On the Interplay Between Self-adaptation and Energy Efficiency

Soyaba Hasan and Yehia Elkhatib

School of Computing Science, University of Glasgow, UK.

Contributing authors: *first.last*@glasgow.ac.uk;

Abstract

Self-adaptation is an increasingly popular approach for enabling systems to make agile and data-driven decisions without human intervention. However, techniques for measuring the performance of a self-adaptive system and its effects on context are lacking. In this study, we present a mechanism that enables customizable and comprehensive evaluation of self-adaptive systems in real time. We apply our proposal to an example of resource management in a data center setting. We find that a self-adaptive approach to managing resources is multiple orders of magnitude better in terms of energy efficiency, especially as the size of the infrastructure grows, at the expense of slightly lower performance ($<$ **10%**).

Keywords: Self-adaptation, Performance measurement, Energy efficiency

1 Introduction

A self-adaptive system can handle changes without external intervention. This topic has been of great interest for many years, dating back to the notion of autonomic computing [1]. Self-adaptation is used in various domains such as self-driving vehicles, smart grids, mobile applications, the Internet of Things, and cloud computing. Self-adaptive system architectures are predominantly employed for their ability to make prompt decisions informed by prior experience, reduce room for human error and bias, and optimize resource utilization.

Despite the great interest in self-adaptive systems, there is a lack of means to understand their behavior in a real-time and comprehensive manner [2]. Evaluating the effectiveness of self-adaptive systems is a significant challenge due to their inherent complexity, dynamics, and unpredictable operating environments. In particular,

real-time feedback is essential for intended users and software engineers to leverage the capabilities of a system and the possibilities of its context, while also addressing potential limitations.

Despite the availability of runtime measures for self-adaptive systems, their adequacy for the above needs is questionable. Existing methods can collect real-time data on system behavior, but they are (i) overly complex and challenging to deploy, necessitating significant time and resources; and/or (ii) inadequate to reflect and analyze the effectiveness or provide actionable insights.

In this study, we devise a mechanism to evaluate in real-time the runtime performance of a self-adaptive job dispatching system. This is a widely used application of self-adaptation to manage servers in cloud data centers. In doing so, we aim to develop appropriate techniques that could be applied to other self-adaptive systems. Moreover, we investigate the interplay introduced by self-adaptation, specifically in terms of system performance and energy efficiency.

Our contributions are as follows:

1. We developed a mechanism to evaluate the performance and resource utilization of a self-adaptive system in real-time.
2. We apply this mechanism by integrating it with the Simdex [3] framework.
3. We investigate the trade-off between system performance and overhead when self-adaptation is used for resource management.

2 Problem Space

In this section, we describe the problem space and examine related works.

2.1 Background

A self-adaptive system can autonomously adapt in response to a change, such as a modification in the context in which the application runs or one in the fabric of the application itself. In either case, adaptation is performed programmatically based on a feedback mechanism, such as a Monitor, Analyze, Plan, Execute, and Knowledge (MAPE-K) control loop. Reacting (or sometimes even proacting) in such a programmatic manner enables applications to act in an agile and data-centric manner, which makes self-adaptive systems much more reliable, flexible, and efficient.

Self-adaptation has been applied to enable independent systems to evolve the way in which they interact with other systems, *e.g.,* recommendations in decentralized overlays [4] and automating system composition [5, 6] in smart home [7] and Internet of Vehicles [8] environments. Self-adaptive methods have also been employed to make centralized optimization decisions. A very common example of this is used in data centers, where decisions such as virtual machine consolidation [9], placement [10], and auto-scaling [11] can be made using self-adaptive control loops.

