Journal of Grid Computing manuscript No.
(will be inserted by the editor)

Passive Network Awareness as a Means for Improved Grid Scheduling

Yehia Elkhatib · Chris Edwards

The final publication is available at Springer via http://dx.doi.org/10.1007/s10723-015-9332-0

Abstract Grids enable sharing resources of heterogeneous nature and administration. In such distributed systems, the network is usually taken for granted which is potentially problematic due to the complexity and unpredictability of public networks that typically underlie grids. This article introduces GridMAP, a mechanism for considering the network state for enhancing grid scheduling. Network measurements are collected in a passive manner from a user-centric vantage point. This mechanism has been evaluated on a production e-science grid infrastructure, with results showing the ability of GridMAP to improve grid scheduling with minimal network, computational and deployment overheads.

Keywords network awareness · network analysis · network performance prediction · passive network measurement · grid scheduling

1 Introduction

Grids are complex networked systems made up of heterogeneous, shared, and geographically distributed resources. Grids deal with such a complicated setup by employing a *scheduler* that finds resources that best suit a particular job. The scheduler is essentially solving an optimisation problem between resource availability and job requirements and, hence, its performance is a measure of its ability to match jobs to resources that satisfy their requirements. Given the relative ease of obtaining job requirements (usually specified by the user), scheduler performance hinges on its ability to identify available resources and to assess their capabilities. In a branch of distributed computing that still relies to a great part on "data to computation" rather than on computation mobility, it is thus necessary to obtain accurate information about network performance since this directly affects data transfer. However, most grid technologies rely on either or both assumptions about the network: it is *overprovisioned* while offering best-effort only, or it supports *advance reservation*. We find neither to be valid.

The overprovisioning argument, i.e. increasing network capacity to meet foreseeable utilisation demand, is common as to many it is enough to dissolve concerns about non-guaranteed performance. However, overprovisioning is not always appropriate, sufficient or 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 [1]. Second, overprovisioning is only a transitory remedy considering the rate at which application requirements grow [2]. Third, overprovisioning offers no guarantees, just an optimistic outlook that increased capacity would be sufficient. At times of contention surges, such best-effort approach fails resulting in inflated RTT [3]. Fourth, overprovisioning is sometimes practically not possible due to cost or otherwise, e.g. in wireless environments.

In situations where overprovisioning is not possible, advance resource reservation is called upon to ensure adequate network performance. This approach, however, is only feasible where there is a certain level of control over network resources, such as in bespoke infrastructures, but not public networks. Moreover, it requires a priori knowledge of job requirements which is seldom viable.

The School of Computing & Communications, Lancaster University, LA1 4WA, U.K.
E-mail: { y.elkhatib, c.edwards }@lancaster.ac.uk

There is, however, a third approach arising from autonomic communications: *Network-awareness* is recognising the network as a shared resource, the use of which should be managed without assuming control. A common example of this is TCP flow control and congestion avoidance algorithms. Applying this in grids, network measurements would be used to optimise scheduling. We argue that a network-aware grid scheduler resulting from such integration is better at locating resources than a network-oblivious one.

Most current scheduling technologies are network-oblivious and hence do not cater for the network requirements of applications. This phenomenon is surprising considering the importance given to network-awareness in the literature [4–7]. Nonetheless, there has been some network-awareness efforts yet they suffer from certain drawbacks (§2). This provides impetus for enabling network-aware scheduling through providing grid schedulers with accurate updates about the state of the network, in order to allow them to adapt to changes in the network as they do to changes in the state of computational resources.

This work aims to enhance grid network management using sustainable and scalable monitoring techniques. We propose GridMAP, a solution designed to passively collect network performance measurements and use them to forecast the state of network paths. Such forecasts are used to enrich resource supervision, enabling network-aware scheduling.

Although network-aware scheduling is not particularly a new proposal, previous proposals (such as [8–10]) have very little if any experimental evidence of their effectiveness in real world scenarios. We evaluate GridMAP by using it to enhance scheduling in a UK-wide production grid infrastructure that runs very high loads and is subject to occasional outages. This setup is chosen to investigate how GridMAP can help grid applications adapt to infrastructure unpredictability. The results of our evaluation demonstrate unmistakable improvements engendered by passive network measurement. We found that scheduling performance is improved and the fluctuations within it are significantly attenuated. RTT was reduced by a factor between 16.8% and 57.7%, while throughput was increased by a factor between 12.4% and 136.6%. Our results also examine the network, computational and deployment overheads of using GridMAP, which were found to be minimal. Such significant improvements to the execution of real-world grid applications ratify our argument for network-aware grid technologies.

The rest of the article is organised as follows. §2 presents related work and highlights shortcomings in supporting network-aware grid scheduling. §3 introduces GridMAP design and §4 discusses implementation details and the evaluation setup using real-world grid applications. §5 presents the results of deploying GridMAP on a production grid infrastructure. Finally, §6 concludes and presents future work.

2 Related Work

A Grid Information System (GIS) carries out 4 dependent information handling functions: *a*) collection; *b*) management; *c*) dissemination; and *d*) consumption. Considering that a GIS is the main supplier of status information to a scheduler, we argue that current GISs do not offer suitable forms of facilitating network-aware scheduling. We turn our attention to this argument by reviewing the drawbacks of current GISs in light of each of the above functions.

2.1 Information Collection

Successfully managing the tradeoffs associated with collecting network information is the first step to attain network-aware scheduling. We explore these in light of the most common techniques used to acquire network measurements: active measurement, tomography, and ICMP.

Active measurement tools (e.g. gloperf [4] and DIANA [11]) are widely used for their high reliability and accuracy. The accuracy of such tools is in fact not as guaranteed due to the *preludial effect*: during times of increased load, metrics computed using a probe that preludes a real flow could vary considerably from those experienced by the real flow itself [12,13].

There are two main overheads associated with active techniques. The first is added traffic which fills communication channels and router queues, ultimately resulting in inflated RTT and packet loss. Second is deployment overhead due to required peer coordination. This can hinder deployment on production sites due to fear of service disruption, security concerns, etc. Other requirements such as kernel modification (e.g. Wren [14]) and use of designated measurement machines (e.g. NWS [15]) also present deployment obstacles.

Some tools attempt to reduce measurement overhead but only to sacrifice accuracy. For example, NWS resolves to using probes of reduced size that are not sufficient to challenge TCP and yield underestimated bandwidth [16–18]. Other tools (such as gloperf [4], DIANA [11] and NSSD [19]) reduce the number of, rather than the size of, probes by adopting tomography. Using this technique, measurement is carried out only be-

tween a subset of machines and then used to estimate for non-measuring ones.

An alternative is to replace active probes with ICMP messages (e.g. iPlane [20] and SONoMA [21]). This is unreliable [22] as it is not uncommon to rate-limit [23] or even filter [24] ICMP traffic.

