <http://www.cs.wisc.edu/condor/hawkeye/>. [cited on p. 54]
- [16] JANET, the UK's Education and Research Network. <http://www.ja.net/>. [cited on p. 108]
- [17] JNCA (Java Netflow Collector & Analyzer). <http://jnca.sourceforge.net/>. [cited on p. 27]

- [18] Load Monitor - NGS. <http://ngs.ac.uk/load-monitor>. [cited on p. 68]
- [19] NCS - Network Characterization Service. <http://www-didc.lbl.gov/NCS>. [cited on p. 23]
- [20] NeoTrace, A Graphical Traceroute Tool (discontinued). <http://www.neoworx.com/products/neotrace/default.asp>. [cited on p. 21]
- [21] NetBoy - Complete Real-time Visual Internet and LAN network Monitoring Suite! <http://www.amtsoft.com/netboy/>. [cited on p. 26]
- [22] NeTraMet. <http://www.caida.org/tools/measurement/netramet/>. [cited on p. 27]
- [23] Nettetst. <http://dsd.lbl.gov/~boverhof/nettest.html>. [cited on p. 23]
- [24] Network Measurement Working Group. <http://nmwg.internet2.edu>. [cited on p. 42]
- [25] NEye (Network Eye). <http://freshmeat.net/projects/neye>. [cited on p. 27]
- [26] NGS Current Members. <http://ngs.ac.uk/member-sites/current-members>. [cited on p. 204]
- [27] nuTTCP. <ftp://ftp.lcp.nrl.navy.mil/pub/nuttcp/>. [cited on p. 23]
- [28] OMII-UK: GridSAM. <http://www.omii.ac.uk/wiki/GridSAM>. [cited on p. 204]
- [29] OMII-UK: OGSA-DAI. <http://www.omii.ac.uk/wiki/OGSADAI>. [cited on p. 204]
- [30] OpenLDAP. <http://www.openldap.org/>. [cited on p. 43]
- [31] Oxford e-Research Centre provides cloud computing resource. <http://www.oerc.ox.ac.uk/news/oxford-e-research-centre-provides-cloud-computing>. [cited on p. 163]
- [32] Pcapture - Tool for Capturing Packets from the Network. <http://linux.maruhn.com/sec/pcapture.html>. [cited on p. 26]
- [33] PingPlotter. <http://www.pingplotter.com/>. [cited on p. 26]
- [34] POV-Ray - The Persistence of Vision Raytracer. <http://www.povray.org/>. [cited on p. 108]
- [35] rsyslog. <http://www.rsyslog.com/>. [cited on p. 129]
- [36] Synack. <http://www-iepm.slac.stanford.edu/tools/synack>. [cited on p. 22]
- [37] Test Traffic Measurements Service. <http://www.ripe.net/test-traffic/>. [cited on p. 20]
- [38] The Globus Alliance. <http://www.globus.org/>. [cited on p. 199]
- [39] The Large Hadron Collider. <http://lhc.web.cern.ch>. [cited on p. 4]
- [40] The LHC Computing Grid Project. <http://lhc.web.cern.ch/LCG>. [cited on p. 4]
- [41] The NLANR Active Measurement Pproject. <http://amp.nlanr.net/active/>. [cited on p. 20, 48]

- [42] The Prefix WhoIs Project - WhoB & Layer Four Traceroute (LFT). <http://pwhois.org/lft/>. [cited on p. 21]
- [43] The Scripps Research Institute - Today's Research, Tomorrow's Cures. <http://www.scripps.edu/>. [cited on p. 127]
- [44] Traceping. http://av9.physics.ox.ac.uk:8097/www/traceping_description.html. [cited on p. 21]
- [45] TTCP Utility. <http://www.pcausa.com/Utilities/pcattcp.htm>. [cited on p. 23]
- [46] VisualRoute. <http://www.visualroute.com/>. [cited on p. 21]
- [47] WAND Network Research Group: libtrace. <http://research.wand.net.nz/software/libtrace.php>. [cited on p. 26]
- [48] WAND Network Research Group: Visualisation. <http://research.wand.net.nz/software/visualisation.php>. [cited on p. 26]
- [49] Wren. <http://www.cs.wm.edu/~lowekamp/wren.html>. [cited on p. 49]
- [50] Ahmar Abbas. *Grid Computing: A Practical Guide to Technology and Applications*. Charles River Media, Rockland, MA, USA, 2003. [cited on p. 198]
- [51] David Abramson, Rajkumar Buyya, and Jonathan Giddy. A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. *Future Generation Computer Systems*, 18(8):1061–1074, October 2002. [cited on p. 4]
- [52] David Abramson, Rok Susic, Jonathan Giddy, and Martin Cope. The Laboratory Bench: Distributed Computing for Parametised Simulations. In *Proceedings of the Parallel Computing and Transputers Conference*, pages 2–7, November 1994. [cited on p. 37]
- [53] Davide Adami, Stefano Giordano, and Michele Pagano. Dynamic Network Resources Allocation in Grids through a Grid Network Resource Broker. In Franco Davoli, Norbert Meyer, Roberto Pugliese, and Sandro Zappatore, editors, *Grid Enabled Remote Instrumentation*, Signals and Communication Technology, pages 115–130. Springer US, 2009. [cited on p. 64]
- [54] Jay Aikat, Jasleen Kaur, F. Donelson Smith, and Kevin Jeffay. Variability in TCP Round-Trip Times. In *Proceedings of the Third ACM SIGCOMM Conference on Internet Measurement (IMC'03)*, pages 279–284, New York, NY, USA, October 2003. ACM. [cited on p. 22]
- [55] Hiroshi Akima. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. *Journal of the ACM*, 17(4):589–602, October 1970. [cited on p. 98]
- [56] Rashid J. Al-Ali, Omer F. Rana, David W. Walker, Sanjay Jha, and Shaleeza Sohail. G-QoS: Grid Service Discovery Using QoS Properties. *Computers and Artificial Intelligence*, 21(4):363–382, 2002. [cited on p. 64]

- [57] William Allcock (editor), Joe Bester, John Bresnahan, Sam Meder, Pawel Plaszczak, and Steven Tuecke. *GridFTP: Protocol Extensions to FTP for the Grid*. Global Grid Forum, April 2003. <http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf>. [cited on p. 4]
- [58] Gabrielle Allen, Werner Benger, Tom Goodale, Hans-Christian Hege, Gerd Lanfermann, André Merzky, Thomas Radke, Edward Seidel, and John Shalf. The Cactus Code: A Problem Solving Environment for the Grid. In *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC'00)*, pages 253–260, Washington, DC, USA, August 2000. IEEE Computer Society. [cited on p. 4]
- [59] Mark Allman. A Web Server's View of the Transport Layer. *ACM SIGCOMM Computer Communication Review*, 30(5):10–20, October 2000. [cited on p. 22]
- [60] Mark Allman, Wesley M. Eddy, and Shawn Ostermann. Estimating Loss Rates with TCP. *ACM SIGMETRICS Performance Evaluation Review*, 31(3):12–24, December 2003. [cited on p. 25]
- [61] Mark Allman, Vern Paxson, and W. Richard Stevens. *RFC 2581: TCP Congestion Control*. Internet Engineering Task Force, April 1999. [cited on p. 63]
- [62] George S. Almasi and Allan Gottlieb. *Highly Parallel Computing (2nd ed.)*. Benjamin-Cummings, Redwood City, CA, USA, 1994. [cited on p. 35]
- [63] Giovanni Aloisio, Massimo Cafaro, Italo Epicoco, Ro Fiore, Daniele Lezzi, Maria Mirto, and Silvia Mocavero. iGrid, a Novel Grid Information Service. In *Proceedings of the First European Grid Conference (EGC'05)*, pages 506–515. Springer-Verlag, February 2005. [cited on p. 50]
- [64] Giovanni Aloisio, Massimo Cafaro, Italo Epicoco, Daniele Lezzi, Maria Mirto, and Silvia Mocavero. The Design and Implementation of the GridLab Information Service. In *Proceedings of the Second International Workshop on Grid and Cooperative Computing (GCC'03)*, number 3032 in Lecture Notes in Computer Science, pages 131–138. Springer-Verlag, December 2003. [cited on p. 54]
- [65] David P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (Grid'04)*, pages 4–10, Washington, DC, USA, November 2004. IEEE Computer Society. [cited on p. 200]
- [66] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: An Experiment in Public-Resource Computing. *Communications of the ACM*, 45(11):56–61, November 2002. [cited on p. 4, 108]
- [67] Paolo Andreetto, Sergio Andreatto, Giuseppe Avellino, Stefano Beco, Andrea Cavallini, Marco Cecchi, Vincenzo Ciaschini, Alvisè Dorise, Francesco Giacomini, Alessio

- Gianelle, Ugo Grandinetti, Andrea Guarise, Andrea Krop, Roberto Lops, Alessandro Maraschini, Vincenzo Martelli, Moreno Marzolla, Marco Mezzadri, Elisabetta Molinari, Salvatore Monforte, Fabrizio Pacini, Marco Pappalardo, Andrea Parrini, Giuseppe Pata-
nia, Luca Petronzio, Rosario Piro, Maurizio Porciani, Francesco Prelz, David Rebatto,
Elisabetta Ronchieri, Massimo Sgaravatto, Valerio Venturi, and Luigi Zangrando. The
gLite Workload Management System. *Journal of Physics Conference Series*, 119(6), July
2008. [cited on p. 125, 206]
- [68] Demetres Antoniadis, Manos Athanatos, Antonis Papadogiannakis, Evangelos P.
Markatos, and Constantine Dovrolis. Available Bandwidth Measurement As Simple
As Running wget. In Mark Allman and Matthew Roughan, editors, *Proceedings of
the Passive and Active Measurement Conference (PAM'06)*, pages 61–70, March 2006.
[cited on p. 22, 24]
- [69] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy
Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia.
A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, April 2010.
[cited on p. 163, 164]
- [70] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman,
Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute Anoma-
lies with Paris traceroute. In *Proceedings of the Sixth ACM SIGCOMM Conference on
Internet Measurement (IMC'06)*, pages 153–158, New York, NY, USA, October 2006. ACM.
[cited on p. 21]
- [71] Björn Augustsson. Xtraceroute. <http://freshmeat.net/projects/Xtraceroute/>.
[cited on p. 21]
- [72] Mark Baker and Garry Smith. GridRM: A Resource Monitoring Architecture for the Grid.
In *Proceedings of the Third International Workshop on Grid Computing (Grid'02)*, pages
268–273, London, UK, November 2002. Springer-Verlag. [cited on p. 54]
- [73] Amnon Barak and Oren La'adan. The MOSIX Multicomputer Operating System for High
Performance Cluster Computing. *Future Generation Computer Systems*, 13(4–5):361–
372, March 1998. [cited on p. 37]
- [74] Chadi Barakat, Eitan Altman, and Walid Dabbous. On TCP Performance in a Heteroge-
neous Network: A Survey. *IEEE Communications Magazine*, 38(1):40–46, January 2000.
[cited on p. 17]
- [75] Chaitanya Baru, Reagan Moore, Arcot Rajasekar, and Michael Wan. The SDSC Storage
Resource Broker. In *Proceedings of the 1998 Conference of the Centre for Advanced
Studies on Collaborative Research (CASCON'98)*. IBM Press, November 1998. [cited on p. 4,
96]