2.2 Related Work

The Rainbow framework [12] is commonly used to evaluate software quality. Although it provides a general strategy for quality evaluation, it does not offer guidance on

applying appropriate methods to reflect system performance. RELAX [13] supports model-based runtime verification and provides a comprehensive framework for the development of evaluation tools. It combines statistical techniques to build robust tools that can adapt to changing environmental conditions. However, this can result in high computational overhead for large-scale systems and may cause increased resource utilization and downtime. Fusion [14] employs dynamic and static verification techniques to detect errors in self-adaptive software systems. It generates a complete model of the system behavior and compares it with the actual performance during runtime. Similar systems were devised to gather knowledge of self-adaptive volunteer computing platforms [15]. However, they rely on the accuracy of static models, which may not accurately reflect performance, and cannot handle unexpected runtime behavior.

The Dynamico framework [16] utilizes formal methods to verify the behavior of self-adaptive systems at runtime. However, it relies on static models that may not accurately capture the behavior of the system and incur high computational and resource costs. SimuLizar [17] used a model-driven approach to analyze performance at design time but neglected uncertainty at runtime. Zanshin [18] combined machine learning and knowledge-based techniques to monitor complex systems under fluctuating conditions and optimize system behavior. However, maintenance of Zanshin and similar frameworks (*e.g.,* [19]) requires specialized expertise. Consequently, they are not well suited for smaller systems that do not feature advanced levels of self-adaptation.

In summary, as indicated by Gerostathopoulos et al. [2], most evaluation methods for self-adaptive systems lack validity and verification. Concretely, these methods often fail to consider the dynamic nature of such systems and lack specific guidance on applying appropriate evaluation strategies according to system design. Our work aims to address this gap in the literature.

3 Design

Our aim is to support users by developing an evaluation of self-adaptive systems. In this section, we introduce our design and use case.

3.1 Requirements

The functional requirements of our design are to:

- enable concurrent tests, *i.e.,* to compare different evaluation scenarios;
- support multiple performance metrics for users to acquire a complete picture of the system behavior and performance;
- provide a user-friendly interface that simplifies test setup, execution, and analysis;
- handle large-scale data and remain responsive under diverse workloads; and
- store and retrieve measurement logs.

Non-functional requirements are as follows:

- reliability even under heavy workloads, to ensure users can efficiently analyze results and identify patterns;
- handle and analyze information in real-time with minimal delays;
- handle multiple concurrent experiments and promptly;
- retrieve data effortlessly;

- extensible for the integration of new features and performance metrics; and
- easy to maintain and support with clear documentation and code organization.

Goals that are outside the scope of our system but are included in this manuscript:

- process and analyze large amounts of data to extract insights; and
- generate and customize visualizations.

3.2 Simdex Environment

Before explaining the measurement mechanism, we introduce the environment in which it operates. Simdex [3] is a tool for simulating self-adaptive job-dispatching systems. It is designed to replay workload logs and use self-adaptive mechanisms to experiment with enhancing system performance in response to changing conditions.

We chose Simdex because of its scalability and flexibility as a job-dispatching system that supports self-adaptive control loops. Its configurable and extendable features make it a robust tool to incorporate our evaluation system. Thereupon, users can customize the system configurations, manage job requests, optimize resource allocation, and analyze dynamic settings using our evaluation system. In contrast, other self-adaptive workload operation tools such as DEECo [20] and RTX [21] are substantially more complex in terms of integrating custom measurement mechanisms.

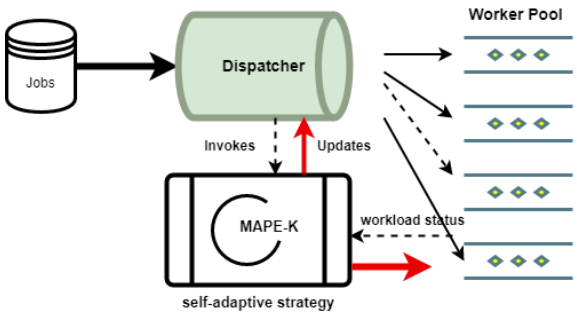

Fig. 1 An overview of the architecture of the Simdex simulation environment.

As illustrated in Figure 1, a user-submitted job is passed on to the dispatcher module that examines the job and assigns it to an appropriate worker. Each worker has a single queue and processes jobs in a first-come-first-served order. A worker's status can be either 'active' or 'inactive', which is altered based on workload demand.