2.2 Information Management

GIS architectures are either centralised or distributed. Centralised GISs (such as iPlane and DiPerF [25]) suffer from two main drawbacks: restricted scalability [26] and a single point of failure that threatens availability.

Decentralised or distributed models are better suited to maintain scalability and availability. Examples include Globus MDS [27], and R-GMA [28]. Many of these are based around the GMA multi-publisher model [29]. The main problem with this setup is the overlap of monitoring responsibilities resulting in data discrepancy. Thus, obtaining a valid representation of system resource status comes at a significant maintenance overhead [30, 26]. Such costs associated with the asymmetric dispersion of information across disjoint sources undermines the provision of network-awareness to schedulers, and impedes long-term performance analysis.

2.3 Information Dissemination

GMA-based GISs actually complicate network-aware scheduling by employing multiple publishers. 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 a model, it needs to query the directory service to learn about the available information collectors and then proceed to query each of them about information they manage. This is a lengthy and taxing procedure considering the high performance demands of grids. The situation is worsened as the number of registered collectors increases [26] which, ironically, limits the scalability of such decentralised GISs.

2.4 Information Consumption

Some GISs (such as eTOP [31] and Ganglia [32]) are developed for human consumption and not for interoperation with other grid solutions. This is useful to users and administrators who wish to visualise load status and investigate anomalies on a case-by-case basis. It does not, however, make it easy for schedulers to obtain network state information.

2.5 Summary

Current GIS solutions are not suitable for enabling efficient and accurate network-aware scheduling for 3 main reasons. First, many GISs adopt different techniques to strike a balance between accuracy and overhead. Such practices either complicate deployment or compromise accuracy which is problematic when built upon (e.g. for prediction) as even relatively small error margins get significantly amplified [16]. Second, the architectural model of many GISs either suffers from poor scalability and availability, or creates a high communication overhead to manage and disseminate collected information. Finally, some GISs are not designed to produce machine-readable output and are thus not suitable. The few GISs that elude such drawbacks (such as [33]) tend to focus on CPU resources and not the network. In contrast, GridMAP is a solution that is specifically designed for schedulers to improve their network utilisation. It relies on a measurement technique that is low-overhead and easy to deploy, and has a distributed data management model to facilitate scalability.

3 GridMAP Design

GridMAP (Grid Monitoring, Analysis and Prediction) [34] is specifically designed to aid grid schedulers attain network-aware scheduling. This is done by providing information about the state of the network, enabling schedulers to adapt to contention over end-to-end network resources. In order to avoid the shortcomings of similar works, GridMAP is designed to meet certain requirements: low overhead, easy deployment, interoperability, and the capability to mine through collected network information to provide additional information such as performance predictions.

GridMAP carries out 3 tasks: to collect, manage and disseminate measurements. These tasks are implemented using a twofold design: *Pasinemda* carries out the measurements and hands them over to the *GridMAP service*, which stores and manages the measurements and uses them to provide predictions of network performance on request to grid schedulers.

3.1 Information Collection

The *PASsIve NEtwork Measurement DAemon* (Pasinemda) runs as a background daemon on the user machine, passively calculating 2 fundamental network metrics, namely RTT and throughput, by relying on one of the intrinsic properties of grid applications which is the abundance of TCP flows [35].

A TCP flow is made up of three phases: connection establishment or *handshake*, data exchange, and connection termination or *teardown*. The handshake process, as portrayed in Figure 1, is initiated by an empty packet with the SYN flag raised in the TCP header. This is referred to as a *SYN message*. The destination machine replies with a similar packet with two TCP flags set: SYN and ACK. The process is completed with an ACK message from the handshake initiator. This three-way handshake process establishes a connection and its associated packet sequence numbers.

By monitoring such flows naturally created by grid applications, RTT r is measured as the delay between SYN messages and their corresponding SYN-ACK responses, i.e. $r = t_1 - t_0$. Achieved throughput h is calculated as the amount of data transmitted per unit of time, i.e. $h = \frac{d}{t_{flow}}$ where d is the total number of data bytes sent in the TCP flow and t is the total duration of the flow from handshake acknowledgement till the time of the first FIN signal identifies teardown initiation.

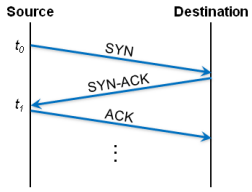


Fig. 1: TCP Three-Way Handshake Process

This unobtrusive approach of using TCP handshakes to extract RTT keeps added overheads to a minimum while at the same time refrains from compromising accuracy [36–38]. It also avoids the disadvantages of ICMP-based probes that could be blocked [22]. However, this technique has traditionally been applied using synthetic SYN packets, which can be easily mistaken for TCP-SYN floods [37]. Pasinemda avoids this by not creating any artificial network probing traffic.

Pasinemda is light-weight (i.e. low overhead) and its deployment is independent of other tools or measures. Pasinemda's unobtrusive nature makes grid traffic monitoring an involuntary process that requires no intervention but is sustained as traffic passes through a node submitting jobs and/or data to the grid. 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. Pasinemda parses any cached measurements into XML using the NM-WG schema and submits them to the service. This is carried out, by default, every 15 minutes[1]. This is received on the service side by invoking the Measurement Receiver module, as will be described in the following subsection. More design details of Pasinemda as a measurement daemon has been put forward in [39], along with an evaluation of the accuracy of its measurements against ping and iperf. These measurements constitute the information that flows through the rest of the GridMAP system. The remainder of this paper, however, is concerned with the rest of the GridMAP system and how Pasinemda's measurements are managed and processed in order to improve the performance of grid scheduling.

3.2 Information Management & Dissemination

The *GridMAP service* is an application whose functionality is exposed via stateful grid service end points, conforming to WSRF and OGSI specifications. It is deployable on one computer or across a number of them.

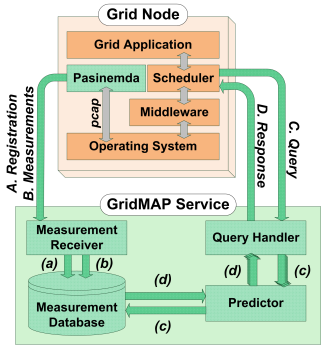


Fig. 2: Interactions of GridMAP Service with Pasinemda and Grid Schedulers

The way the GridMAP service interacts with Pasinemda and grid schedulers is depicted in Figure 2. In the figure, external interactions are labelled using capital letters (e.g. 'A'). Internal operations resulting from each external interaction are labelled using the same letter in small case (e.g. 'a'). The service has 4 main internal components: *Measurement Receiver*, *Measurement Database*, *Predictor*, and *Query Handler*. There is also the service stub, an intrinsic gatekeeping component of any Web or grid service, which provides a WSDL definition of the service and what information is needed to invoke the service's functionality. The stub also marshals communications between local and remote components

[1] The reporting period could be changed independently by each daemon through a command line argument to suit the needs of different applications.

using XML over SOAP. The service stub is omitted to simplify the portrayal of invocations.