- [76] Daniel M. Batista, Cesar G. Chaves, and Nelson L. S. da Fonseca. Scheduling Virtual Machines and Grid Tasks on Clouds. In *Proceedings of the First Workshop on Network Virtualization and Intelligence for Future Internet (WNetVirt'10)*, April 2010. [cited on p. 164]
- [77] Daniel M. Batista and Nelson L. S. da Fonseca. A Survey of Self-Adaptive Grids. *IEEE Communications Magazine*, 48(7):94–100, July 2010. [cited on p. 71]
- [78] Nadia Ben Azzouna, Fabrice Clérot, Christine Fricker, and Fabrice Guillemin. A Flow-based Approach to Modeling ADSL Traffic on an IP Backbone Link. *Annals of Telecommunications*, 59(11):1260–1299, November 2004. [cited on p. 59]
- [79] Peter Benko and Andras Veres. A Passive Method for Estimating End-to-End TCP Packet Loss. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'02)*, volume 3, pages 2609–2613. IEEE Computer Society, November 2002. [cited on p. 22, 25]
- [80] Francine Berman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter High-Performance Schedulers, pages 279–309. Morgan Kaufmann, San Francisco, CA, USA, 1998. [cited on p. 71]
- [81] Francine Berman, Geoffrey Fox, and Anthony J.G. Hey. The Grid: Past, Present, Future. In Francine Berman, Geoffrey Fox, and Anthony J. G. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, pages 9–50. John Wiley & Sons, 2003. [cited on p. 198]
- [82] Saleem N. Bhatti, Søren-Aksel Sørensen, Peter Clarke, and Jon Crowcroft. Network QoS for Grid Systems. *International Journal of High Performance Computing Applications*, 17(3):219–236, August 2003. [cited on p. 64]
- [83] Steven Blake, David L. Black, Mark A. Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. *RFC 2475: An Architecture for Differentiated Services*. Internet Engineering Task Force, December 1998. [cited on p. 63]
- [84] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. *Web Services Architecture*. W3C Working Group Notes. The World Wide Web Consortium, February 2004. [cited on p. 202]
- [85] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1970. [cited on p. 97]
- [86] Robert Braden. *RFC 1122: Requirements for Internet Hosts – Communication Layers*. Internet Engineering Task Force, October 1989. [cited on p. 20]
- [87] Nevil Brownlee. Network Management and Realtime Traffic Flow Measurement. *Journal of Network and Systems Management*, 6(2):223–228, 1998. [cited on p. 27]

- [88] Nevil Brownlee. Using NeTraMet for Production Traffic Measurement. In *Proceedings of the 2001 IEEE/IFIP International Symposium on Integrated Network Management*, pages 213–226, May 2001. [cited on p. 27]
- [89] Dario Bruneo, Marco Scarpa, and Antonio Puliafito. A GSPN Model to Analyze Performance Parameters in gLite Grids. In *Proceedings of the Seventeenth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'08)*, pages 198–203, Washington, DC, USA, June 2008. IEEE Computer Society. [cited on p. 122]
- [90] Stephen Burke, Simone Campana, Elisa Lanciotti, Patricia Méndez Lorenzo, Vincenzo Miccio, Christopher Nater, Roberto Santinelli, and Andrea Sciabà. *gLite 3.1 User Guide*, 1.2 edition, April 2009. [cited on p. 208]
- [91] Prasad Calyam, Jialu Pu, Weiping Mandrawa, and Ashok Krishnamurthy. OnTimeDetect: Dynamic Network Anomaly Notification in perfSONAR Deployments. In *Proceedings of the Eighteenth IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'10)*, pages 328–337, Los Alamitos, CA, USA, August 2010. IEEE Computer Society. [cited on p. 161]
- [92] Agustín Caminero, Omer Rana, Blanca Caminero, and Carmen Carrión. Performance Evaluation of an Autonomic Network-aware Metascheduler for Grids. *Concurrency and Computation: Practice & Experience*, 21(13):1692–1708, September 2009. [cited on p. 64]
- [93] Mario Cannataro and Domenico Talia. The Knowledge Grid. *Communications of the ACM*, 46(1):89–93, January 2003. [cited on p. 60]
- [94] Robert L. Carter and Mark E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. *Performance Evaluation*, 27-28:297–318, October 1996. [cited on p. 24]
- [95] David A. Case, Thomas E. Cheatham, Tom Darden, Holger Gohlke, Ray Luo, Kenneth M. Merz, Alexey Onufriev, Carlos Simmerling, Bing Wang, and Robert J. Woods. The Amber Biomolecular Simulation Programs. *Journal of Computational Chemistry*, 26(16):1668–1688, December 2005. [cited on p. 128]
- [96] Jeffrey Case, Mark Fedor, Martin Lee Schoffstall, and James Davin. *RFC 1157: Simple Network Management Protocol (SNMP)*. Internet Engineering Task Force, May 1990. [cited on p. 26]
- [97] Jeffrey Case, Russ Mundy, David Partain, and Bob Stewart. *RFC 2570: Introduction to Version 3 of the Internet-standard Network Management Framework*. Internet Engineering Task Force, April 1999. [cited on p. 26]
- [98] Rui Castro, Mark Coates, Gang Liang, Robert Nowak, and Bin Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499–517, August 2004. [cited on p. 29]

- [99] Edmond W.W. Chan, Xiapu Luo, and Rocky K.C. Chang. Reliable Asymmetric Path Capacity Measurement. In *Proceedings of the Fifth International Student Workshop on Emerging Networking Experiments and Technologies (Co-Next'09)*, pages 55–56, New York, NY, USA, December 2009. ACM. [cited on p. 22]
- [100] Christopher Chatfield. *The Analysis of Time Series: An introduction*. Chapman & Hall Texts in Statistical Science. Chapman & Hall, fifth edition, 1996. [cited on p. 99]
- [101] Ling-Jyh Chen, Tony Sun, Li Lao, Guang Yang, M.Y. Sanadidi, and Mario Gerla. Estimation Link Capacity in High Speed Networks. In *Proceedings of the Fifth IFIP International Conference on Networking*, pages 98–109. Springer, May 2006. [cited on p. 24]
- [102] Ling-Jyh Chen, Tony Sun, Guang Yang, M.Y. Sanadidi, and Mario Gerla. AdHoc Probe: Path Capacity Probing in Wireless Ad Hoc Networks. In *Proceedings of the First International Conference on Wireless Internet*, pages 156–163, July 2005. [cited on p. 24]
- [103] Ling-Jyh Chen, Tony Sun, Guang Yang, M.Y. Sanadidi, and Mario Gerla. End-to-End Asymmetric Link Capacity Estimation. In *Proceedings of the Fourth IFIP International Conference on Networking*, pages 780–791, 2005. [cited on p. 24]
- [104] Thomas M. Chen and Lucia Hu. Internet Performance Monitoring. *Proceedings of the IEEE*, 90(9):1592–1603, September 2002. [cited on p. 27, 68]
- [105] Brian Cheney. Membrane Permeation. <http://www.ngs.ac.uk/membrane-permeation>. [cited on p. 142]
- [106] Ann Chervenak, Robert Schuler, Carl Kesselman, Scott Koranda, and Brian Moe. Wide Area Data Replication for Scientific Collaborations. In *Proceedings of the Sixth IEEE/ACM International Workshop on Grid Computing (Grid'05)*, pages 1–8, November 2005. [cited on p. 95]
- [107] Bryan Christianson. WhatRoute. <http://www.whatroute.net/>. [cited on p. 21]
- [108] Augusto Ciuffoletti, Tiziana Ferrari, Antonia Ghiselli, and Maria Cristina Vistoli. Architecture of Monitoring Elements for the Network Element Modeling in a Grid Infrastructure. In *Proceedings of the 2003 Conference for Computing in High Energy and Nuclear Physics (CHEP'03)*, March 2003. [cited on p. 54]
- [109] Augusto Ciuffoletti, Leonardo Merola, Francesco Palmieri, and Guido Pardi, Silvio and Russo. Optimizing Bulk Data Transfers Using Network Measurements: A practical case. *Journal of Physics: Conference Series*, 219(6), April 2010. [cited on p. 102]
- [110] Benoit Claise (editor), Stewart Bryant, Simon Leinen, Thomas Dietz, and Brian H. Trammell. *RFC 5101: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*. Internet Engineering Task Force, January 2008. [cited on p. 27]

- [111] Benoit Claise (editor), Ganesh Sadasivan, Vamsi Valluri, and Martin Djernaes. *RFC 3954: Cisco Systems NetFlow Services Export Version 9*. Internet Engineering Task Force, October 2004. [cited on p. 27]
- [112] Andy Cooke, Alasdair J G Gray, Lisha Ma, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Rob Byrom, Laurence Field, Steve Hicks, Jason Leake, Manish Soni, and Antony Wilson. R-GMA: An Information Integration System for Grid Monitoring. In *Proceedings of the Eleventh International Conference on Cooperative Information Systems (CoopIS'03)*, pages 462–481, November 2003. [cited on p. 50]
- [113] A. Corrente and L. Tura. Security Performance Analysis of SNMPv3 with Respect to SNMPv2c. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'04)*, volume 1, pages 729–742, April 2004. [cited on p. 30]
- [114] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Proceedings of the Twenty-Fourth International Conference on Distributed Computing Systems (ICDCS'04)*, pages 178–187, Los Alamitos, CA, USA, March 2004. IEEE Computer Society. [cited on p. 52]
- [115] Les Cottrell. Pingroute. <http://www.slac.stanford.edu/comp/net/pingroute.readme>. [cited on p. 21]
- [116] Chuck Cranor, Yuan Gao, Theodore Johnson, Vlaidslav Shkapenyuk, and Oliver Spatscheck. Gigascope: High performance network monitoring with an SQL interface. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD'02)*, page 623, New York, NY, USA, June 2002. ACM. [cited on p. 54]
- [117] Joel M. Crichlow. *An Introduction to Distributed and Parallel Computing*. Prentice-Hall, Upper Saddle River, NJ, USA, 1988. [cited on p. 33]
- [118] Jon Crowcroft, Steven Hand, Richard Mortier, Timothy Roscoe, and Andrew Warfield. QoS's Downfall: At the bottom, or not at all! In *Proceedings of the ACM SIGCOMM Workshop on Revisiting IP QoS (RIPQoS'03)*, pages 109–114, New York, NY, USA, October 2003. ACM. [cited on p. 30]
- [119] David E. Culler, Jaswinder Pal Singh, and Anoop Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, San Francisco, CA, USA, 1997. [cited on p. 36]
- [120] K. Czajkowski, S. Fitzgerald, Ian Foster, and Carl Kesselman. Grid Information Services for Distributed Resource Sharing. In *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, pages 181–194, August 2001. [cited on p. 43]

- [121] David de Roure, Mark A. Baker, Nicholas R. Jennings, and Nigel R. Shadbolt. The Evolution of the Grid. In Fran Berman, Geoffrey Fox, and Anthony J.G. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, pages 65–100. John Wiley & Sons, March 2003. [cited on p. 198]
- [122] Stijn De Smet, Pieter Thysebaert, Bruno Volckaert, Filip De Turck, Bart Dhoedt, and Piet Demeester. A Performance Oriented Grid Monitoring Architecture. In *Proceedings of the Second Workshop on End-to-End Monitoring Techniques and Services (E2EMON)*, pages 23–28, October 2004. [cited on p. 68]
- [123] Thomas A. Defanti, Ian Foster, Michael E. Papka, Rick Stevens, and Tim Kuhfuss. Overview of the I-WAY: Wide Area Visual Supercomputing. *The International Journal of Supercomputer Applications and High Performance Computing*, 10(2):123–131, June 1996. [cited on p. 62, 199]
- [124] Luca Deri and Stefano Suin. Ntop: Beyond ping and traceroute. In *Proceedings of the Tenth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'99)*, pages 271–284, London, UK, October 1999. Springer-Verlag. [cited on p. 26]
- [125] Tony Dewitt, Thomas Gross, Bruce Lowekamp, Nancy Miller, Peter Steenkiste, Jaspal Subhlok, and Dean Sutherland. ReMoS: A Resource Monitoring System for Network-Aware Applications. Technical Report CMU-CS-97-194, Carnegie Mellon School of Computer Science, 1997. [cited on p. 44]
- [126] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What Do Packet Dispersion Techniques Measure? In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*, pages 905–914. IEEE, April 2001. [cited on p. 24]
- [127] Allen B. Downey. Using pathchar to Estimate Internet Link Characteristics. *ACM SIGCOMM Computer Communication Review*, 29(4):241–250, October 1999. [cited on p. 23, 29]
- [128] Robert E. Duke and Lee G. Pedersen. PMEMD 3. University of North Carolina-Chapel Hill, 2003. [cited on p. 128]
- [129] S.G. Dykes, K.A. Robbins, and C.L. Jeffery. An Empirical Evaluation of Client-side Server Selection Algorithms. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, volume 3, pages 1361–1370. IEEE, March 2000. [cited on p. 22]
- [130] Thomas Dzubin. fping - A program to ping hosts in parallel. <http://fping.sourceforge.net/>. [cited on p. 20]