If a self-adaptive (SA) strategy is being examined, the dispatcher triggers the MAPE-K loop self-adaptive strategy to identify the state of the worker nodes, ensuring that the system can adapt and respond to changes in workload. The self-adaptive controller then updates the dispatcher with the feedback decision, and the dispatcher directs jobs to the worker nodes responsible for processing them. This adaptive process adheres to two specific rules:1) it activates an inactive queue if any queue in the pool still has more than one job, and 2) when a worker does not have jobs queued, the system deactivates the idle queue.

In contrast, a non-self-adaptive (NSA) strategy is studied using a fixed number of active workers, whereby resources are statically allocated to submitted jobs.

3.3 Monitoring Mechanism

Through integration with Simdex, our evaluation system captures real-time performance metrics including latency, throughput, average response time, total delay, resource utilization, and power consumption. These metrics provide the user with a comprehensive understanding of the behavior and performance of the MAPE-K loop under study. Our measurement logic is implemented in Python and paired with custom scripts that use libraries such as `pandas` and `matplotlib` to automate analysis and visualization.

We designed the evaluation process following the principles outlined in the framework of Villegas et al. [22] and identified the metrics (detailed in Section 4.1) to develop a robust evaluation strategy. Users can customize the evaluation system to suit their requirements. For example, they can investigate different infrastructure configurations and scaling behaviors under heavy workloads. Users obtain real-time performance insights through automatic data visualizations that are made available.

4 Evaluation

We conducted a series of experiments using a range of worker configurations and workloads. We now describe the evaluation setup and results in detail.

4.1 Metrics

We are interested in assessing the costs and benefits of applying self-adaptive strategies to optimize resource and energy utilization. Thus, we measured the following variables, inspired by the catalog built by Da Silva et al. [23]:

- **Throughput** is the rate at which the system can process jobs. Higher values indicate better performance.
- **Latency** refers to the time it takes for a job to be processed by the system. Low latency is an indication of higher performance and improved user experience.
- **Total delay** is the total amount of time it takes for a job to be completed, from the time it was submitted to the system until the time it is finished. It includes the time spent waiting in queues, processing by workers, and any other delays that may occur during job execution.
- **Average Response time** is a measure of system responsiveness. It is equal to the amount of time taken for a system to respond to a job request.
- **Resource utilization**, a fundamental metric in job-dispatching systems, refers to the fraction of available resources that were used to complete a given set of tasks.
- **Power consumption** refers to the amount of energy consumed by the system over a period of time. Reducing this expenditure can result in improved energy efficiency, which is important particularly in resource-constrained environments.

4.2 Independent and Confounding Variables

Our independent variable in each experiment was the number of jobs, which varied within the range of 15,000–30,000. We vary the worker configuration between 2 and 14, changing their status from 'inactive' and 'active' according to the scaling strategy.

4.3 Results

We now describe the findings of our experiments in detail.

4.3.1 Overhead

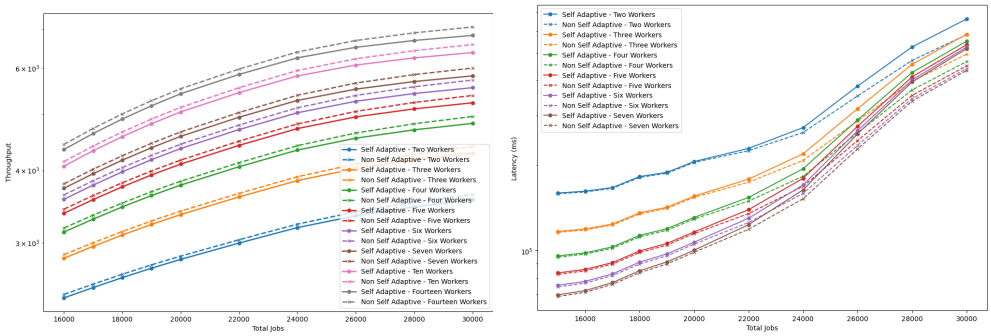

Fig. 2 The throughput as the number of jobs served per second (left), and job latency in milliseconds (right). The plots depict the magnitude of the additional overhead incurred by applying the self-adaptive resource management strategy. The impact on performance is discernible yet minimal.

