Optimization of Component Allocations in Middleware Platforms using Performance Models

Felix Willnecker

fortiss GmbH – An-Institut Technische Universität München
Guerickestr. 25, 80805 München, Germany
willnecker@fortiss.org

Abstract: Distributed enterprise applications are typically implemented as system-of-systems composed of components and linked via middleware. These systems often utilize corresponding resources far below available capacity. In order to increase resource utilizations the consolidation of components demands several tests on environments comparable to the production system. Performance models can be used to landscape such system architectures and to simulate changes in the component topology or resource environment without harming production systems. Therefore, this work aims at extracting performance models from distributed middleware platforms. Based on these models, an architecture optimizer is built to test different allocation topologies. Subsequently, the optimized model is simulated and the prediction accuracy of architecture changes is evaluated in this work. This allows architects to evaluate component changes and topology variations without a replica of the production system.

1 Motivation and Purpose

Middleware based distributed system-of-systems architectures are state of the art in large scale enterprise applications [BVD+14]. These systems are composed of components that can be moved and duplicated from one instance to another using a deployment management software [Woo09]. The placement of components is a complicated task that today is merely assisted by logical topology recommendations [Woo09, KKR11]. These recommendations can work as guidelines but cannot answer the questions on how to size the target environment for a specific deployment unit and how to optimize the topology to a certain optimization goal.

Logical topologies utilize the hardware below their possible capacity as the average load of data centers today is under 25% [PVR12]. Virtualized server environments already reduce this overprovisioning thus, increasing the hardware utilization [SB10]. However, virtualized server environments limit the optimization opportunities to the granularity level of single virtual machines. Middleware systems rely on more fine-grained deployment units, allowing operation engineers and architects to utilize unused capacity more efficiently. Planning and testing such changes in productive environments comprises risks for the stability. In addition, productive alike test environments and productive systems have comparable prices. Furthermore, such environments are usually used to capacity by

several projects executing load tests.

Simulations on performance models can optimize performance metrics, license, hardware, and energy costs by evaluating alternative topologies without tests in the productive environment [KKR11, BVK13]. This thesis extracts performance models from middleware platforms in order to simulate component allocation optimizations and analyze the impact of changing or introducing components to this environment. This work investigates the interfaces of distributed Java EE components to build performance models for middleware platforms. An optimizer based on this model completes the work of this thesis.

2 Research Questions and Approach

This section provides an overview of the main research questions (RQ) and how this dissertation tries to answer them.

RQ1: Which architecture-level performance model extraction techniques and performance monitoring solutions are applicable for middleware platforms?
Automatic performance model extraction technologies and resource estimation approaches have been proposed to the scientific community [BVK13] [SCZK14]. The proposed approaches focus on single applications and partly ignore the distribution aspect of modern applications. The available model generators disregard the component allocation aspect and mainly focus on the software components, their resource demands, and relationships. Research on complete middleware environments including the deployment locations for distributed systems has not been conducted.

This change of scope requires reconsidering the level of granularity feasible for the performance model generation. Currently available approaches generate fine grained performance models and detect component dependencies that are not available through a public interface [BVK13]. Computing power and a processable complexity level limit the granularity of such performance models. The granularity decision also depends on the entailed monitoring solutions. Industry as well as scientific solutions are available to monitor middleware platforms [vHWH12, Gre11].

Selecting an appropriate monitoring solution, choosing the right level of granularity for the performance models, as well as adapting and extending available performance model generators are the main challenges for this research question. This work will identify, evaluate, and synthesize available monitoring solutions and model generators based on a literature review. Based on this review, we will select applicable monitoring solutions and conduct a series of controlled experiments according to the Design Science Methodology using SPEC benchmarks [Hen06]. These experiments identify the combination of monitoring and model generator solutions that are best suited for middleware based distributed systems.

RQ2: Which architecture optimization approaches can be adapted for architecture-level performance models extracted from middleware platforms?
This research question is settled between the research domains of architecture optimization and self-adaptive software systems and tries to adapt and extend available optimization

algorithms to optimize the component allocations of a productive environment [KKR11, ST09].