- [131] Wesley M. Eddy. *RFC 4987: TCP SYN Flooding Attacks and Common Mitigations*. Internet Engineering Task Force, August 2007. [cited on p. 22]
- [132] EGEE. EGEE Network Performance Monitoring - e2emonit. <http://www.egee-npm.org/e2emonit/>. [cited on p. 55]
- [133] EGEE. *Job Description Language Attributes Specification for the gLite Middleware (v0.9)*, November 2007. [cited on p. 207]
- [134] Yehia El-khatib, Chris Edwards (editors), Dragana Damjanovic, Werner Hei, Michael Welzl, Burkhard Stiller, Paulo Gonalves, Patrick Loiseau, Pascale Vicat-Blanc Primet, Linghang Fan, Jin Wu, Yichao Yang, Yanbo Zhou, Yao Hu, Li Li, Sha Li, Shaohua Liu, Xiaolei Ma, Ming Yang, Liang Zhang, Wang Kun, ZhiYong Liu, Yanming Chen, Tao Liu, Chen Zhang, , and Lin Zhang. *Survey of Grid Simulators and a Network-level Analysis of Grid Applications*. Technical report, University of Innsbruck, April 2008. Pages 34–83. Available from <http://www.comp.lancs.ac.uk/~elkhatib/docs/D2.0.pdf>. [cited on p. 59]
- [135] Yehia El-khatib and Christopher Edwards. A Survey-based Study of Grid Traffic. In *Proceedings of the First International Conference on Networks for Grid Applications (GridNets'07)*, pages 1–8. ICST, October 2007. [cited on p. 59, 75]
- [136] Yehia El-khatib and Christopher Edwards. Diversity of Grid Traffic: A Survey-based Study. In *Proceedings of the Eighth EPSRC Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET'07)*, pages 56–61, Liverpool, UK, June 2007. The School of Computing and Mathematical Sciences, Liverpool John Moore's University. [cited on p. 59, 75]
- [137] Yehia El-khatib, Christopher Edwards, Michael Mackay, and Gareth Tyson. Providing Grid Schedulers with Passive Network Measurements. In *Proceedings of the Eighteenth International Conference on Computer Communications and Networks (ICCCN'09)*, pages 1–6, Washington, DC, USA, 2009. IEEE Computer Society. [cited on p. 106, 162]
- [138] Jeremy Elson. tcpflow – TCP Flow Recorder. <http://www.circlemud.org/~jelson/software/tcpflow/>. [cited on p. 26]
- [139] Dietmar W. Erwin. UNICORE: A Grid Computing Environment. *Concurrency and Computation: Practice & Experience*, 14(13-15):1395–1410, December 2002. [cited on p. 3]
- [140] Damien Miller et al. flowd - Fast, Secure and Flexible NetFlow Collector. <http://www.mindrot.org/projects/flowd/>. [cited on p. 27]
- [141] Claudio Favi and Grenville Armitage. Dynamic Performance Limits of the Benko-Veres Passive TCP Packet Loss Estimation Algorithm. In *Proceedings of the Australian Telecommunications Networks & Applications Conference (ATNAC'04)*, pages 336–340, Sydney, Australia, December 2004. [cited on p. 25]

- [142] Narcis Fernandes-Fuentes. High-throughput Virtual Drug Screening Using the NGS. <http://www.ngs.ac.uk/high-throughput-virtual-drug-screening-using-ngs>. [cited on p. 142]
- [143] Tiziana Ferrari and Francesco Giacomini. Network Monitoring for Grid Performance Optimization. *Computer Communications*, 27(14):1357–1363, September 2004. Network Support for Grid Computing. [cited on p. 51]
- [144] Tiziana Ferrari (editor), Jim Austin, Peter Clarke, Martyn Fletcher, Mark Gaynor, Tom Jackson, Gigi Karmous-Edwards, Peter Kunszt, Mark J. Leese, Jason Leigh, Paul D. Meador, Inder Monga, Volker Sander, and Peter Tomsu. Grid Network Services Use Cases from the e-Science Community. Technical Report GFD-I.122, Open Grid Forum, May 2007. Version: 2.12. [cited on p. 58]
- [145] Laurence Field and Markus W. Schulz. Grid Deployment Experiences: The path to a production quality LDAP based grid information system. In *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP'04)*, pages 723–726, Geneva, Switzerland, September 2004. CERN. [cited on p. 68]
- [146] Silvia Figueira, Sumit Naiksatam, Howard Cohen, Doug Cutrell, Paul Daspit, David Gutierrez, Doan B. Hoang, Tal Lavian, Joe Mambretti, Steve Merrill, and Franco Travostino. DWDM-RAM: Enabling Grid Services with Dynamic Optical Networks. In *Proceedings of the Fourth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'04)*, pages 707–714, April 2004. [cited on p. 141]
- [147] Steven Fitzgerald, Ian Foster, Carl Kesselman, Gregor von Laszewski, Warren Smith, and Steven Tuecke. A Directory Service for Configuring High-Performance Distributed Computations. In *Proceedings of the Sixth IEEE International Symposium on High Performance Distributed Computing (HPDC-6)*, pages 365–375, August 1997. [cited on p. 43, 61]
- [148] Sally Floyd, Tom Henderson, and Andrei Gurtov. *RFC 3782: The NewReno Modification to TCP's Fast Recovery Algorithm*. Internet Engineering Task Force, April 2004. [cited on p. 63]
- [149] Sally Floyd and Eddie Kohler. Internet Research Needs Better Models. *ACM SIGCOMM Computer Communication Review*, 33(1):29–34, January 2003. [cited on p. 161]
- [150] Sally Floyd and Eddie Kohler. *Tools for the Evaluation of Simulation and Testbed Scenarios*. Internet Engineering Task Force, fifth edition, February 2008. <http://tools.ietf.org/html/draft-irtf-tmrg-tools-05>. [cited on p. 22]
- [151] Sally Floyd and Vern Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001. [cited on p. 14, 122]

- [152] The Distributed Management Task Force. Common Information Model. <http://www.dmtf.org/standards/cim>. [cited on p. 42]
- [153] Grant Foster. Wavelets for Period Analysis of Unevenly Sampled Time Series. *Astronomical Journal*, 112:1709–1729, October 1996. [cited on p. 98]
- [154] Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, September 1997. [cited on p. 3, 46, 58, 62, 71, 199]
- [155] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, USA, 2003. [cited on p. 3, 58]
- [156] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. Grid Services for Distributed System Integration. *IEEE Computer*, 35(6):37–46, June 2002. [cited on p. 4, 202]
- [157] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In Francine Berman, Geoffrey Fox, and Anthony J. G. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, pages 217–248. John Wiley & Sons, March 2003. [cited on p. 4, 202]
- [158] Ian Foster, Hiro Kishimoto, Andreas Savva, Dave Berry, Abdeslem Djaoui, Andrew Grimshaw, Bill Horn, Fred Maciel, Frank Siebenlist, Ravi Subramaniam, Jem Treadwell, and Jeffrin von Reich. *The Open Grid Services Architecture v1.0*, January 2005. Open Grid Forum GFD-I.030. [cited on p. 4, 202]
- [159] Ian Foster, Hiro Kishimoto, Andreas Savva, Dave Berry, Abdeslem Djaoui, Andrew Grimshaw, Bill Horn, Fred Maciel, Frank Siebenlist, Ravi Subramaniam, Jem Treadwell, and Jeffrin von Reich. *The Open Grid Services Architecture v1.5*, July 2006. Open Grid Forum GFD-I.080. [cited on p. 203]
- [160] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud Computing and Grid Computing 360-Degree Compared. In *Proceedings of the Grid Computing Environments Workshop (GCE'08)*, pages 1–10, November 2008. [cited on p. 58, 163]
- [161] Geoffrey Fox, Galip Aydin, Hasan Bulut, Harshawardhan Gadgil, Shrideep Pallickara, Marlon Pierce, and Wenjun Wu. Management of Real-time Streaming Data Grid Services: Research Articles. *Concurrency and Computation: Practice & Experience*, 19(7):983–998, May 2007. [cited on p. 61]
- [162] James Frey, Todd Tannenbaum, Miron Livny, Ian Foster, and Steven Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, 5(3):237–246, July 2002. [cited on p. 4]

- [163] Aleš Friedl, Sven Ubik, Alexandros Kapravelos, Michalis Polychronakis, and Evangelos P. Markatos. Realistic Passive Packet Loss Measurement for High-Speed Networks. In *Proceedings of the First International Workshop on Traffic Monitoring and Analysis (TMA'09)*, pages 1–7, May 2009. [cited on p. 25]
- [164] Michael Frumkin and Rob F. Van der Wijngaart. NAS Grid Benchmarks: A Tool for Grid Space Exploration. *Cluster Computing*, 5(3):247–255, July 2002. [cited on p. 126]
- [165] Wei Fu and Qian Huang. GridEye: A Service-oriented Grid Monitoring System with Improved Forecasting Algorithm. In *Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops (GCCW'06)*, pages 5–12, October 2006. [cited on p. 54, 68]
- [166] Fabrizio Gagliardi, Bob Jones, Mario Reale, and Stephen Burke. European DataGrid Project: Experiences of Deploying a Large Scale Testbed for E-science Applications. In *Performance Evaluation of Complex Systems: Techniques and Tools*, volume 2459, pages 255–264. Springer-Verlag, September 2002. [cited on p. 4, 50]
- [167] Arijit Ganguly, Abhishek Agrawal, P. Oscar Boykin, and Renato J. Figueiredo. WOW: Self-Organizing Wide Area Overlay Networks of Virtual Workstations. In *Proceedings of the Fifteenth International Symposium on High-Performance Distributed Computing (HPDC-15)*, pages 30–42, Los Alamitos, CA, USA, June 2006. IEEE Computer Society. [cited on p. 65]
- [168] Peng Gao, Tao Liu, Yanming Chen, Xingyao Wu, Yehia El-khatib, and Christopher Edwards. The Measurement and Modeling of a P2P Streaming Video Service. In *Proceedings of the Second International Conference on Networks for Grid Applications (GridNets'08)*, pages 24–34. Springer, October 2008. [cited on p. 59]
- [169] Lee Garber. Denial-of-Service Attacks Rip the Internet. *IEEE Computer*, 33(4):12–17, April 2000. [cited on p. 22]
- [170] Albrecht Gebhardt, Thomas Petzoldt, and Martin Maechler. akima: Interpolation of Irregularly Spaced Data. <http://cran.r-project.org/web/packages/akima/>, November 2009. [cited on p. 99]
- [171] Michael Gerndt, Roland Wismüller, Zoltán Balaton, Gábor Gombás, Péter Kacsuk, Zsolt Németh, Norbert Podhorszki, Hong-Linh Truong, Thomas Fahringer, Marian Bubak, Erwin Laure, and Thomas Margalef. Performance Tools for the Grid: State of the Art and Future. Technical report, Technische Universität München, January 2004. [cited on p. 54]
- [172] Moustafa Ghanem, Yike Guo, John Hassard, Michelle Osmond, and Mark Richards. Sensor Grids for Air Pollution Monitoring. In *Proceedings of the Third UK e-Science All-hands Meeting (AHM'04)*, September 2004. [cited on p. 60]

- [173] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. The Flattening Internet Topology: Natural evolution, unsightly barnacles or contrived collapse? In *Proceedings of the Ninth International Conference on Passive and Active Network Measurement (PAM'08)*, pages 1–10, Berlin, Heidelberg, 2008. Springer-Verlag. [cited on p. 159]
- [174] GLUE-WG. GLUE Schema. <http://glueschema.forge.cnaf.infn.it/>. [cited on p. 41]
- [175] Kalman Graffi, Sebastian Kaune, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Load Balancing for Multimedia Streaming in Heterogeneous Peer-to-peer Systems. In *Proceedings of the Eighteenth International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '08)*, pages 99–104, New York, NY, USA, 2008. ACM. [cited on p. 159]
- [176] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, and Paul F. Reynolds Jr. Legion: The Next Logical Step Toward a Nationwide Virtual Computer. Technical Report CS-94-20, Department of Computer Science, University of Virginia, Charlottesville, Virginia, USA, June 1994. [cited on p. 37]
- [177] Herbert R. J. Grosch. High Speed Arithmetic: The Digital Computer as a Research Tool. *Journal of the Optical Society of America*, 43(4):306–310, April 1953. [cited on p. 33]
- [178] Yunhong Gu and Robert L. Grossman. UDT: UDP-based Data Transfer for High-Speed Wide Area Networks. *Computer Networks*, 51(7):1777–1799, May 2007. [cited on p. 75]
- [179] Mehmet H. Gunes and Kamil Sarac. Analyzing Router Responsiveness to Active Measurement Probes. In *Proceedings of the Tenth International Conference on Passive and Active Network Measurement (PAM'09)*, pages 23–32, Berlin, Heidelberg, 2009. Springer-Verlag. [cited on p. 18]
- [180] Dan Gunter and James Magowan. An Analysis of “Top N” Event Descriptions. Technical report, Global Grid Forum, January 2004. [cited on p. 42]
- [181] Dan Gunter, Brian Tierney, Keith Jackson, Jason Lee, and Martin Stoufer. Dynamic Monitoring of High-Performance Distributed Applications. In *Proceedings of the Eleventh IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, page 163, Washington, DC, USA, July 2002. IEEE Computer Society. [cited on p. 48]
- [182] Dan Gunter, Brian L. Tierney, Aaron Brown, Martin Swany, John Bresnahan, and Jennifer M. Schopf. Log Summarization and Anomaly Detection for Troubleshooting Distributed Systems. In *Proceedings of the Eighth IEEE/ACM International Conference on Grid Computing (Grid'07)*, pages 226–234, September 2007. [cited on p. 129]
- [183] Andreas Hanemann, Jeff W. Boote, Eric L. Boyd, Jérôme Durand, Loukik Kudarimoti, Roman Lapacz, D. Martin Swany, Szymon Trocha, and Jason Zurawski. PerfSONAR: A Service Oriented Architecture for Multi-domain Network Monitoring. In *Proceedings*