Pasinemda goes through a bootstrapping phase (step A in Figure 2) during which it registers with the GridMAP service the hostname of the user machine where it resides and all associated IP addresses. These details are stored in the database (step a). Then, Pasinemda periodically submits cached measurements to the GridMAP service using NMWG XML schema [40] over HTTP (step B). This not only conforms to standards but also allows communication through firewalls. The *Measurement Receiver* inserts the measurements received from Pasinemda into the *Measurement Database (MDB)* (step b). MDB is a repository that holds all the performance measurements collected by Pasinemda. It is managed using MySQL and distributed using statement-based MySQL Replication.

This modular, symmetric design allows the GridMAP service to scale to meet higher demand simply by adding new machines to the set that hosts the GridMAP service. Conformity is maintained across all machines by duplicating data changes in an asynchronous fashion. This improves system resilience against machine failure, including the service front.

The *Predictor (PR)* component comprises of numerical algorithms (described in Section 3.3) that analyse period-based or observation-based measurement history from MDB triggered by a query from a grid scheduler (described in the next paragraph). Period-based history analysis is where measurements obtained during the past period of time (say, 12 hours) are used for prediction. The is more suitable for production grids where frequent network activity is anticipated. On the other hand, observation-based history analysis is where a certain number of past measurements are used for prediction regardless of when they were collected. This method is better suited for grids that would generate sparse measurements, such as volunteer computing grids.

The *Query Handler (QH)* is invoked when a grid scheduler contacts the service enquiring about the network performance of particular grid sites (step C in Figure 2). QH triggers a series of calls to PR using different combinations between the enquiring user node and the candidate grid sites (steps c). The PR calls are distributed to different instances of the GridMAP service deployment in a round-robin fashion. Once complete (steps d), QH summarises the network performance forecasted by PR for each candidate site into a weighted list. Finally, the weighted list is combined and sent back to the querying scheduler using NM-WG schema (step D).

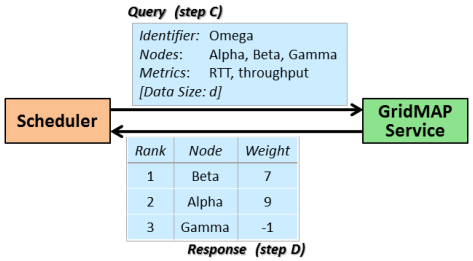


Fig. 3: Query and Response Exchange between Scheduler and GridMAP Service

Now that we have dissected the components of the GridMAP service, we reduce it to a black box and focus on the information exchanged between it and a grid scheduler. In the example portrayed in Figure 3, a scheduler running on user machine Omega is about to allocate resources to execute a job. The scheduler invokes the GridMAP service by identifying itself, the set of grid sites that satisfy the job's computational requirements (Alpha, Beta, and Gamma), and the network metrics that are of interest to this type of job (RTT and throughput). The total size of the files to be transferred (data files, parameter files, executables, etc.) is also provided since throughput is specified as a metric of interest.

The GridMAP service responds with a list weighted according to the projected network performance based on the requested metrics. Lower weight indicates better predicted network performance. In the example, the performance of the network path to site Beta is anticipated to be better than that to site Alpha by a ratio of 7:9 based on previous network measurements recorded by the service (more details on the weighing algorithm in the next subsection). A weight of (-1) is used to signify a lack of measurements for a user machine-grid site pair, as in the case of site Gamma. The scheduler then picks the optimal site according to local policy in light of GridMAP's response as well other GIS information.

3.3 Data Analysis

For the predictor component to predict future metrics based on the history in the database, it needs an algorithm that can process an irregularly-spaced discrete data series. This algorithm also needs to be rapid as a long turnaround would defy the purpose. Thus, there is an emphasis on efficiency and pragmatism as opposed to maximising forecasting accuracy.

Common analysis methods (such as spectral analysis or Box-Jenkins) are not applicable here as the measurements do not form a regular time series. Other

techniques suited for irregularly spaced series (such as weighted wavelet Z-transform, parametric clustering, and iterative singular spectrum analysis) are computationally intensive. We therefore employ interpolation to compensate for inconsistent periodicity by approximating the value of missing samples. This then qualifies the series to be used by traditional forecasting methods.

Let x_i denote a measurement taken at time t_i. X represents an irregularly spaced series of N distinct measurement pairs:

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

This series is then interpolated using the Akima spline fitting method [41] as it is very stable to outliers and has minimum data assumptions. Moreover, being a parsimonious method, its implementation is of low complexity and requires a single linear application per series. The result is an equispaced series X':

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

which facilitates the extraction of accurate growth trends. It cannot, however, be used to produce accurate forecasts through extrapolation 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 measurements. Therefore, forecasting for the coming period, $\hat{x}_N$, is computed using linear stochastic filters. This procedure continuously adapts to new measurements as they become available, and is fast to execute.

When $N < 3$, past samples are not sufficient to avoid sensitivity to exponential smoothing shifts and hence a moving average is used:

$$\hat{x}_1 = x'_0; \hat{x}_2 = \frac{x'_0 + x'_1}{2} \tag{1}$$

For $N \geq 3$, exponential smoothing is used to improve accuracy:

$$\hat{x}_N = \alpha x'_{N-1} + (1 - \alpha)\hat{x}_{N-1} \tag{2}$$

where α is the *smoothing constant*, in the range $0 < \alpha < 1$, that declines exponentially for older samples. The process is reduced to a non-recursive form by continuously replacing previous forecasts into equation 2:

$$\hat{x}_N = \alpha \sum_{i=1}^{N-1} (1 - \alpha)^{i-1} x'_{N-i} + (1 - \alpha)^{N-1} x'_0 \tag{3}$$

Following this, the Mean Square Error (MSE) is computed. This will be used to tweak α in order to improve the subsequent forecast. The last values of α, MSE, PR invocation parameters and returned results are cached under the scheduler's identifier for a lease-based lifecycle, which in our implementation is 60 minutes. This relatively short value is set to reduce caching overhead whilst allowing enough room for schedulers to reuse and build on previous queries. Longer periods would be required for user machines with more sparse network interactions and hence measurements.

Finally, the predictions are used to calculate weights that will be used to order the list of candidate sites. The weights are unitless indicators of 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:

$$w_r = a\hat{r} \tag{4}$$

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

$$w_h = \frac{d}{\hat{h}} \tag{5}$$

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:

$$w = bw_r + cw_h \tag{6}$$

Computing w could be fine-tuned through the GridMAP service interface by passing the values of constants a, b, and c.

3.4 Information Consumption

This function deals with incorporating GridMAP into the scheduling process. Essentially, it is broken down into the following tasks:

- *Interrupt* the matchmaking process at a suitable time 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.
- *React* to lack of measurements indicated by negative weights.