We found that SA consistently underperforms when compared to NSA for all job ranges and worker configuration scenarios. This is manifested in the form of a 2.11% lower throughput and 10.74% higher latency levels, as depicted in the plots of Figure 2. The disparity between the two systems becomes more prominent as the number of jobs increases; specifically, the gap widens for settings of 24,000 jobs and above. This is to be expected, as self-adaptation adds a layer of complexity that includes introspection and decision making, which adversely influences system performance.

4.3.2 User-perceived Responsiveness

We make several observations from the plots in Figure 3. First, the SA variant of the system is as responsive as the NSA cognate for low job numbers. Second, SA becomes decreasingly responsive beyond 20,000 jobs. Third, the disparity between SA and NSA rapidly worsens with more jobs, as the former features an exponential growth in delay and response times, while the latter follows a linear trend with a low slope. Finally, the disparity worsens as the size of the infrastructure (*i.e.,* number of workers) increases.

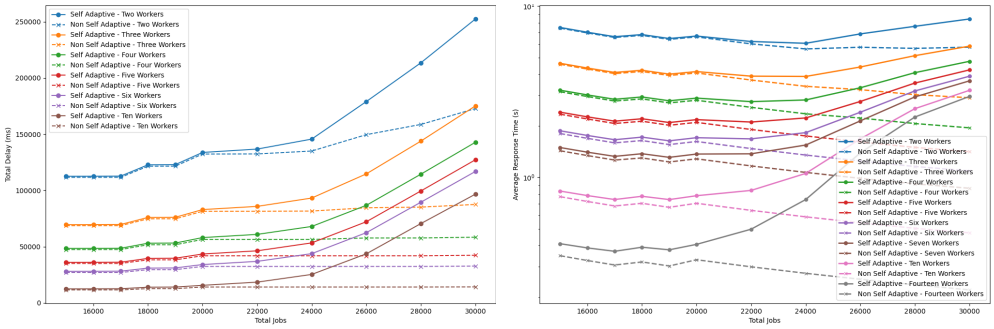

Fig. 3 The total delay (left) and response time (right) for both strategies. These results show that a self-adaptive strategy is feasible, but only to a point where an increased number of jobs and/or workers bring about exponentially degrading responsiveness.

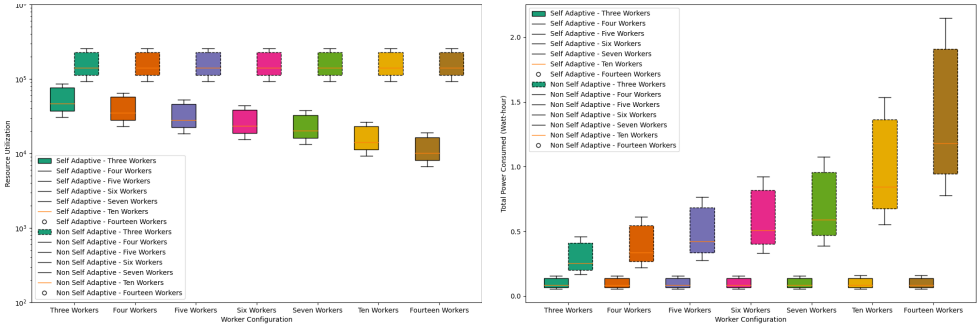

Fig. 4 The levels of resource utilization (left) refers to the ratio of the total uptime to the resources used to process the jobs; note the logarithmic scale. The corresponding power consumption (right) is given in Watt-hours. Inspecting these findings, we identify the benefits gained by adopting a self-adaptive management strategy: it allows much more efficient utilization of available resources as the infrastructure expands, and it minimizes energy consumption owing to the proactive and dynamic management of running workers.