- of the Third International Conference on Service-Oriented Computing (ICSOC'05)*, pages 241–254, December 2005. [cited on p. 51]
- [184] Terence Harmer, Peter Wright, Christina Cunningham, and Ron Perrott. Provider-Independent Use of the Cloud. In *Proceedings of the Fifteenth International Euro-Par Conference on Parallel Processing (Euro-Par'09)*, pages 454–465, Berlin, Heidelberg, August 2009. Springer-Verlag. [cited on p. 164]
- [185] Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1992. [cited on p. 99]
- [186] Qi He, Constantinos Dovrolis, and Mostafa Ammar. On the Predictability of Large Transfer TCP Throughput. *Computer Networks*, 51(14):3959–3977, October 2007. [cited on p. 28, 66]
- [187] Werner Hei, Michael Welzl (editors), Thomas Bocek, Dragana Damjanovic, Yehia El-khatib, Linghang Fan, Hasan Hasan, David Hausheer, Tao Liu, Cristian Morariu, Marcelo Pasin, Pascale Vicat-Blanc Primet, Peter Racz, Gregor Schaffrath, Burkhard Stiller, Jin Wu, Chen Zhang, Lin Zhang, Shaohua Liu, Xiaolei Ma, Li Li, Chongwu Chen, Kun Wang, and Wei Li. Survey of State of the Art. Technical report, University of Innsbruck, May 2007. Pages 9–18. Available from <http://www.ec-gin.eu/corpsite/manage/assets/d1.0-update.pdf>. [cited on p. 59]
- [188] Doug L. Hoffman, Amy Apon, Larry Dowdy, Baochuan Lu, Nathan Hamm, Linh Ngo, and Hung Bui. *Performance Modeling of Enterprise Grids*, volume 132 of *International Series in Operations Research & Management Science*, pages 169–201. Springer US, 2010. [cited on p. 122]
- [189] Charles C. Holt. Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages. Office of Naval Research (ONR) Research Memorandum 52, Carnegie Institute of Technology, Pittsburgh, PA, April 1957. [cited on p. 160]
- [190] Martin Horneffer. Assessing Internet Performance Metrics Using Large-Scale TCP-SYN based Measurements. In *Proceedings of the Passive and Active Measurement Workshop (PAM'00)*, April 2000. [cited on p. 22]
- [191] Ningning Hu and P. Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, August 2003. [cited on p. 24]
- [192] Richard Hughes-Jones. UDPmon: A Tool for Investigating Network Performance. <http://www.hep.man.ac.uk/~rich/net>. [cited on p. 23]
- [193] Bla Hullr, Sndor Laki, Jzsef Stger, Istvn Csabai, and Gbor Vattay. SONoMA: A Service Oriented Network Measurement Architecture. Technical Report TR-ELTE-

- CNL-2010/1, Department of Physics of Complex Systems, Eötvös Loránd University, Budapest, Hungary, May 2010. [cited on p. 53]
- [194] Geoff Huston. Analyzing the Internet's BGP Routing Table. *The Internet Protocol Journal*, 4(1), March 2001. [cited on p. 21]
- [195] Mouhamad Ibrahim, Eitan Altman, Pascale Primet, Giovanna Carofiglio, and Georg Post. A Simulation Study of Passive Inference of TCP Rate and Detection of Congestion. In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'09)*, pages 1–10, Brussels, Belgium, Belgium, October 2009. ICST. [cited on p. 22]
- [196] The Scripps Research Institute. AutoDock. <http://autodock.scripps.edu/>. [cited on p. 127]
- [197] The Scripps Research Institute. FightAIDS@Home. <http://fightaidsathome.scripps.edu/>. [cited on p. 127]
- [198] Alexandru Iosup and Dick Epema. GRENCHMARK: A Framework for Analyzing, Testing, and Comparing Grids. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, pages 313–320, Washington, DC, USA, May 2006. IEEE Computer Society. [cited on p. 126]
- [199] Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, and Dick H. J. Epema. The Grid Workloads Archive. *Future Generation Computer Systems*, 24(7):672–686, July 2008. [cited on p. 126]
- [200] John J. Irwin and Brian K. Shoichet. ZINC: A Free Database of Commercially Available Compounds for Virtual Screening. *Journal of Chemical Information and Modeling*, 45(1):177–182, January 2005. [cited on p. 127]
- [201] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. In *Proceedings of the Second ACM SIGOPS European Conference on Computer Systems (EuroSys'07)*, pages 59–72. ACM, March 2007. [cited on p. 60]
- [202] Van Jacobson. Pathchar. <http://www.caida.org/tools/utilities/others/pathchar>. [cited on p. 21, 23]
- [203] Van Jacobson. pathchar - A Tool to Infer Characteristics of Internet Paths. <ftp://ftp.ee.lbl.gov/pathchar/msri-talk.pdf>, April 1997. [cited on p. 14]
- [204] Van Jacobson, Steven McCanne, and Craig Leres. TCPDUMP/LIBPCAP Public Repository. <http://www.tcpdump.org/>. [cited on p. 26, 91]

- [205] Nicolas Jacq, Jean Salzemann, Florence Jacq, Yannick Legré, Emmanuel Medernach, Johan Montagnat, Astrid Maaß, Matthieu Reichstadt, Horst Schwichtenberg, Mahendrakar Sridhar, Vinod Kasam, Marc Zimmermann, Martin Hofmann, and Vincent Breton. Grid-enabled Virtual Screening Against Malaria. *Journal of Grid Computing*, 6(1):29–43, December 2008. [cited on p. 142]
- [206] Manish Jain and Constantinos Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Transactions on Networking*, 11(4):537–549, August 2003. [cited on p. 24]
- [207] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, and Donald F. Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *Proceedings of the Twenty Third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, pages 1582–1592, March 2004. [cited on p. 22, 25]
- [208] Gerard Paul Java. IPTraf - An IP Network Monitor. <http://iptraf.seul.org/>. [cited on p. 26]
- [209] Gwilym M. Jenkins and Donald G. Watts. *Spectral Analysis and Its Applications*. Series in Time Series Analysis. Holden-Day, 1968. [cited on p. 97]
- [210] Hao Jiang and Constantinos Dovrolis. Passive Estimation of TCP Round-Trip Times. *ACM SIGCOMM Computer Communication Review*, 32(3):75–88, July 2002. [cited on p. 22, 76]
- [211] Richard H. Jones. Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations. *Technometrics*, 22(3):389–395, August 1980. [cited on p. 99]
- [212] Rick Jones. NetPerf. <http://www.netperf.org/>. [cited on p. 23]
- [213] Roelof Jonkman. NetSpec: Philosophy, Design and Implementation. Master's thesis, University of Kansas, February 1998. [cited on p. 23]
- [214] Sunil Kalidindi and Matthew J. Zekauskas. Surveyor: An Infrastructure for Internet Performance Measurements. In *Proceedings of the Internet Global Summit (INET'99)*, June 1999. [cited on p. 20]
- [215] Rudolf E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME: Journal of Basic Engineering*, 82(D):35–45, March 1960. [cited on p. 99]
- [216] Rohit Kapoor, Ling-Jyh Chen, Li Lao, Mario Gerla, and M. Y. Sanadidi. CapProbe: A Simple and Accurate Capacity Estimation Technique. *ACM SIGCOMM Computer Communication Review*, 34(4):67–78, October 2004. [cited on p. 24]
- [217] Elliott Karpilovsky, Alexandre Gerber, Dan Pei, Jennifer Rexford, and Aman Shaikh. Quantifying the Extent of IPv6 Deployment. In *Proceedings of the Tenth International*

- Conference on Passive and Active Network Measurement (PAM'09)*, pages 13–22, Berlin, Heidelberg, 2009. Springer-Verlag. [cited on p. 28]
- [218] Vinod Kasam, Jean Salzemann, Nicolas Jacq, Astrid Maaß, and Vincent Breton. Large Scale Deployment of Molecular Docking Application on Computational Grid infrastructures for Combating Malaria. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID'07)*, pages 691–700, Washington, DC, USA, May 2007. IEEE Computer Society. [cited on p. 142]
- [219] Dina Katabi and Charles Blake. Inferring Congestion Sharing and Path Characteristics for Packet Interarrival Times. Technical Report MIT TR-828, Massachusetts Institute of Technology, Cambridge, MA, USA, December 2001. [cited on p. 22]
- [220] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'02)*, pages 89–102, New York, NY, USA, October 2002. ACM. [cited on p. 17]
- [221] Sachin Katti, Dina Katabi, Charles Blake, Eddie Kohler, and Jacob Strauss. MultiQ: Automated Detection of Multiple Bottleneck Capacities Along a Path. In *Proceedings of the Fourth ACM SIGCOMM Conference on Internet Measurement (IMC'04)*, pages 245–250, New York, NY, USA, October 2004. ACM. [cited on p. 25]
- [222] Srinivasan Keshav. *Congestion Control in Computer Networks*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1992. Available at <http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/91/thesis/keshav.th.tar.Z>. [cited on p. 23]
- [223] Srinivasan Keshav and Rosen Sharma. Issues and Trends in Router Design. *IEEE Communications Magazine*, 36(5):144–151, May 1998. [cited on p. 63]
- [224] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998. [cited on p. 202]
- [225] Ken Keys, David Moore, Ryan Koga, Edouard Lagache, Michael Tesch, and k claffy. The Architecture of CoralReef: An Internet Traffic Monitoring Software Suite. In *Proceedings of the Second Passive and Active Measurement Workshop (PAM'01)*, April 2001. [cited on p. 27]
- [226] Dmitri Kondrashov and Michael Ghil. Spatio-Temporal Filling of Missing Points in Geophysical Data Sets. *Nonlinear Processes in Geophysics*, 13(2):151–159, May 2006. [cited on p. 98]
- [227] Balázs Kónya. The NorduGrid/ARC Information System. Technical report, NorduGrid, May 2010. [cited on p. 42]

- [228] Balazs Konya, Laurence Field, and Sergio Andreozzi. GLUE Working Group (GLUE). <http://forge.ogf.org/sf/projects/glue-wg>. [cited on p. 42]
- [229] Klaus Krauter, Rajkumar Buyya, and Muthucumar Maheswaran. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. *Software: Practice and Experience*, 32(2):135–164, February 2002. [cited on p. 54]
- [230] Elmar Krieger and Gert Vriend. Models@Home: Distributed Computing in Bioinformatics Using a Screensaver Based Approach. *Bioinformatics*, 18(2):315–318, February 2002. [cited on p. 200]
- [231] Kevin Lai and Mary Baker. Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In *Proceedings of the Third USENIX Symposium on Internet Technologies and Systems (USITS'01)*, Berkeley, CA, USA, March 2001. USENIX Association. [cited on p. 24]
- [232] T. V. Lakshman, Bernhard Suter, and U. Madhow. Window-Based Error Recovery and Flow Control with a Slow Acknowledgement Channel: A Study of TCP/IP Performance. In *Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'97)*, volume 3, pages 1199–1209, Washington, DC, USA, April 1997. IEEE Computer Society. [cited on p. 17]
- [233] Ryan Lance and Ian Frommer. Round-Trip Time Inference via Passive Monitoring. *ACM SIGMETRICS Performance Evaluation Review*, 33(3):32–38, December 2005. [cited on p. 22]
- [234] Erwin Laure, Steve Fisher, kos Frohner, Claudio Grandi, Peter Kunszt, Ales Krenek, Olle Mulmo, Fabrizio Pacini, Francesco Prelz, John White, Maite Barroso, Predrag Buncic, Frédéric Hemmer, Alberto Di Meglio, and Łke Edlund. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45, 2006. [cited on p. 3, 206]
- [235] Craig A. Lee, James Stepanek, Rich Wolski, Carl Kesselman, and Ian Foster. A Network Performance Tool for Grid Environments. In *Proceedings of the 1999 ACM/IEEE Conference on Supercomputing (SC'99)*, page 4, New York, NY, USA, November 1999. ACM. [cited on p. 46]
- [236] Jason Lee, Dan Gunter, Martin Stoufer, and Brian Tierney. Monitoring Data Archives for Grid Environments. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing (SC'02)*, pages 1–10, Los Alamitos, CA, USA, November 2002. IEEE Computer Society Press. [cited on p. 48]
- [237] Seungjun Lee. eTOP: End-to-end Performance Measurement Infrastructure on KOREN and APAN Links. Available at <http://master.apan.net/meetings/busan03/measurement/koren.ppt> and <http://etop.apan.net>, 2003. [cited on p. 50]
- [238] Mark Leese. End-to-End Performance: Issues and Suggestions. In *TERENA Fifth NRENs and Grids Workshop*, June 2007. [cited on p. 66]