Depending on the available grid software stack, the above tasks are implemented using one of three methods. The first option is to modify the scheduler source code (if available) which provides great programmability, but could potentially be significantly time-consuming. An alternative is *hooking* where the scheduler's behaviour is altered at runtime by intercepting method invocations and injecting modified calls and/or data structures. However, this entails a great deal of reverse engineering which could make it just as time-consuming.

The third method is *wrapping* where the scheduling process is veered by altering job descriptions. This method requires little knowledge of the scheduler's inner workings, but offers limited flexibility as it is bound to the constraints of the scheduler interface.

4 Experimental Setup

This section presents the setup devised to measure the impact of GridMAP on scheduling.

4.1 GridMAP Deployment

4.1.1 Pasinemda

The user machine runs an instance of Pasinemda which allows it to collect network measurements based on job submissions and file transfers, and report them to the GridMAP service every 15 minutes. At job submission time, the user machine interacts with the GridMAP service to achieve network-aware resource allocation. The user machine was a Dual-Core AMD Opteron 2.2GHz with 2GB RAM running CentOS 5.3, JDK 6_16, Globus Toolkit 4.2.1 and Apache Axis2 1.5.1, and it resided on the Lancaster University campus.

4.1.2 GridMAP Service

To improve availability, the GridMAP service was deployed over 4 separate machines. These were in fact virtual machines (VMs) hosted on the EmuLab testbed [42]. Each was an Intel Xeon 3GHz with 896MB RAM and 3GB disk space running Red Hat 9, JDK 6_16, GT 4.2.1, Apache Axis2 1.5.1 and Apache Tomcat 6.0.20. The database was managed by MySQL Community Server 5.1.42. Data analysis was carried out using R 2.9.2. EmuLab is located in the University of Utah in Salt Lake City, UT, USA. This makes the GridMAP service reachable from the user machine in Lancaster via a 20-hop connection of 180ms RTT on average.

4.1.3 Induced Load

The deployment of the GridMAP service was augmented by a dummy deployment to generate network, storage and processing overhead on the service during the experiments. This included 20 VMs, also hosted on EmuLab, running Pasinemda. At boot time, every VM randomly selects a number s between 5 and 60 to represent the number of minutes before which Pasinemda reports its cached measurements to the GridMAP service. The VMs exchange a dummy data set (10MB in size) using GridFTP in the following manner: each VM issues a query to the GridMAP service, transfers one of the data sets to one of the other 19 VMs, both 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 running Red Hat 9, JDK 6_16 and GT 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 VMs also serve to synthesise a realistic network and processing load on the service.

4.2 Grid Testbed

The National Grid Service (NGS) was chosen to run the experiments. It is a production grid infrastructure put in place to aid UK researchers in deploying and running grid applications. The NGS follows an *'islands of grids'* (IoG) model: partner and affiliate sites (i.e. the islands) are hierarchical systems in the form of high performance computers or clusters. Each site is locally governed by a *Local Resource Manager* (LRM) typically running from the *headnode*, a gateway to the site's *worker nodes* that execute jobs and store data. Sites are interconnected via SuperJANET5, the UK's academic network backbone operating at 10Gbps.

The NGS software stack comprises of Linux, GT and gLite. The basic NGS virtual organisation (VO), called *ngs.ac.uk*, encompasses resources at 5 sites. Resources at 20 more sites are available for members of other VOs. Coordination between LRMs of different sites is done by the gLite Workload Management System (WMS) [43]. A user has to supplement each submitted job with a job description file that specifies the job properties and requirements using JDL, a high level language. Accordingly, 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 its pool of local resources. WMS is thus a *meta-scheduler*.

4.3 Grid Applications

We used real grid applications from the e-science domain. We decided to choose one application as a case

study from each of the 3 main classes of grid applications: HTC, HPC, and Data Grid.

High Throughput Computing (**HTC**) applications carry out profuse, linearly executed computations based on input parameters which could be either preset (i.e. parameter sweep applications) or not (e.g. Monte Carlo simulations). Such computations are referred to as *embarrassingly parallel* since their processes are run in a virtually independent fashion. High Performance Computing (**HPC**) applications too are CPU-intensive and not data-intensive. However, the execution nature of these computations is parallel over a number of grid nodes with significant interaction between the different processes. **Data Grid** applications exchange large datasets (containing observational data, simulation results, etc.) between grid nodes in order to be processed (e.g. sanitised, transposed, etc.) and reliably stored.

We now describe the case studies chosen from the above application classes.

4.3.1 A: AutoDock

AutoDock [44] is a suite of open-source **HTC** molecular docking[2] applications. The total input size (docking model parameters and atomic structure maps) does not usually exceed 10MB in size. The output report details the docking scenarios undertaken and the resulting molecular interactions and is typically under 100kB in size. AutoDock 4.0.1 is used, available at 3 sites: *Leeds*, *OeRC* and *RAL*.

4.3.2 B: PMEMD

Amber [45] is a suite of molecular dynamics programs, the most common of which is Simulated Annealing with NMR-Derived Energy Restraints (Sander) which is an **HPC** program that assigns certain atoms to different processors, with MPI-based inter-processor communication. Particle Mesh Ewald Molecular Dynamics (PMEMD) is a variant of Sander with improved performance. Typical input, hardly ever exceeding 10MB, details molecular structure and properties. This is accompanied with a configuration file to control the simulation. Execution could last up to a few hours per process. Output size depends on the properties of the molecular system (number of atoms, initial coordinates, velocities, force fields, etc.) but is usually small compared the input. We use Amber 10.0 available at 2 sites: *Leeds* and *RAL*.[3]

[2] This technique investigates the binding of molecules, such as drugs, to the receptors of 3D structures, such as proteins.

[3] The authors thank these sites for providing access to the Amber application suite.

4.3.3 C: LogStore

Due to a lack of access to a representative **Data Grid** application, LogStore was developed. Analogous to NetLogger, LogStore archives records of system events and notifications obtained from a group of computers. The files are compressed and sent as a grid job to be decompressed, sanity checked, and indexed in an RDBMS. The monitored computers were 32 VMs used over 11 months to carry out various functions including prototyping, data analysis, and hosting the GridMAP service. LogStore is available on 5 sites: *Leeds*, *Manchester*, *OeRC*, *RAL* and *Westminster*. The total size of the repository collected over 11 months was 110.66MB.

4.4 Implementing GridMAP on NGS

Based on the options discussed in Section 3.4, we opted for wrapper-based integration for its ease of development and applicability to a range of widely used schedulers, particularly metaschedulers (such as Condor-G, GridWay and gLite WMS) that are commonly used in large grid deployments. Our wrapper, called *gSched*, is implemented for gLite WMS. gSched resides on the user machine described in Subsection 4.1.1. From this location, gSched is able to log into the WMS interface and site headnodes, and issue commands to inquire about the state of resources and to manage jobs based on network information retrieved from GridMAP. gSched reacts to a negative weight by opening a half-connection to the remote grid node in question and immediately closing it. This acquires one initial RTT measurement per candidate grid node without resembling a Denial-of-Service attack. This allows GridMAP to collect initial information about nodes on which no performance is available.