Architecture optimizations have been proposed to improve logical component topologies and support architecture decisions during design time [KKR11]. Especially component allocation decisions are a complex and time consuming activity [KKR11]. Unfortunately, during design time resource demands and the infrastructure hosting the application can only be estimated. The accuracy of the model as well as the optimization recommendations depend on these assumptions.

Optimization of running systems is covered in the domain of self-adaptive software systems [ST09]. These systems act automatically to a certain degree and can react within a limited set of rules. These rules can include increasing the number of available servers to compensate an unusually high amount of user requests or detecting security breaches and close certain system accesses [ST09]. To detect the need of action self-adaptive software monitors performance data like hardware utilization and network throughput. This data is used to reason the application's structure and to react in order to keep the system stable, reliable, and secure.

Research from the domain of self-adaptive systems as well as from design time architecture optimizations is leveraged to build component allocation recommendations with architecture-level performance models. The contribution of this research question is a component allocation optimizer, which utilizes performance models from the experiment in RQ1. The optimizer reuses and extends algorithms from the field of architecture optimization with performance models extracted from a running system. Subsequently, the results are compared to improvements that self-adaptive systems can achieve. This comparison is conducted as a controlled experiment according to the Design Science Methodology and aims to demonstrate an improvement to the available approaches from both research domains.

RQ3: What is the accuracy of simulated optimization results compared to real system measurements?
Evaluating the proposed approach requires three steps: Validating the accuracy of the generated architecture-level performance models, the change prediction capabilities, and the selected optimization algorithm.

The evaluation of the performance model is conducted by a controlled experiment. Hence, we generate such a model from a real distributed middleware based application. This generated model is then used to simulate different load and usage scenarios. The same scenarios are processed on the real system. When processing such test runs, performance metrics like throughput, response time or utilization are measured and compared with the simulation results. The accuracy of the model is assessed based on the error among simulation results and measurements.

Estimates of the influence of changing components or newly introduced components are also validated with an experiment. Simulations of these changes are compared with real changes to the application. The change effect prediction is essential to demonstrate that architecture changes can be simulated with the generated performance model.

The last step is the evaluation of the optimization. The optimization algorithm calcu-

lates alternative component topologies by predicting component allocation changes. The topologies are optimized in terms of performance metrics, hardware, license, or energy costs. The optimization algorithm must prove that the proposed component topology improves according to the optimization goal(s).

References

[BVD+14] Andreas Brunnert, Christian Vögele, Alexandru Danciu, Matthias Pfaff, Manuel Mayer, and Helmut Krcmar. Performance Management Work. *Business & Information Systems Engineering*, 6(3):177–179, 2014.

[BVK13] Andreas Brunnert, Christian Vögele, and Helmut Krcmar. Automatic Performance Model Generation for Java Enterprise Edition (EE) Applications. In *Computer Performance Engineering*, volume 8168 of *Lecture Notes in Computer Science*, pages 74–88. Springer Berlin Heidelberg, 2013.

[Gre11] Bernd Greifeneder. Method and system for processing application performance data ouside of monitored applications to limit overhead caused by monitoring, June 7 2011. US Patent 7,957,934.

[Hen06] John L. Henning. SPEC CPU2006 Benchmark Descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, September 2006.

[KKR11] Anne Koziolek, Heiko Koziolek, and Ralf Reussner. PerOpteryx: Automated Application of Tactics in Multi-objective Software Architecture Optimization. In *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*, QoSA-ISARCS '11, pages 33–42, New York, NY, USA, 2011. ACM.

[PVR12] M. Pawlish, A.S. Varde, and S.A. Robila. Analyzing utilization rates in data centers for optimizing energy management. In *Green Computing Conference (IGCC), 2012 International*, pages 1–6, June 2012.

[SB10] B. Speitkamp and M. Bichler. A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers. *Services Computing, IEEE Transactions on*, 3(4):266–278, Oct 2010.

[SCZK14] Simon Spinner, Giuliano Casale, Xiaoyun Zhu, and Samuel Kounev. LibReDE: A Library for Resource Demand Estimation. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ICPE '14, pages 227–228, New York, NY, USA, 2014. ACM.

[ST09] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive Software: Landscape and Research Challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

[vHWH12] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ICPE '12, pages 247–248, New York, NY, USA, 2012. ACM.

[Woo09] Bobby Woolf. WebSphere SOA and JEE in Practice, 2009.