- [239] Maozhen Li and Mark Baker. *The Grid Core Technologies*. John Wiley & Sons, 2005. [cited on p. 54]
- [240] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the Eighth International Conference on Distributed Computing Systems (ICDCS'88)*, June 1988. [cited on p. 37]
- [241] Karawynn Long, Malcolm Nixon, Scott Le Grand, and Stephanie Wukovitz. folderol - An open-sourced distributed computing client for exploring the results of the Human Genome Project. <http://sourceforge.net/projects/folderol/>. [cited on p. 200]
- [242] Chris Lonvick. *RFC 3164: The BSD syslog Protocol*. Internet Engineering Task Force, August 2001. [cited on p. 129]
- [243] Bruce B. Lowekamp. Combining Active and Passive Network Measurements to Build Scalable Monitoring Systems on the Grid. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):19–26, March 2003. [cited on p. 25, 49]
- [244] Xiapu Luo, Edmond W. W. Chan, and Rocky K. C. Chang. Design and Implementation of TCP Data Probes for Reliable and Metric-Rich Network Path Monitoring. In *Proceedings of the 2009 USENIX Annual Technical Conference (USENIX'09)*, pages 4–17, Berkeley, CA, USA, June 2009. USENIX Association. [cited on p. 22]
- [245] Tiejun Ma, Yehia El-khatib, Michael Mackay, and Chris Edwards. Characterising A Grid Site's Traffic. In *Proceedings of Nineteenth ACM International Symposium on High Performance Distributed Computing (HPDC-19)*, pages 707–716, New York, NY, USA, June 2010. ACM. [cited on p. 59]
- [246] Jason Maassen and Henri E. Bal. Smartsockets: Solving the Connectivity Problems in Grid Computing. In *Proceedings of the Sixteenth international Symposium on High Performance Distributed Computing (HPDC-16)*, pages 1–10, New York, NY, USA, June 2007. ACM. [cited on p. 87]
- [247] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proceedings of the Seventh USENIX Symposium on Operating Systems Design and Implementation (OSDI'06)*, pages 367–380, Berkeley, CA, USA, November 2006. USENIX Association. [cited on p. 54]
- [248] Bruce A. Mah. pchar: A Tool for Measuring Internet Path Characteristics. <http://www.kitchenlab.org/www/bmah/Software/pchar>. [cited on p. 23]
- [249] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas Anderson. User-level Internet Path Diagnosis. *ACM SIGOPS Operating Systems Review*, 37(5):106–119, December 2003. [cited on p. 22]

- [250] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy H. Katz. Towards an Accurate AS-level traceroute Tool. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'03)*, pages 365–378, New York, NY, USA, August 2003. ACM. [cited on p. 21]
- [251] Vladislav Marinov and Jürgen Schönwälder. Performance Analysis of SNMP over SSH. In *Large Scale Management of Distributed Systems, Proceedings of the Seventeenth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'06)*, pages 25–36, October 2006. [cited on p. 30]
- [252] Angelos K. Marnerides, Dimitrios P. Pazaros, and David Hutchison. Detection and Mitigation of Abnormal Traffic Behavior in Autonomic Networked Environments. In *Student Workshop of the Fourth ACM SIGCOMM Conference on emerging Networking EXperiments and Technologies (CoNEXT'08)*, December 2008. [cited on p. 160]
- [253] H. Stele Martin, Anthony McGregor, and John G. Cleary. Analysis of Internet Delay Times. In *Proceedings of the Passive and Active Measurement Workshop (PAM'00)*, pages 33–40, April 2000. [cited on p. 22, 78]
- [254] Tim Mather, Subra Kumaraswamy, and Shahed Latif. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Media, 2009. [cited on p. 163]
- [255] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, July 1997. [cited on p. 53]
- [256] Warren Matthews and Les Cottrell. The PingER Project: Active Internet Performance Monitoring for the HENP Community. *Communications Magazine, IEEE*, 38(5):130–136, May 2000. [cited on p. 20]
- [257] Richard McClatchey, Ashiq Anjum, Heinz Stockinger, Arshad Ali, Ian Willers, and Michael Thomas. Data Intensive and Network Aware (DIANA) Grid Scheduling. *Journal of Grid Computing*, 5(1):43–64, January 2007. [cited on p. 53, 65]
- [258] Tony McGregor, Hans-Werner Braun, and Jeff Brown. The NLANR Network Analysis Infrastructure. *Communications Magazine, IEEE*, 38(5):122–128, May 2000. [cited on p. 48]
- [259] Larry McVoy and Carl Staelin. Imbench: Portable Tools for Performance Analysis. In *Proceedings of the USENIX Annual Technical Conference (ATC'96)*, pages 23–23, Berkeley, CA, USA, January 1996. USENIX Association. [cited on p. 23]
- [260] E. Meijering. A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing. *Proceedings of the IEEE*, 90(3):319–342, March 2002. [cited on p. 98]
- [261] Bob Melander, Mats Bjorkman, and Per Gunningberg. A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks. In *Proceedings of the IEEE*

- Global Telecommunications Conference (GLOBECOM'00)*, pages 415–420, November 2000. [cited on p. 24]
- [262] Christopher Metz. IP Routers: New Tool for Gigabit Networking. *Internet Computing, IEEE*, 2(6):14–18, November 1998. [cited on p. 63]
- [263] David Moore, Ram Periakaruppan, Jim Donohoe, and k claffy. Where in the World is netgeo.caida.org? In *Proceedings of the Tenth Annual Internet Society Conference (INET'00)*, July 2000. [cited on p. 21]
- [264] Biswanath Mukherjee, Dhritiman Banerjee, S. Ramamurthy, and Amarnath Mukherjee. Some Principles for Designing a Wide-Area WDM Optical Network. *IEEE/ACM Transactions on Networking*, 4(5):684–696, October 1996. [cited on p. 63]
- [265] Sumit Naiksatam, Silvia Figueira, Stephen A. Chiappari, and Nirdosh Bhatnagar. Analyzing the Advance Reservation of Lightpaths in Lambda-Grids. In *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)*, volume 2, pages 985–992, May 2005. [cited on p. 64]
- [266] Guido Negri. Put Your Hands on gLite. <http://lxmi.mi.infn.it/~gnegri/docs/handsOn.ppt>. [cited on p. 206]
- [267] Jeremy Nowell, Kostas Kavoussanakis, Charaka Palansuriya, Michal Piotrowski, Florian Scharinger, Paul Graham, Bartosz Dobrzelecki, and Arthur Trew. Standards-based Network Monitoring for the Grid. *Philosophical Transactions of the Royal Society A*, 367(1897):2495–2505, June 2009. [cited on p. 66]
- [268] Tobias Oetiker. MRTG - The Multi Router Traffic Grapher. In *Proceedings of the Twelfth USENIX Systems Administration Conference (LISA'98)*, pages 141–148, Berkeley, CA, USA, December 1998. USENIX Association. [cited on p. 27]
- [269] Angela Orebaugh, Gilbert Ramirez, Josh Burke, and Larry Pesce. *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Syngress Publishing, 2006. [cited on p. 26]
- [270] Shawn Ostermann. tcptrace. <http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html>. [cited on p. 26]
- [271] Francesco Palmieri. Network-Aware Scheduling for Real-Time Execution Support in Data-Intensive Optical Grids. *Future Generation Computer Systems*, 25(7):794–803, July 2009. [cited on p. 64]
- [272] Craig Partridge, Philip P. Carvey, Ed Burgess, Isidro Castineyra, Tom Clarke, Lise Graham, Michael Hathaway, Phil Herman, Allen King, Steve Kohalmi, Tracy Ma, John Mcallen, Trevor Mendez, Walter C. Milliken, Ronald Pettyjohn, John Rokosz, Joshua Seeger, Michael Sollins, Steve Storch, Benjamin Tober, and Gregory D. Troxel. A 50-Gb/s IP Router. *IEEE/ACM Transactions on Networking*, 6(3):237–248, June 1998. [cited on p. 63]

- [273] Vern Paxson. End-to-End Routing Behavior in the Internet. *ACM SIGCOMM Computer Communication Review*, 36(5):41–56, October 2006. [cited on p. 86]
- [274] Vern Paxson and Sally Floyd. Why We Don't Know How to Simulate the Internet. In *Proceedings of the Twenty-Ninth Conference on Winter Simulation (WSC'97)*, pages 1037–1044, Washington, DC, USA, December 1997. IEEE Computer Society. [cited on p. 122]
- [275] Vern Edward Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, Berkeley, CA, USA, 1998. [cited on p. 65]
- [276] Ram Periakaruppan and Evi Nemeth. GTrace - A Graphical Traceroute Tool. In *Proceedings of the Thirteenth USENIX Systems Administration Conference (LISA'99)*, pages 69–78, Berkeley, CA, USA, November 1999. USENIX Association. [cited on p. 21]
- [277] Antonio Delgado Peris, Patricia Méndez Lorenzo, Flavia Donno, Andrea Sciabà, Simone Campana, and Roberto Santinelli. *LCG-2 User Guide*. LHC Computing Grid, CERN, 2.3 edition, August 2005. <https://edms.cern.ch/file/454439//LCG-2-UserGuide.html>. [cited on p. 206]
- [278] Dimitrios Pezaros, Manolis Sifalakis, Stefan Schmid, and David Hutchison. Dynamic Link Measurements using Active Components. In *Proceedings of the Sixth IFIP International Working Conference on Active Networks (IWAN'04)*, Lecture Notes in Computer Science. Springer Verlag, Heidelberg, October 2004. [cited on p. 28, 66]
- [279] Peter Phaal, Sonia Panchen, and Neil McKee. *RFC 3176: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*. Internet Engineering Task Force, September 2001. [cited on p. 27]
- [280] Rob Pike, Sean Dorward, Robert Griesemer, and Sean Quinlan. Interpreting the Data: Parallel analysis with Sawzall. *Scientific Programming - Dynamic Grids and Worldwide Computing*, 13:277–298, October 2005. [cited on p. 60]
- [281] Norbert Podhorszki, Zoltán Balaton, and Gabor Gombás. Monitoring Message-Passing Parallel Applications in the Grid with GRM and Mercury Monitor. In *Proceedings of the Second European Across Grids Conference (AxGrids '04)*, volume 3165 of *Lecture Notes in Computer Science*, pages 179–181. Springer, January 2004. [cited on p. 54]
- [282] Ravi Prasad, Constantinos Dovrolis, Margaret Murray, and kc claffy. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. *IEEE Network*, 17(6):27–35, November 2003. [cited on p. 24]
- [283] Ravi S. Prasad, Constantinos Dovrolis, and Bruce A. Mah. The Effect of Layer-2 Switches on Pathchar-like Tools. In *Proceedings of the Second ACM SIGCOMM Workshop on Internet Measurement (IMW'02)*, pages 321–322, New York, NY, USA, November 2002. ACM. [cited on p. 29]