We now discuss how gSched works within the NGS "big picture" setup and then in detail.

4.4.1 Scheduling Procedure

gSched queries gLite WMS about the best sites for a particular job description. It also questions the GridMAP service about the anticipated network state for these sites. gSched then uses both responses to deliver input files to the remote site and to signal WMS to manage job execution. This is depicted in Figure 4.

Hence, gSched is able to formulate its own user-centric scheduling policy via the capacities of the the available software stack. It allows it to circumvent authentication and information retrieval difficulties (associated mainly with Globus GRAM), and to bypass common administrative restrictions (e.g. preventing net-

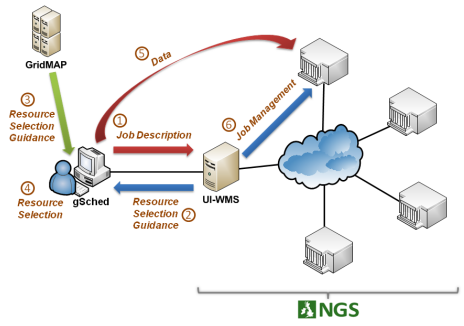


Fig. 4: Scheduling a job in the NGS infrastructure; using gSched introduces network-aware scheduling (steps 3–4)

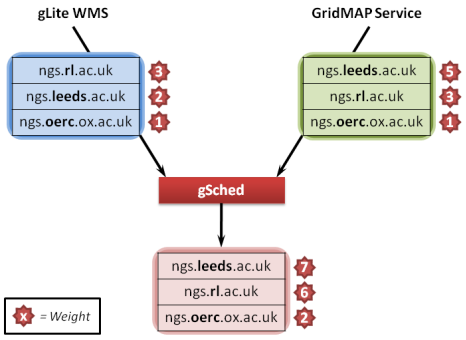


Fig. 5: An Example of Weighting Compatible Resources

work measurement). In effect, gSched is an application-level scheduler that effortlessly integrates GridMAP into NGS and similar grid deployments.

4.4.2 Consumption of GridMAP Information

WMS's response to an ordinary job description is an ordered list of grid sites that potentially satisfy the job's computational requirements. gSched assigns these sites weights according to their order, then passes this on to the GridMAP service. The GridMAP service returns a weighted list reordered according to the expected health of the candidate connections. Respective weights from both lists are then added to determine the best candidate as the one with the highest aggregate weight. This ensures that site selection is optimised for *both* computational and network status. Ties are settled based on GridMAP's rank. In the case of a negative weight provided by the GridMAP service, gSched starts a TCP handshake with the candidate site, obtains an RTT measurement and caches it, and then closes the connection.

An example is given in Figure 5: gSched concludes that the best site to schedule to is *ngs.leeds.ac.uk*. It proceeds to stage the input files directly to the user space on that site and, as a final step, augments the job description file with that chosen site as a requirement.

5 Evaluation

5.1 Evaluation Questions

The experiments aim to answer the following questions.

I. *How much can GridMAP enhance network utilisation?*
II. *How much can GridMAP improve scheduling?*
III. *How does GridMAP affect different grid applications?*
IV. *What is the cost of using GridMAP?*

Questions I and II both attempt to fundamentally assess the usefulness of employing GridMAP's analysis of passively-acquired network measurements to influence scheduling decisions. Question III sets out to investigate such improvements for different types of grid applications. The final objective is to inspect another impact of GridMAP by examining the overheads associated with it. Questions I-III are discussed in Section 5.2 while Question IV is addressed in Section 5.3.

It is important to note that we are not evaluating the accuracy of the prediction algorithm in isolation as this is not the focus of our work. Instead, we are concerned with the effect of network-awareness on grid scheduling. Our objective is to gauge the profit or forfeit, if any, of introducing network-awareness through the passive measurements provided by GridMAP. Comparing GridMAP against other network-aware systems is not possible due to their intrusive nature (as described in §2.1).

5.2 Evaluation Results

Three experiments were designed to answer Questions I, II and III. Each experiment involved submitting 100 jobs of one of the case studies (Table 1) using gSched with and without GridMAP.

Table 1: Summary of the Case Studies' Characteristics

Case Study	Sites	Input (MB)	Output (MB)	Approx. CPU (FLOP/Byte input)
AutoDock	3	2.83	0.08	43,000
PMEMD	2	5.20	0.08	728,000
LogStore	5	110.66	0.00006	2,600

The scenario of using gSched on its own renders it a non-intervening wrapper around the basic WMS functionality. Using gSched with GridMAP, represents a network-aware variant of WMS. We present the results of the 3 experiments under these 2 scenarios in addition to a *worst case scenario* where the site with the worst connection according to GridMAP is chosen. This is further highlights the effect of network-awareness on each application.

5.2.1 Network Management

Network utilisation is inspected to quantify how much GridMAP helps grid software in getting more out of the network by choosing sites with better connectivity. Pasinemda's measurements of RTT and achieved throughput are used for this purpose.

Fig. 6: Change in RTT across experiments

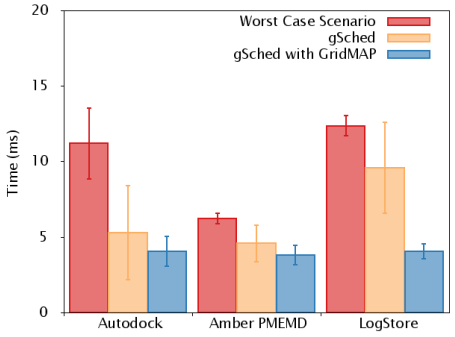
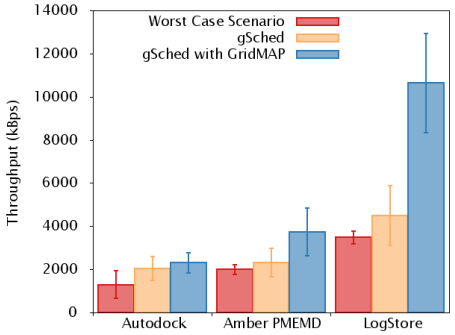


Fig. 7: Change in Throughput across experiments



Consistent with expectation, we observed a significant improvement in network utilisation brought by the use of GridMAP. Mean RTT measurements, portrayed in Figure 6, was reduced by 23.3%, 16.8% and 57.7% for AutoDock, PMEMD and LogStore, respectively. Mean throughput was increased by 12.4%, 60.8% and 136.6% for the respective experiments as depicted in Figure 7. The differences were tested using a paired samples t-test ($p < 0.005$). The differences were highly significant for all case studies. This improvement 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. We also computed the 99.5% confidence intervals for the two metrics to further test the reliability of our findings. They are presented in Table 2.