4.3.3 Optimization

The utilization levels of NSA were comparatively higher than those of SA, both in terms of making efficient use of the available computational resources and the amount of energy consumed. With self-adaptation, resource utilization decreases by 84.35% with an increasing number of workers in the 15,000–30,000 job range. In Figure 4, the box plots for SA show a clear trend of decreasing resource utilization with an increasing number of workers, whereas the box plots for NSA are relatively constant.

Furthermore, the power consumption of SA is relatively consistent across *all* job ranges and worker configurations. Overall, this level is 86.84% that of NSA. In contrast, NSA exhibits a noticeable rise in power consumption as the number of jobs and workers increases, primarily because idle workers consume energy unnecessarily. This suggests that SA can adjust its resource allocation to optimize utilization and improve efficiency,

while NSA cannot do so. This is evident in unequal job allocation, wherein some workers are overwhelmed while others are underutilized. This results in inadequate resource utilization and increased power consumption.

5 Discussion and Concluding Remarks

The findings presented here represent a real-world manifestation of an interesting trade-off when it comes to employing self-adaptive approaches, which is at the core of this work: performance versus resource utilization.

In our experiments, we demonstrated how a self-adaptive resource management strategy allows the system to process tasks efficiently using only the necessary number of workers at any given time. This further supports the claim that self-adaptive resource management offers a significant advantage over non-adaptive alternatives. However, such improvements come at the cost of reduced application performance, as evidenced by throughput and latency results. These compromises must be considered when deciding whether a self-adaptive strategy should be employed in a given context. For example, in contexts where there is only a small infrastructure of workers or where real-time responsiveness is of high importance, using a self-adaptive strategy might be excessive and harm the quality of experience, thus being an inappropriate approach. However, a self-adaptive strategy would be an efficient and cost-effective solution for resource allocation in dynamic environments or those with a large optimization space, that is, where a manual/static scheduling strategy would need to consider a large number of factors.

Our future work will focus on the application of this approach to more dynamic environments, namely, the scheduling of data processing events in edge infrastructures. Such infrastructures are liable to sudden and dramatic changes; hence, agile and data-driven decision-making would potentially be of great benefit.

Acknowledgments. This work was partly supported by the UK EPSRC under grant number EP/R010889/2.

References

[1] IBM: An architectural blueprint for autonomic computing (4th edition). Technical Report G507-2065-00, IBM, 17 Skyline Drive, Hawthorne, NY 10532, USA (June 2006)

[2] Gerostathopoulos, I., Vogel, T., Weyns, D., Lago, P.: How do we evaluate self-adaptive software systems?: A ten-year perspective of seams. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 59–70 (2021). https://doi.org/10.1109/SEAMS51251.2021.00018

[3] Kruliš, M., Bureš, T., Hnětynka, P.: Simdex: A simulator of a real self-adaptive job-dispatching system backend. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 167–173 (2022). https://doi.org/10.1145/3524844.3528078

[4] Frey, D., Kermarrec, A.-M., Maddock, C., Mauthe, A., Roman, P.-L., Taïani, F.: Similitude: Decentralised adaptation in large-scale P2P recommenders. In: Distributed Applications and Interoperable Systems (DAIS), pp. 51–65 (2015)

[5] Nundloll, V., Elkhatib, Y., Elhabbash, A., Blair, G.S.: An ontological framework for opportunistic composition of IoT systems. In: International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), pp. 614–621 (2020)

[6] Elhabbash, A., Nundloll, V., Elkhatib, Y., Blair, G.S., Sanz Marco, V.: An ontological architecture for principled and automated system of systems composition. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 85–95 (2020). https://doi.org/10.1145/3387939.3391602

[7] Wang, Z., Elkhatib, Y., Elhabbash, A.: HolonCraft – an architecture for dynamic construction of smart home workflows. In: Conference on Future Internet of Things and Cloud (FiCloud), pp. 213–220 (2022). https://doi.org/10.1109/FiCloud57274.2022.00036