- [284] Chunming Qiao and Myungsik Yoo. Optical burst switching (OBS) - a new paradigm for an optical Internet. *Journal of High Speed Networks, Special issue on optical networking*, 8(1):69–84, March 1999. [cited on p. 63]
- [285] Juergen Quittek, Stewart Bryant, Benoit Claise, Paul Aitken, and Jeff Meyer. *RFC 5102: Information Model for IP Flow Information Export*. Internet Engineering Task Force, January 2008. [cited on p. 27]
- [286] R Development Core Team. *R: A Language and Environment for Statistical Computing*, 2009. ISBN 3-900051-07-0. [cited on p. 26, 99]
- [287] Ioan Raicu, Zhao Zhang, Mike Wilde, Ian Foster, Pete Beckman, Kamil Iskra, and Ben Clifford. Toward Loosely Coupled Programming on Petascale Systems. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (SC'08)*, pages 1–12, Piscataway, NJ, USA, November 2008. IEEE Press. [cited on p. 57]
- [288] Rajiv Ramaswami and Kumar N. Sivarajan. Design of Logical Topologies for Wavelength-Routed All-Optical Networks. In *Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM'95)*, volume 3, page 1316, Washington, DC, USA, April 1995. IEEE Computer Society. [cited on p. 63]
- [289] Sushant Rewaskar, Jasleen Kaur, and F. Donelson Smith. A Passive State-Machine Approach for Accurate Analysis of TCP Out-of-Sequence Segments. *ACM SIGCOMM Computer Communication Review*, 36(3):51–64, July 2006. [cited on p. 25]
- [290] Sushant Rewaskar, Jasleen Kaur, and F. Donelson Smith. Accuracy of Probing Techniques in Estimating TCP Loss Rates, September 2006. [cited on p. 24, 29]
- [291] Vinay J. Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient Available Bandwidth Estimation for Network Paths. In *Proceedings of the Fourth Passive and Active Measurement Workshop (PAM'03)*, April 2003. [cited on p. 24]
- [292] Eric C. Rosen, Arun Viswanathan, and Ross Callon. *RFC 3031: Multiprotocol Label Switching Architecture*. Internet Engineering Task Force, January 2001. [cited on p. 63]
- [293] Alain J. Roy. *End-to-End Quality of Service for High-End Applications*. PhD thesis, The University of Chicago, August 2001. [cited on p. 64]
- [294] Federico D. Sacerdoti, Mason J. Katz, Matthew L. Massie, and David E. Culler. Wide Area Cluster Monitoring with Ganglia. In *Proceedings of the Fifth IEEE International Conference on Cluster Computing (CLUSTER'03)*, pages 289–298, Los Alamitos, CA, USA, December 2003. IEEE Computer Society. [cited on p. 54, 160]

- [295] Stefan Sariou, P. Krishna Gummadi, and Steven D. Gribble. Sprobe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments. In *Proceedings of the Twenty First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, June 2002. [cited on p. 24]
- [296] Stefan Savage. Sting: a TCP-based Network Measurement Tool. In *Proceedings of the Second USENIX Symposium on Internet Technologies and Systems (USITS'99)*, volume 2, pages 7–15, Berkeley, CA, USA, October 1999. USENIX Association. [cited on p. 22]
- [297] Angel Schneider (editor). Cloud Hosting Awareness Survey: Key Findings. <http://www.rackspace.com/downloads/surveys/CloudAwarenessSurvey.pdf>, January 2009. Rackspace Hosting, San Antonio, TX 78229, USA. [cited on p. 163]
- [298] Rody Schoonderwoerd. Network Performance Measurement Tools: A Comprehensive Comparison. Master's thesis, Vrije Universiteit Amsterdam, November 2002. Available at <http://www.cs.vu.nl/~kielmann/papers/rody-thesis.pdf>. [cited on p. 27]
- [299] Jennifer M. Schopf. Ten Actions When Grid Scheduling: The User as a Grid Scheduler. In Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz, editors, *Grid Resource Management: State of the Art and Future Trends*, pages 15–23. Kluwer Academic Publishers, Norwell, MA, USA, 2004. [cited on p. 7]
- [300] Jennifer M. Schopf and Ben Clifford. Monitoring Clusters and Grids. *ClusterWorld magazine*, 2(7):2–6, August 2004. [cited on p. 54]
- [301] Michael Schulz and Karl Stattegger. SPECTRUM: Spectral Analysis of Unevenly Spaced Paleoclimatic Time Series. *Computers & Geosciences*, 23(9):929–945, 1997. [cited on p. 160]
- [302] National Grid Service. The NGS in Numbers. <http://ngs.ac.uk/sites/default/files/file/NGSposters/NGSinnnumbers.pdf>, November 2009. [cited on p. 204]
- [303] Srinivasan Seshan, Mark Stemm, and Randy H. Katz. SPAND: Shared Passive Network Performance Discovery. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS'97)*, pages 135–146, Berkeley, CA, USA, December 1997. USENIX Association. [cited on p. 66]
- [304] Yuval Shavitt, Xiaodong Sun, Avishai Wool, and Bülent Yener. Computing the Unmeasured: An Algebraic Approach to Internet Mapping. *IEEE Journal on Selected Areas in Communications*, 22(1):67–78, April 2004. [cited on p. 50]
- [305] Rob Sherwood and Neil Spring. Touring the Internet in a TCP Sidecar. In *Proceedings of the Sixth ACM SIGCOMM Conference on Internet Measurement (IMC'06)*, pages 339–344, New York, NY, USA, October 2006. ACM. [cited on p. 22, 25]
- [306] Brian K. Shoichet. Virtual Screening of Chemical Libraries. *Nature*, 432:862–865, December 2004. [cited on p. 128]

- [307] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and kc claffy. Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM'05)*, pages 306–320. Springer, March 2005. [cited on p. 24]
- [308] Dimitra Simeonidou, Reza Nejabati, and Mike J. O'Mahony. An Optical Network Infrastructure Suitable for Global Grid Computing. In *Proceedings of the TERENA Networking Conference (TNC'04)*, June 2004. [cited on p. 63]
- [309] Warren Smith. A System for Monitoring and Management of Computational Grids. In *Proceedings of the 2002 International Conference on Parallel Processing (ICPP'02)*, page 55, Washington, DC, USA, August 2002. IEEE Computer Society. [cited on p. 54]
- [310] Marc Snir, Steve W. Otto, David W. Walker, Jack Dongarra, and Steven Huss-Lederman. *MPI: The Complete Reference*. MIT Press, Cambridge, MA, USA, 1995. [cited on p. 57]
- [311] Nir Sofer. SocketSniff - Windows Sockets (WinSock) Sniffer. http://www.nirsoft.net/utils/socket_sniffer.html. [cited on p. 26]
- [312] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. A Geometric Approach to Improving Active Packet Loss Measurement. *IEEE/ACM Transactions on Networking*, 16(2):307–320, April 2008. [cited on p. 20, 25]
- [313] Borja Sotomayor and Lisa Childers. *Globus Toolkit 4: Building Java Services*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2006. [cited on p. 101]
- [314] William Stallings. SNMPv3: A Security Enhancement to SNMP. *IEEE Communications Surveys and Tutorials*, 1(1), March 1998. [cited on p. 30]
- [315] Thomas Lawrence Sterling. *Beowulf Cluster Computing with Linux*. MIT Press, Cambridge, MA, USA, 2002. [cited on p. 37]
- [316] W. Richard Stevens. *RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*. Internet Engineering Task Force, January 1997. [cited on p. 63]
- [317] Ellen J. Stokes, Sergio Andreatto, Michel Drescher, and Andreas Savva. Information & Data Modeling in OGSA Grids. Technical report, Open Grid Forum, March 2008. [cited on p. 42]
- [318] Ellen J. Stokes and Lawrence Flon. Job Submission Information Model (JSIM). Technical report, Global Grid Forum, Month 2004. [cited on p. 42]
- [319] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. In *Proceedings of the Third ACM SIGCOMM Conference on Internet Measurement (IMC'03)*, pages 39–44, New York, NY, USA, October 2003. ACM. [cited on p. 24]

- [320] Jaspal Subhlok, Peter Lieu, and Bruce Lowekamp. Automatic Node Selection for High Performance Applications on Networks. *ACM SIGPLAN Notices*, 34(8):163–172, August 1999. [cited on p. 45]
- [321] Mark Sullivan and Andrew Heybey. Tribeca: A system for managing large databases of network traffic. In *Proceedings of the USENIX Annual Technical Conference (ATEC'98)*, pages 13–24, Berkeley, CA, USA, June 1998. USENIX Association. [cited on p. 54]
- [322] Xian-He Sun and Ming Wu. GHS: A Performance System of Grid Computing. In *Proceedings of the Nineteenth International Parallel and Distributed Processing Symposium (IPDPS'05)*, volume 11, page 228a, Los Alamitos, CA, USA, April 2005. IEEE Computer Society. [cited on p. 54]
- [323] Mike SurrIDGE and Colin Upstill. Grid Security: Lessons for Peer-to-Peer Systems. In *Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03)*, pages 2–6, September 2003. [cited on p. 87]
- [324] Martin Swany, Jason W. Zurawski, and Dan Gunter. *NMWG Schema Developers Guide*. OGE, February 2009. <http://anonsvn.internet2.edu/svn/nmwg/trunk/nmwg/doc/devguide/devguide.pdf>. [cited on p. 42]
- [325] François Taïani, Matti Hiltunen, and Rick Schlichting. The Impact of Web Service Integration on Grid Performance. In *Proceedings of the Fourteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-14)*, pages 14–23, Washington, DC, USA, July 2005. IEEE Computer Society. [cited on p. 61, 103]
- [326] Jefferson Tan, David Abramson, and Colin Enticott. REMUS: A Rerouting and Multiplexing System for Grid Connectivity Across Firewalls. *Journal of Grid Computing*, 7(1):25–50, 2009. [cited on p. 87]
- [327] Andrew S. Tanenbaum and Robbert Van Renesse. Distributed Operating Systems. *ACM Computing Surveys*, 17(4):419–470, December 1985. [cited on p. 37]
- [328] Andrew S. Tanenbaum, Robbert van Renesse, Hans van Staveren, Gregory J. Sharp, and Sape J. Mullender. Experiences with the Amoeba Distributed Operating System. *Communications of the ACM*, 33(12):46–63, December 1990. [cited on p. 37]
- [329] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall International, second revised edition, October 2006. [cited on p. 37]
- [330] Osamu Tatebe, Youhei Morita, Satoshi Matsuoka, Noriyuki Soda, and Satoshi Sekiguchi. Grid Datafarm Architecture for Petascale Data Intensive Computing. In *Proceedings of the Second IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, pages 102–110, Washington, DC, USA, June 2002. IEEE Computer Society. [cited on p. 95]

- [331] Ian J. Taylor and Andrew Harrison. *From P2P and Grids to Services on the Web: Evolving Distributed Communities*. Springer Publishing Company, 2009. [cited on p. 198]
- [332] Henri Theil. *Applied Economic Forecasting*. North-Holland, 1966. [cited on p. 161]
- [333] Brian Tierney, Ruth Aydt, Dan Gunter, Warren Smith, Martin Swany, Valerie Taylor, and Rich Wolski. A Grid Monitoring Architecture. Technical report, Global Grid Forum, January 2002. [cited on p. 41]
- [334] Brian Tierney, Brian Crowley, Dan Gunter, Jason Lee, and Mary Thompson. A Monitoring Sensor Management System for Grid Environments. *Cluster Computing*, 4(1):19–28, March 2001. [cited on p. 54]
- [335] Brian Tierney, William E. Johnston, Brian Crowley, Gary Hoo, Christopher Brooks, and Dan Gunter. The NetLogger Methodology for High Performance Distributed Systems Performance Analysis. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC-7)*, pages 260–267, July 1998. [cited on p. 129]
- [336] Brian Tierney, Martin Swany, and Je Boote. perfSONAR: Instantiating a Global Network Measurement Framework. *ACM SIGOPS Operating Systems Review*, 43(4):50–53, January 2010. [cited on p. 51]
- [337] Brian L. Tierney, Jason Lee, Brian Crowley, Mason Holding, Jeremy Hylton, and Jr. Drake, Fred L. A Network-Aware Distributed Storage Cache for Data Intensive Environments. In *Proceedings of the Eighth International Symposium on High Performance Distributed Computing (HPDC-8)*, pages 185–193, August 1999. [cited on p. 63]
- [338] Ajay Tirumala, Mark Gates, Feng Qin, Jon Dugan, and Jim Ferguson. Iperf - The TCP/UDP Bandwidth Measurement Tool. <http://dast.nlanr.net/Projects/Iperf>. [cited on p. 23, 108]
- [339] Michael C. Toren. tcptraceroute. <http://michael.toren.net/code/tcptraceroute/>. [cited on p. 21]
- [340] Franco Travostino, Joe Mambretti, and Gigi Karmous-Edwards, editors. *Grid Networks: Enabling Grids with Advanced Communication Technology*. John Wiley & Sons, 2006. [cited on p. 59]
- [341] George Tsouloupas and Marios D. Dikaiakos. GridBench: A Tool for Benchmarking Grids. *IEEE/ACM International Workshop on Grid Computing*, page 60, November 2003. [cited on p. 122]
- [342] Steve Tuecke. Re: gloperf. http://www-unix.globus.org/mail_archive/discuss/2001/Archive/msg00497.html. [cited on p. 47]