Table 2: 99.5% Confidence Intervals for Improvement in RTT and Throughput

Case Study	RTT (ms)	Ach. Thrpt. (kBps)
AutoDock	1.228 ± 0.642	227.74 ± 178.86
PMEMD	0.767 ± 0.401	1409.79 ± 361.43
LogStore	5.463 ± 1.041	6025.44 ± 828.63

Figures 6-7 depict the dispersion within each experiment as standard error bars. It would initially seem from comparing the standard error bars (especially in Figure 7) that variance deteriorates with the use of GridMAP. To fairly compare the result sets, dispersion of each set was normalised around its mean. Table 3 displays the values of the coefficient of variation (CV) for each result set computed using $CV = \sigma/\mu$.

Table 3: CV Values for RTT and Throughput

Case Study	RTT		Achieved Throughput	
	gSched	gSched +GridMAP	gSched	gSched +GridMAP
AutoDock	0.59	0.24	0.27	0.20
PMEMD	0.26	0.17	0.28	0.30
LogStore	0.31	0.12	0.31	0.22

By comparing the values in Table 3 we find that variability within the RTT and throughput measurement sets is distinctly reduced when GridMAP is used. This shows that overall GridMAP achieves more stable network utilisation, indicating less susceptibility to being affected by changes in the network. The only exception is for the PMEMD application where throughput CV is slightly more when using GridMAP.

In light of the case study characteristics outlined in Table 1, we find that the improvement in network utilisation is dependent on two application-related factors. The first factor is the number of sites which qualify as job execution theatres. As this number increases, there is more variance in network connection quality which

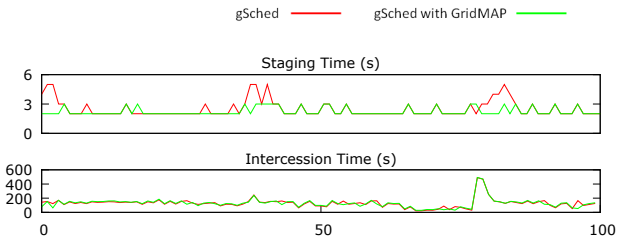
means more options for GridMAP to select from. This can be seen in the magnitude of RTT reduction across case studies. It is also noticeable in the dispersion in RTT and achieved throughput values. The second factor is the size of the application input files. The larger the input the greater the effect of GridMAP on improving network utilisation. This is evident from the contrast in throughput increase between the case studies considering that LogStore shifts more than 20 times as much input data as PMEMD and almost 40 times as much as AutoDock.

5.2.2 Scheduling Performance

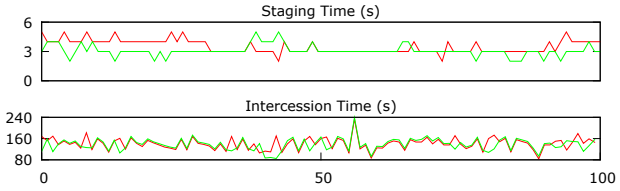
The metric we use to gauge scheduling performance is *staging time* which is the time taken by the scheduler to parse job specifications, find the right resources, prepare the job for execution, and perform housekeeping and accounting functions. In the context of our experiments, the staging time is used by gSched to accept a job description, query gLite WMS followed by GridMAP about resources, formulate a modified job description accordingly, and finally transfer input files. We also look at the *intercession time* which is the time between the user submitting the job and it starting to execute on the allocated resources.

Reducing staging time is the main potential advantage of adopting GridMAP, i.e. to match jobs to appropriate resources in less time. Decreasing intercession time is also important as it includes the amount of time spent in remote execution queues, which signifies the ability of the scheduler to match a job with resources that will tend to it quickly. Job completion time is typically used by HPC applications as a measure of scheduling performance. However, it is not considered in this evaluation due to the impossibility of distinguishing it using gLite WMS.

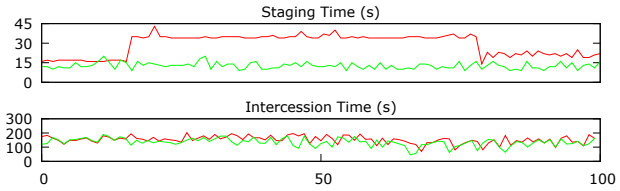
Figure 8 graphs the staging and intercession times recorded for the different jobs scheduled using gSched and gSched with GridMAP. Using GridMAP has a clear effect on reducing both the staging and intercession time, as summarised in Figure 9 depicting the overall change observed in staging time. Introducing GridMAP decreased mean staging time by 10.5%, 12.0% and 56.2% for AutoDock, PMEMD and LogStore, respectively. Again, these differences were found to be significant using paired samples t-test ($p < 0.005$) and the 99.5% confidence intervals, reported in Table 4, were calculated. Both tests demonstrate the unmistakable reduction in staging time when GridMAP is applied. We also notice that this reduction is proportional to the size of the input files to be transferred.



(a) Autodock



(b) Amber PMEMD



(c) LogStore

Fig. 8: Results With and Without GridMAP

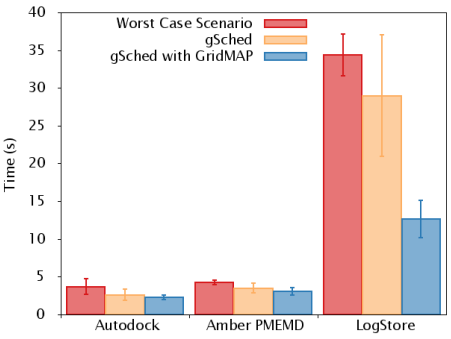


Fig. 9: Change in staging time across experiments

Table 4: 99.5% Confidence Intervals for Improvement in Staging Time

Case Study	Staging Time (s)
AutoDock	0.265 ± 0.211
PMEMD	0.423 ± 0.237
LogStore	16.300 ± 2.423

CV values for staging times are presented in Table 5. These show reduced CV when GridMAP is used, signifying how GridMAP introduces stability, across all

applications, into the scheduling process by allowing it to adapt to perceived and forecasted network changes.

Table 5: CV Values for Staging Time

Case Study	Staging Time	
	gSched	gSched with GridMAP
AutoDock	0.29	0.12
PMEMD	0.19	0.15
LogStore	0.28	0.19

It is important here to provide some context in order to appreciate the significance of such gains in real world grid applications. The staging times presented above are the average for executing a single job. Typical usage of such parametric applications incurs much higher job counts (1,000s-100,000s) in order to study the different possibilities, and hence any gain is a much needed advancement. For example, an expected use of AutoDock involves running 1,000s of docking jobs to subject a protein and its mutations to a very large number of compounds under varying conditions of temperature, pH, etc. WISDOM [46], a grid deployment of virtual screening tools for malaria-related research, submitted 31,231 AutoDock jobs consuming 50.5 CPU years over an effective period of 15 days. A similar deployment executed 77,504 jobs [47]. Other grid applications, including PMEMD, follow similar execution patterns which stress the importance of rapid job turnaround time.