[8] Elhabbash, A., Elkhatib, Y., Bouloukakis, G., Salama, M.: A middleware for automatic composition and mediation in IoT systems. In: International Conference on the Internet of Things (IoT), pp. 127–134 (2022). https://doi.org/10.1145/3567445.3567451

[9] Najafizadegan, N., Nazemi, E., Khajehvand, V.: A MAPE-K loop based model for virtual machine consolidation in cloud data centers. Journal of Computer & Robotics **13**(2), 33–60 (2020)

[10] Elhabbash, A., Elkhatib, Y.: Energy-aware placement of mediation services in IoT systems. In: International Conference on Service-Oriented Computing (ICSOC), pp. 335–350 (2021). https://doi.org/10.1007/978-3-030-91431-8_21

[11] Sliem, M., Salmi, N., Ioualalen, M.: Performance analysis of self-adaptive policies in containerized microservices. In: International Conference on Engineering and Emerging Technologies (ICEET) (2021). https://doi.org/10.1109/ICEET53442.2021.9659601

[12] Cheng, S.-W., Garlan, D., Schmerl, B.: Evaluating the effectiveness of the rainbow self-adaptive system. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 132–141 (2009). https://doi.org/10.1109/SEAMS.2009.5069082

[13] Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.-M.: RELAX: Incorporating uncertainty into the specification of self-adaptive systems. In: International Requirements Engineering Conference (RE), pp. 79–88 (2009). https://doi.org/10.1109/RE.2009.36

[14] Elkhodary, A., Esfahani, N., Malek, S.: FUSION: A framework for engineering

self-tuning self-adaptive software systems. In: Symposium on Foundations of Software Engineering (FSE), pp. 7–16. ACM, New York (2010). https://doi.org/10.1145/1882291.1882296

[15] Elhabbash, A., Bahsoon, R., Tino, P.: Self-awareness for dynamic knowledge management in self-adaptive volunteer services. In: International Conference on Web Services (ICWS), pp. 180–187 (2017). https://doi.org/10.1109/ICWS.2017.31

[16] Tamura, G., Villegas, N.M., Muller, H.A., Duchien, L., Seinturier, L.: Improving context-awareness in self-adaptation using the DYNAMICO reference model. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 153–162 (2013). https://doi.org/10.1109/SEAMS.2013.6595502

[17] Becker, M., Becker, S., Meyer, J.: SimuLizar: Design-time modeling and performance analysis of self-adaptive systems. In: Kowalewski, S., Rumpe, B. (eds.) Software Engineering, pp. 71–84. Gesellschaft für Informatik e.V., Bonn (2013)

[18] Tallabaci, G., Silva Souza, V.E.: Engineering adaptation with Zanshin: An experience report. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 93–102 (2013). https://doi.org/10.1109/SEAMS.2013.6595496

[19] Sobhy, D., Minku, L., Bahsoon, R., Chen, T., Kazman, R.: Run-time evaluation of architectures: A case study of diversification in IoT. Journal of Systems and Software **159** (2020) https://doi.org/10.1016/j.jss.2019.110428

[20] Kit, M., Gerostathopoulos, I., Bures, T., Hnetynka, P., Plasil, F.: An architecture framework for experimentations with self-adaptive cyber-physical systems. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 93–96 (2015)

[21] Schmid, S., Gerostathopoulos, I., Prehofer, C., Bures, T.: Self-adaptation based on big data analytics: A model problem and tool. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 102–108 (2017). https://doi.org/10.1109/SEAMS.2017.20

[22] Villegas, N., Müller, H., Tamura, G., Duchien, L., Casallas, R.: A framework for evaluating quality-driven self-adaptive software systems. In: Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 80–89 (2011). https://doi.org/10.1145/1988008.1988020

[23] Silva, M.B., Bezerra, C., Coutinho, E., Maia, P.H.: A catalog of performance measures for self-adaptive systems. In: The Brazilian Symposium on Software Quality (SBQS). ACM, New York (2021). https://doi.org/10.1145/3493244.3493259