- [343] Steve Tuecke. Globus: Enhancements in Performance, Manageability, and the User Experience. Proceedings of OGF29, June 2010. <http://www.ogf.org/OGF29/materials/2058/Globus+OGF29.pdf>. [cited on p. 44, 68]
- [344] Steven Tuecke, Karl Czajkowski, Ian Foster, Jeff Frey, Steve Graham, Carl Kesselman, David Snelling, and Peter Vanderbilt. *Open Grid Services Infrastructure (OGSI), Version 1.0*, 2003. [cited on p. 4, 202]
- [345] Andy Turner. Geodemographic Modelling. <http://www.ngs.ac.uk/geodemographic-modelling>. [cited on p. 142]
- [346] Jonathan S. Turner. Terabit Burst Switching. *Journal of High Speed Networks, Special Issue on Optical Networking*, 8(1):3–16, March 1999. [cited on p. 63]
- [347] Gareth Tyson. *A Middleware Approach to Building Content-Centric Applications*. PhD thesis, Lancaster University, May 2010. [cited on p. 159]
- [348] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, January 2009. [cited on p. 58, 163]
- [349] Yehuda Vardi. Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data. *Journal of the American Statistical Association*, 91(443), March 1996. [cited on p. 29]
- [350] Sudharshan Vazhkudai, Jennifer M. Schopf, and Ian T. Foster. Predicting the Performance of Wide Area Data Transfers. In *Proceedings of the Sixteenth International Parallel and Distributed Processing Symposium (IPDPS'02)*, page 270, Washington, DC, USA, April 2002. IEEE Computer Society. [cited on p. 48, 65, 66]
- [351] Bryan Veal, Kang Li, and David K. Lowenthal. New Methods for Passive Estimation of TCP Round-Trip Times. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM'05)*, pages 121–134. Springer, March 2005. [cited on p. 25]
- [352] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing. *ACM Computing Surveys*, 38(1):3, March 2006. [cited on p. 57]
- [353] Paulo Veríssimo and Luís Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, Norwell, MA, USA, 2001. [cited on p. 5]
- [354] Grace Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990. [cited on p. 99]
- [355] Eric Weigle and Wu-chun Feng. Dynamic Right-Sizing: A Simulation Study. In *Proceedings of the Tenth International Conference on Computer Communication and Networking (ICCCN'01)*, October 2001. [cited on p. 17]

- [356] Li Wenwei, Zhang Dafang, Yang Jinmin, and Xie Gaogang. On Evaluating the Differences of TCP and ICMP in Network Measurement. *Computer Communications*, 30(2):428–439, January 2007. [cited on p. 18, 22, 65]
- [357] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02)*, pages 255–270, Boston, MA, December 2002. USENIX Association. [cited on p. 120]
- [358] Peter R. Winters. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, 6(3):324–342, 1960. [cited on p. 160]
- [359] Rich Wolski. Dynamically Forecasting Network Performance Using the Network Weather Service. *Cluster Computing*, 1(1):119–132, 1998. [cited on p. 47]
- [360] Xia Xie, Hai Jin, Jin Huang, and Qin Zhang. Grid Information Service Based on Network Latency. In *Proceedings of the First International Conference on Scalable Information Systems (InfoScale'06)*, number 9, New York, NY, USA, May 2006. ACM. [cited on p. 52]
- [361] Ramin Yahyapour and Philipp Wieder (editors). Grid Scheduling Use Cases. Technical Report GFD-I.064, Open Grid Forum, March 2006. [cited on p. 42]
- [362] Asim YarKhan and Jack J. Dongarra. Experiments with Scheduling Using Simulated Annealing in a Grid Environment. In *Proceedings of the Third International Workshop on Grid Computing (GRID'02)*, volume 2536 of *Lecture Notes in Computer Science (LNCS)*, pages 232–242. Springer, November 2002. [cited on p. 71]
- [363] Muhammad Murtaza Yousaf and Michael Welzl. A Reliable Network Measurement and Prediction Architecture for Grid Scheduling. In *Proceedings of the IEEE/IFIP International Workshop on Autonomic Grid Networking and Management (AGNM'05)*, pages 1–10, Barcelona, Spain, October 2005. Universitat Politècnica de Catalunya. [cited on p. 52]
- [364] Muhammad Murtaza Yousaf, Michael Welzl, and Malik Muhammad Junaid. Fog in the Network Weather Service: A Case for Novel Approaches. In *Proceedings of the First International Conference on Networks for Grid Applications (GridNets'07)*, pages 1–6, Brussels, Belgium, Belgium, October 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). [cited on p. 66]
- [365] Marcia Zangrilli and Bruce B. Lowekamp. Using Passive Traces of Application Traffic in a Network Monitoring System. In *Proceedings of the Thirteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-13)*, pages 77–86. IEEE, June 2004. [cited on p. 22, 25, 53]

- [366] Marcia Zangrilli and Bruce B. Lowekamp. Applying Principles of Active Available Bandwidth Algorithms to Passive TCP Traces. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM'05)*, pages 333–336. Springer, March 2005. [cited on p. 22, 25, 53]
- [367] Marcia Zangrilli and Bruce B. Lowekamp. Transparent Optimization of Grid Server Selection With Real-Time Passive Network Measurements. In *Proceedings of the Third International Conference on Broadband Communications, Networks, and Systems (BROADNETS'06)*. IEEE, October 2006. [cited on p. 53]
- [368] Serafeim Zanicolas and Rizos Sakellariou. A Taxonomy of Grid Monitoring Systems. *Future Generation Computer Systems*, 21(1):163–188, January 2005. [cited on p. 54]
- [369] Jeremy D. Zawodny and Derek J. Balling. *High Performance MySQL: Optimization, Backups, Replication, Load-balancing, and More*. O'Reilly & Associates, Sebastopol, CA, USA, 2004. [cited on p. 94]
- [370] Xianan Zhang, Flavio Junqueira, Matti A. Hiltunen, Keith Marzullo, and Richard D. Schlichting. Replicating Nondeterministic Services on Grid Environments. In *Proceedings of the Fifteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-15)*, pages 105–116, June 2006. [cited on p. 61]
- [371] Xiao-Tao Zhang, Wei Zhang, Xiong Xiong, Qi-Wen Wang, and Cui-Yu Li. A Model-based Clustering for Time-series with Irregular Interval. In *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, volume 5, pages 2883–2888, August 2004. [cited on p. 98]
- [372] Xuehai Zhang, Jeffrey L. Freschl, and Jennifer M. Schopf. Scalability Analysis of Three Monitoring and Information Systems: MDS2, R-GMA, and Hawkeye. *Journal of Parallel and Distributed Computing*, 67(8):883–902, August 2007. [cited on p. 50, 67, 68]
- [373] Han Zheng, Eng Keong Lua, Marcelo Pias, and Timothy G. Griffin. Internet Routing Policies and Round-Trip-Times. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM'05)*, pages 236–250. Springer, March 2005. [cited on p. 86]

Appendices

Rise of the Grid

This appendix outlines the historical background of the grid computing paradigm. The aim here is not to provide a detailed chronicle of events that lead to the grid as there exists an ample number of publications that do that (c.f. [50, 121, 81, 331]). Instead, the aim is to help in understanding the different viewpoints when it comes to defining grid applications by studying the development of grid-enabling technologies.

A.1 Beginings

The emergence of distributed systems was mainly brought by the introduction of the VLSI technology. This technological breakthrough in chip design meant that it was feasible to exceed the performance a powerful processor delivers by replacing it with a number of slower (and cheaper) processors. This was the cue for a myriad of architectures where multiple processors worked together under the umbrella of one system. The second technological advance that shaped future direction of distributed systems was the introduction of advanced cross-platform communication technologies. This meant that computers could communicate with one another as long as they spoke the same language, i.e. communication protocol.

Initially, these technological advances promoted different uses that remained detached for quite a while. The former advance instigated investment in designing high-performance systems that could undertake complex problems. The latter advance, on the other hand, inspired collaboration on the organisational and individual levels.

Eventually, both technologies developed to reach greater heights in terms of what they offered. They also became much more affordable which aided in them becoming largely ubiquitous. Mainframes and minicomputers eventually became extinct and were replaced by the more abundant and cost-effective computational power provided by microcomputers. Conjointly, supercomputers became a common asset of many businesses, universities and government organisations in the developed world. LAN and WAN technologies were simultaneously developing.

Nevertheless, this progress did not suppress the need for even higher performance computing or communication. On the contrary, it enticed more and more application needs. This ever growing increase in application requirements and the ubiquity of computer systems brought on the first major milestone in the history of the grid.

A.2 Metacomputing

The proliferation of high-performance distributed systems paved the way for the next step in distributed computing and, perhaps, the first generation of grid systems. The objective was to build a *metacomputer*, i.e. a virtual supercomputer assembled using a large number of geographically distributed but interconnected computational resources. Two initiatives spearheaded this revolution in the mid 1990s: I-Way and FAFNER.

The I-Way networking experiment [123] connected supercomputers, databases, scientific instruments, and other resources located in 17 different North American sites using 11 high-speed (IP and ATM) networks. I-Way provided an environment with parallel programming, scheduling and security functionalities along with a robust distributed file system.

I-Way demonstrated the capabilities that could be offered by assembling such a system. It offered huge potential for undertaking difficult scientific problems that were simply infeasible to tackle in any other way. Moreover, I-Way was proof that such powerful and empowering system could be dynamically assembled and scaled using heterogeneous resources that reside in multiple administrative domains. The success of I-Way led to the DARPA investment into developing an infrastructure to enable metacomputing. The Globus Toolkit (GT) [154] was developed by the Globus Alliance [38] to do this by providing solutions to problems that are commonly faced when developing grid applications. GT includes open-source software libraries to support services such as data transfer and management, monitoring and informa-

tion, resource management, fault detection, and security. The abstractions provided by these libraries mask a great deal of the intricacies of managing heterogeneous grid elements and are hence very useful for developing grid applications. GT grew to become the de facto open source solution for grid computing.

The other prominent early example of using distributed resources to solve complex problems is the Factoring via Network-Enabled Recursion (FAFNER) project. This project was aimed at breaking RSA encryption by dividing the problem into embarrassingly parallel numerical factoring sub-problems. These sub-problems could be run on different computers (any computer with 4MB of memory) that are registered for anonymous contribution.

Unlike I-Way, FAFNER did not result in any major core grid technologies; Data and results were communicated via a Web server using HTTP. Nevertheless, it also demonstrated that complex problems could be efficiently tackled using an almost ad hoc set of resources across long distances and through administrative boundaries. More importantly, FAFNER presented a pioneering experience that would be reproduced in what has become to be known recently as *volunteer computing* where users register their willingness to share their own computer as a way of contributing to solving complex problems. The registered computers would be used as workstations to run certain algorithms on small datasets they receive from a central manager. Examples include distributed.net [9], folderol [241], Models@Home [230] and BOINC [65].

A.3 Distributed Communities

The growing ubiquity of computing systems, improving communication technologies, and increasing software sophistication, along with the success of metacomputing projects such as I-Way and FAFNER, all led to the birth of a second generation of heterogeneous distributed systems. Huge academic and commercial efforts were focused on devising such systems, either by creating new technologies or by extending prior ones to enable grid computing for different practices. These technologies ranged from low-level resource management to high-level user interfaces. Each of these technologies was quite specific as each was geared towards a certain target user community and had a certain amount and type of resources to use. Hence, every technology had an almost unique set of goals. However, there were a few commonalities. We use these to present the main ‘themes’ of this period.

There were five main broad themes. Software such as GT and Legion offered a

degree of homogeneity through **middleware** solutions that serve as a platform on top of which grid applications could be built. Other efforts, such as CORBA and Jini, focused on **distributed object architectures** (DOA) to promote interoperability. The distinction between DOA and middleware solutions might seem subtle, yet it dictates a huge difference in how applications are constructed.