5.3 GridMAP's Overheads

Based on the deployment detailed in Section 4.1, we present the associated storage, computational and network overheads in order to answer evaluation question IV which ascertains the feasibility of a real world GridMAP deployment over commodity hardware machines.

5.3.1 Storage Overhead

The mean storage cost of Pasinemda for the user machine scheduling onto the NGS was recorded as 2.48kB with a standard deviation of 1.16. For the 20 load-generating VMs, the mean cost was 58.86kB with a standard deviation of 45.71.

The GridMAP service's storage overhead corresponds to the total amount of space required to handle the database back-end, service front-end, and data analysis elements. 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 which required 4.87MB of disk per replica (19.48MB in total). This cost relies on a number of variables. The first variable is the size of the database which in turn depends on 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. The second variable is the amount of virtual memory required by the Akima splines and exponential smoothing algorithms, which are mostly null except at times of high load. The final variable is the degree of distribution and data replication exercised in the system. In this deployment, we distributed the service over 4 computers.

5.3.2 Computational Overhead

Table 6 presents the mean (μ) and standard deviation (σ) utilisation of CPU and dynamic RAM. Here, CPU utilisation is presented as a percentage of the capacity mentioned in Subsection 4.1.1.

Table 6: Computational Costs of the GridMAP System

Utilisation		Pasinemda	GridMAP Service	
			Idle	Running
CPU (%)	μ	0.0	0.4	2.3
	σ	0.1	0.3	29.1
Memory (MB)	μ	0.7	156.3	193.9
	σ	0.0	19.2	58.0

Pasinemda's overhead was observed across all user machines and was found to be low. We also find that the computational cost recorded every other minute of the GridMAP service and its related processes (such as MySQL, Tomcat, etc.) varies considerably between the states of *Idle*, i.e. average utilisation before the service is loaded, and *Running*, as measurement reports and queries are received. Nevertheless, these demands are hardly substantial even considering the moderate capability of the computers used to host the GridMAP service.

5.3.3 Network Overhead

We recorded the volume of network traffic generated by GridMAP to carry out its operations. Pasinemda's measurement reports were recorded on the user machine to be 0.16kB/min on average, while the instances on the 20 load-generating VMs were found to create an average of 1.78kB/min. The GridMAP service's network overhead is created by its internal maintenance operations, which includes traffic to administer data management,

load balancing, data replication and other processes. This was measured at 7.64kB/min on average.

6 Conclusions & Future Work

Motivated by a disparity between the network requirements of grid applications and the technologies that cater for them, this article put forward an argument for network-aware scheduling. We investigated the shortcomings of current solutions and proposed GridMAP as a mechanism to actualise efficient network-aware scheduling. We evaluated GridMAP using varied real world applications on the NGS, a UK production grid infrastructure. We developed a simple wrapper to integrate GridMAP into such a sophisticated ecosystem with very little effort, allowing us to circumvent the administrative restrictions of production grid deployments. We found that GridMAP enables better network management across all applications. Such dynamic network utilisation in turn engenders improved grid scheduling as reflected by reduced staging times. Across all applications, RTT was reduced by a factor between 16.8% and 57.7%, and throughput was increased by a factor between 12.4% and 136.6%. As a consequence, staging time was improved by a factor ranging from 10.5% to 56.2%. The magnitude of change in these metrics is dependant on two application-specific factors: input data size and fluctuation in connection quality between candidate grid sites. We also found that GridMAP reduced the variance in staging times reflecting better adaptation to changes in the network.

The article also presented the different costs of GridMAP, establishing the unobtrusive nature of Pasinemda as an effective tool to collect information from complex networked systems such as grids. This allows it to be easily deployed in various situations, including those with limited resources. Moreover, the overhead study concluded that the GridMAP service is deployable with fairly small disk, CPU, memory, and network footprints.

There are a number of avenues for future work. There is potential for horizontal development, e.g. numerical analysis plugins to provide further network performance prediction. There is also room for vertical expansion through collecting a wider range of network metrics, e.g. jitter and packet loss. More importantly, though, we are working on extending the GridMAP functionality to enhance network efficiency across cloud data centres. This could be extended further to manage network usage across different cloud service providers, requiring brokerage between potentially non interoperable APIs, a substantial undertaking.

Acknowledgements The authors thank Gordon S. Blair and Paul Grace for reviewing earlier versions, and Matthew Sperrin for advice on statistical significance. This work was partly funded by EC-GIN, a FP6 STREP project (contract number 045256).

References

1. Akella, A., Seshan, S., Shaikh, A.: An Empirical Valuation of Wide-Area Internet Bottlenecks. In: Proc. ACM SIGCOMM IMC, pp. 101–114 (2003)
2. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20122017. Tech. rep., Cisco Press (2013)
3. Lee, C., Lee, D.K., Yi, Y., Moon, S.: Operating a Network Link at 100%. In: Proc. PAM, pp. 1–10 (2011)
4. Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. The International Journal of Supercomputer Applications and High Performance Computing **11**(2), 115–128 (1997)
5. Berman, F.: The Grid: Blueprint for a New Computing Infrastructure, chap. High-Performance Schedulers, pp. 279–309. Morgan Kaufmann (1998)
6. Batista, D.M., da Fonseca, N.L.S.: A Survey of Self-Adaptive Grids. IEEE Communications Magazine **48**(7), 94–100 (2010)
7. Qureshi, M.B., Dehnavi, M.M., Min-Allah, N., Qureshi, M.S., Hussain, H., Rentifis, I., Tziritas, N., Loukopoulos, T., Khan, S.U., Xu, C.Z., et al.: Survey on grid resource allocation mechanisms. Journal of Grid Computing pp. 1–43 (2014)
8. Agarwal, A., Kumar, P.: Multidimensional QoS Oriented Task Scheduling in Grid Environments. International Journal of Grid Computing & Applications (IJGCA) **2**(1), 28–37 (2011)
9. Tomás, L., Caminero, A.C., Carrión, C., Caminero, B.: Network-aware meta-scheduling in advance with autonomous self-tuning system. Future Gener. Comp. Sys. **27**(5), 486–497 (2011)
10. Chinnaiah, V., Somasundaram, T.S.: A grid resource brokering strategy based on resource and network performance in grid. Future Gener. Comp. Sys. **28**(3), 491–499 (2012)
11. McClatchey, R., Anjum, A., Stockinger, H., Ali, A., Willers, I., Thomas, M.: Data Intensive and Network Aware (DIANA) Grid Scheduling. Journal of Grid Computing **5**(1), 43–64 (2007)
12. Pezaros, D., Sifalakis, M., Schmid, S., Hutchison, D.: Dynamic Link Measurements using Active Components. In: Proc. IFIP Conf. on Active Networks (2004)
13. He, Q., Dovrolis, C., Ammar, M.: On the Predictability of Large Transfer TCP Throughput. Computer Networks **51**(14), 3959–3977 (2007)
14. Lowekamp, B.B.: Combining Active and Passive Network Measurements to Build Scalable Monitoring Systems on the Grid. ACM SIGMETRICS Performance Evaluation Review **30**(4), 19–26 (2003)
15. Wolski, R.: Dynamically Forecasting Network Performance Using the Network Weather Service. Cluster Computing **1**(1), 119–132 (1998)
16. Vazhkudai, S., Schopf, J.M., Foster, I.T.: Predicting the Performance of Wide Area Data Transfers. In: Proc. IPDPS, p. 270 (2002)
17. Yousaf, M.M., Welzl, M., Junaid, M.M.: Fog in the Network Weather Service: A Case for Novel Approaches. In: Proc. Conf. on Networks for Grid Appl., pp. 1–6 (2007)

18. Goga, O., Teixeira, R.: Speed Measurements of Residential Internet Access. In: Proc. PAM, pp. 168–178 (2012)
19. Huang, A.C., Steenkiste, P.: Network-sensitive service discovery. Journal of Grid Computing **1**(3), 309–326 (2003)
20. Madhyastha, H.V., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., Venkataramani, A.: iPlane: An Information Plane for Distributed Services. In: Proc. USENIX OSDI, pp. 367–380 (2006)
21. Hullár, B., Laki, S., Stéger, J., Csabai, I., Vattay, G.: SONoMA: A Service Oriented Network Measurement Architecture. Tech. Rep. TR-ELTE-CNL-2010/1, Eötvös Loránd University, Hungary (2010)
22. Wenwei, L., Dafang, Z., Jinmin, Y., Gaogang, X.: On Evaluating the Differences of TCP and ICMP in Network Measurement. Computer Communications **30**(2), 428–439 (2007)
23. Paxson, V.E.: Measurements and Analysis of End-to-End Internet Dynamics. Ph.D. thesis, University of California at Berkeley (1998)
24. Ganguly, A., Agrawal, A., Boykin, P.O., Figueiredo, R.J.: WOW: Self-Organizing Wide Area Overlay Networks of Virtual Workstations. In: Proc. IEEE HPDC, pp. 30–42 (2006)
25. Raicu, I., Dumitrescu, C., Ripeanu, M., Foster, I.: The design, performance, and use of diperf: An automated distributed performance evaluation framework. Journal of Grid Computing **4**(3), 287–309 (2006)
26. Zhang, X., Freschl, J.L., Schopf, J.M.: Scalability Analysis of Three Monitoring and Information Systems: MDS2, R-GMA, and Hawkeye. J of Parallel and Distributed Computing **67**(8), 883–902 (2007)
27. Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., Tuecke, S.: A Directory Service for Configuring High-Performance Distributed Computations. In: Proc. IEEE HPDC, pp. 365–375 (1997)
28. Cooke, A.W., Gray, A.J., Nutt, W., Magowan, J., Oevers, M., Taylor, P., Cordenonsi, R., Byrom, R., Cornwall, L., Djaoui, A., et al.: The relational grid monitoring architecture: Mediating information about the grid. Journal of Grid Computing **2**(4), 323–339 (2004)
29. Tierney, B., Aydt, R., Gunter, D., Smith, W., Swany, M., Taylor, V., Wolski, R.: A Grid Monitoring Architecture. Tech. rep., Global Grid Forum (2002)
30. Zanikolas, S., Sakellariou, R.: A taxonomy of grid monitoring systems. Future Gener. Comp. Sys. **21**(1), 163 – 188 (2005). DOI 10.1016/j.future.2004.07.002
31. Lee, S.: eTOP: End-to-end Performance Measurement Infrastructure on KOREN and APAN Links. `http://master.apan.net/meetings/busan03/measurement/koren.ppt` (2003)
32. Sacerdoti, F.D., Katz, M.J., Massie, M.L., Culler, D.E.: Wide Area Cluster Monitoring with Ganglia. In: Proc. IEEE CLUSTER, pp. 289–298 (2003)
33. Al-Ali, R.J., Amin, K., Von Laszewski, G., Rana, O.F., Walker, D.W., Hategan, M., Zaluzec, N.: Analysis and provision of qos for distributed grid applications. Journal of Grid Computing **2**(2), 163–182 (2004)
34. Elkhatib, Y.: Monitoring, Analysing and Predicting Network Performance in Grids. Ph.D. thesis, Lancaster University (2011)
35. Elkhatib, Y., Edwards, C.: A Survey-based Study of Grid Traffic. In: Proc. Conf. on Networks for Grid Applications (2007)
36. Savage, S.: Sting: A TCP-based network measurement tool. In: Proc. of USENIX Symposium on Internet Technologies and Systems, vol. 2 (1999)
37. Horneffer, M.: Assessing Internet Performance Metrics Using Large-Scale TCP-SYN based Measurements. In: Proc. PAM (2000)
38. Jiang, H., Dovrolis, C.: Passive Estimation of TCP Round-Trip Times. ACM SIGCOMM Computer Communication Review **32**(3), 75–88 (2002)
39. Elkhatib, Y., Edwards, C., Mackay, M., Tyson, G.: Providing Grid Schedulers with Passive Network Measurements. In: Proc. ICCCN (2009)
40. Network Measurement Working Group. `http://nmwg.internet2.edu`
41. Akima, H.: A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. J. of the ACM **17**(4), 589–602 (1970)
42. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An Integrated Experimental Environment for Distributed Systems and Networks. In: Proc. USENIX OSDI, pp. 255–270 (2002)
43. Marco, C., Fabio, C., Alvise, D., Antonia, G., Francesco, G., Alessandro, M., Moreno, M., Salvatore, M., Fabrizio, P., Luca, P., et al.: The gLite workload management system. In: Advances in Grid and Pervasive Computing, pp. 256–268. Springer (2009)
44. AutoDock, the scripps research institute. `http://autodock.scripps.edu/`
45. Case, D.A., Cheatham, T.E., Darden, T., Gohlke, H., Luo, R., Merz, K.M., Onufriev, A., Simmerling, C., Wang, B., Woods, R.J.: The Amber Biomolecular Simulation Programs. Journal of Computational Chemistry **26**(16), 1668–1688 (2005)
46. Jacq, N., Salzemann, J., Jacq, F., Legré, Y., Medernach, E., Montagnat, J., Maaß, A., Reichstadt, M., Schwichtenberg, H., Sridhar, M., Kasam, V., Zimmermann, M., Hofmann, M., Breton, V.: Grid-enabled Virtual Screening Against Malaria. J of Grid Comp. **6**(1), 29–43 (2008)
47. Kasam, V., Salzemann, J., Jacq, N., Maaß, A., Breton, V.: Large Scale Deployment of Molecular Docking Application on Computational Grid infrastructures for Combating Malaria. In: Proc. CCGrid, pp. 691–700 (2007)