The third theme was **resource brokers** or **schedulers** which focused on seeking and aggregating unused resources. Examples include Condor, LSF, Nimrod, and SGE (formerly CODINE). The fourth theme of this generation included tools that were created to allow users to have **simple and interactive access** to the services and resources that are available via the grid. Examples include end-user execution interfaces such as GECCO and visualisation portals such as Webflow. An important example here is SETI@Home. This was a volunteer computing project that aimed at analysing sound samples captured in outer space to search for any patterns that could signify the existence of extraterrestrial intelligence.

The final theme was **integrated systems** which cover many different areas that the previous four themes did. These solutions provide abstractions to allow highly complex applications (e.g. intensive computations on large distributed databases) to use the grid for scientific and engineering purposes. Examples include Cactus, DataGrid, UNICORE, and Jxta.

All the work done under the above themes was at least partially motivated by the prospects of grid computing which commanded attention in research and business circles. What followed were two important developments. The first was the exodus of legacy systems that were confined to a fixed geographical location to a realm where systems incorporate many resources and users from more than one organisation. The second development is of more importance and it consisted of a large number of initiatives to create scientific applications of remarkable scale. An example of these initiatives is the LHC. These applications were extremely demanding, and hence they became and still are to a great extent the driving force behind grid computing.

Finally, this phase in grid computing history also saw the birth of the Global Grid Forum (GGF). The GGF was a congregation of shared interest that included several research and work groups that explored and developed a myriad of grid technologies and standards. This work had a significant impact as it enabled the birth of a number of grid projects aimed at solving complex computational problems, the main benefactor being the scientific research community at large. In essence, the GGF played a key role in constructing highly ambitious research projects, referred to as

e-science. In doing so, the GGF had another important achievement in drawing more interest to the grid computing world. This knock-on effect saw the GGF steadily grow in size until it merged with an industry interest group called Enterprise Grid Alliance (EGA) to form the OGF.

A.4 Standards

The expansion of the distributed computing scene came with clear benefits in terms of creating new grounds for applications. However, just like what happens with various fields of computer science, this expansion was uncontrolled and undirected. This resulted in a large number of efforts that focused on more or less the same things but were ‘non-interoperable’.

Such lack of interoperability clashed with the original vision of the grid [224]. The architecture had to be modified to facilitate more integration. This was achieved when, following the recommendations put forward in [157, 156], the GGF passed the Open Grid Services Architecture (OGSA) [158] as a standard for grid applications. OGSA built on the strengths of Web service technologies, namely different systems interoperating using structured text over HTTP. OGSA describes means of sharing information between different types of resources, and is thus an effort to maintain high interoperability in heterogeneous environments.

The Open Grid Services Infrastructure (OGSI) [344] was another GGF standard that provided the interfaces for the underlying infrastructure. Both the OGSA and OGSI standards were adopted by the e-science and e-business communities. The use of these standard interfaces offered much more interoperability between grid components and applications. This marked another milestone in the pursuit for more engaging computing.

A.5 Service-Oriented World

In the meanwhile, Web services were gaining even more success, and there was a growing feeling within the grid community that using a Service-Oriented Architecture (SOA) is the way forward for deploying dynamic, interoperable grid applications. This prompted further convergence between the Web and grid communities, which was attained when the WSRF¹ standard replaced OGSI as the infrastructure standard for

¹The Web Services Resource Framework (WSRF) is part of the Web Services Architecture specifications [84] that were defined by the Organization for the Advancement of Structured Information

OGSA version 1.5. Using this setting, OGSA became an “a particular profile for the application of core [Web Service] standards” [159].

In order to be WSRF-compliant, all services need to comply to a number of standards; including being defined using WSDL, exchange information structured using XML over SOAP, traverse XML objects using XML Path Language (XPath), etc.

The National Grid Service

This appendix provides some general information about the hardware and software infrastructure of the NGS, with some focus on its scheduling provisions.

In November 2009 the NGS comprised of 15,000 processing cores at 25 sites around the UK [302]. These and other resources, including data repositories and scientific instruments, are connected by SuperJANET, the UK's academic network. Core NGS services are hosted at sites in Belfast, Cardiff, Didcot, Glasgow, Lancaster, Leeds, London, Manchester, and Oxford. More information about the resources at individual sites is available from [26]. Users of the NGS can also use their certificates to access resources outside the NGS such as those of GridPP, EGEE, GILDA, OpenScienceGrid and TeraGrid. In the near future, NGS will be integrated with the European Grid Infrastructure (EGI) [12].

The NGS software stack includes an array of grid technologies to enable deployment of a wide range of grid applications. Local resource management is a site-specific resolution and ranges between PBS, SGE, LSF and Condor. Globus Toolkit is primarily used as the middleware solution. However, due to the integration with GridPP which uses EGEE's gLite as its middleware solution, gLite is also supported and is adequately integrated with GT. UNICORE is also available to use. Globus Grid Security Infrastructure (GSI) and Shibboleth are used for authentication and authorisation purposes. For data services, a series of data management solutions are installed on different sites including AFS, MySQL, Oracle, SRB, and others. OMII-UK solutions are also ready to use with NGS, including GridSAM [28] and OGSA-DAI [29].

A number of other tools are made available by NGS to help users with certificates, keys, proxies, files and workflows.

Until quite recently, NGS users were expected to carry out their own metascheduling by assigning jobs to instances of GT running on LRMs of their choice. This was not only tedious and assumed a certain level of familiarity with grid technologies (such as GT), but it also meant that most users defaulted to static scheduling. To avoid the drawbacks of static scheduling, some users would come up with their own dynamic scheduling mechanisms. This obviously requires updated resource information, which involves even more knowledge of NGS's inner workings from the users.

In December 2009, NGS adopted gLite 3.1 WMS as a more convenient way of submitting jobs to the grid. WMS finds the most suitable resources for a submitted job, sends the job along with any input files for execution on these resources on behalf of the user (using GridFTP), keeps a record of the job's execution state, and retrieves any output files and/or error logs. This is a considerable improvement in terms of usability and efficiency as it saves users a substantial amount of work that would have involved using GT GRAM, GridFTP, SRB and possibly tailored scripts.

gLite WMS

gLite [234] is a service-oriented middleware solution developed by the EGEE project [10]. It is the successor of the LCG-2 middleware [277], replacing it on CERN's LCG. This appendix reviews gLite's meta-scheduling service, the Workload Management System (WMS) [67].

C.1 Architecture

The gLite WMS architecture is shown in Figure C.1¹. The diagram shows the different gLite components that are involved in scheduling a job using WMS.

The User Interface (UI) is the users' access point to a gLite-managed grid. The Resource Broker (RB) is the central element in the gLite architecture. It is essentially a grid-wide meta-scheduler not dissimilar to Condor-G. It oversees all procedures necessary for job execution such as information retrieval, matchmaking, execution stage preparation and event logging. Information about grid resources are kept in BDII, an LDAP directory service, while events and job status changes are stored by the Logging & Bookkeeping (LB) database. Once a job is cleared for execution, the RB communicates all required executables, scripts, parameters, and input files to the Computing Element (CE). The CE is also called the headnode as it acts as a site's interface with the rest of the grid. Behind the headnode lies a number of Worker Nodes (WN) and Storage Elements (SE). Files stored on SEs are indexed on the LCG

¹This figure is adapted from [266].

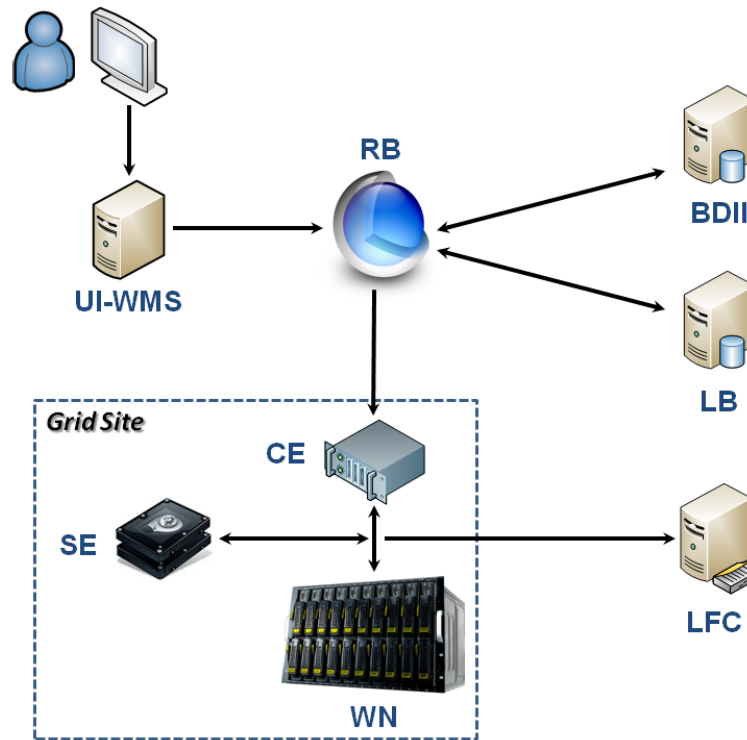


Figure C.1: *gLite Job Workflow*

File Catalog (LFC). All interactions between gLite elements are carried out through Web Service interfaces.

C.2 Job Description

In order to submit a job to gLite WMS, its properties and requirements need to be defined using the Job Description Language (JDL) [133]. JDL is a high-level language to describe job properties and requirements using attributes from the GLUE Schema introduced in 3.2.2. Listing C.1 shows a JDL file describing a video encoding job.

```

Type = "Job";
JobType = "Normal";
Executable = "lencod.exe";
Arguments = "-d encoder_main.cfg";
InputSandbox = {"video.yuv", "encoder_main.cfg", "lencod.exe"};
InputSandboxBaseURI = "gsiftp://ngsui03.ngs.ac.uk:2811/home/ngs0725/videoEnc/";
StdOutput = "video.264";
StdError = "video.err";
OutputSandbox = {"video.264", "video.err", "stats.dat", "data.txt", "leakybucketparam.cfg", "log.dat"};
OutputSandboxBaseDestURI = "gsiftp://ngsui03.ngs.ac.uk:2811/home/ngs0725/videoEnc/output/";
RetryCount = 3;
ShallowRetryCount = -1;
Rank = other.GlueCEStateFreeCPUs;

```

Listing C.1: *JDL Example*

C.3 Scheduling Process

The WMS scheduling state machine is portrayed in Figure C.2.

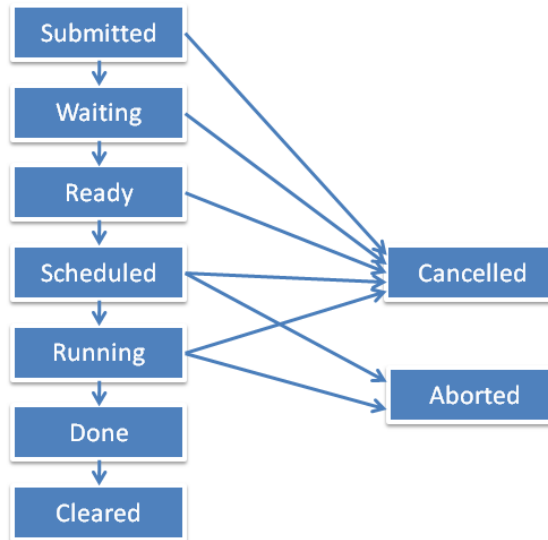


Figure C.2: Job Cycle Using gLite WMS

Once described, the job is ready to be submitted to WMS. WMS parses the JDL, transfers any required files to the RB, and sets the job status to ‘Submitted’. Then WMS starts to look for resources that best fit the job requirements and the status is set to ‘Waiting’. Once appropriate resources are found, the job is allocated a slot in the queue of the remote LRM while WMS creates wrapper scripts and environment variables. The completion of these tasks flag the ‘Ready’ state. Once the remote LRM starts handling the job, queuing it locally, the job status changes to ‘Scheduled’. The status changes to ‘Running’ as soon as the input files are copied to a location accessible from the execution theater and execution starts. If execution completes, output files and logs are sent to the location specified by the JDL and the status is changed to ‘Done’. The status is finally set to ‘Cleared’ as soon as the output is retrieved by the user. If the execution theater is not set up properly or if the execution is preempted by another job, the job is resubmitted to another resource until the maximum retry count specified in the JDL is reached, at which point the status is set to ‘Aborted’. The remaining states are ‘Cancelled’ (if the job is cancelled by the user before it is flagged as either ‘Done’ or ‘Aborted’) and ‘Unknown’.

For more information, refer to section 3.4 of the gLite user manual [90].