Performance Modeling in the Age of Big Data

Some Reflections on Current Limitations

Robert Heinrich
Software Design and Quality
Karlsruhe Institute of Technology
Karlsruhe, Germany
robert.heinrich@kit.edu

Holger Eichelberger, Klaus Schmid
Software Systems Engineering
University of Hildesheim
Hildesheim, Germany
{eichelberger,schmid}@sse.uni-hildesheim.de

***Abstract*—Big Data aims at the efficient processing of massive amounts of data. Performance modeling is often used to optimize performance of systems under development. Based on experiences from modeling Big Data solutions, we describe some problems in applying performance modeling and discuss potential solution approaches.**

***Index Terms*—Performance, modeling, Big Data, Palladio**

I. Motivation

Big Data solutions store, transfer, and process huge data sets. Modeling the architecture of Big Data systems in a component-based fashion and using architectural models for conducting performance analysis is essential to compare design alternatives. As Big Data solutions are typically realized as distributed systems, performance analysis can help to determine the required amount of resources and the distribution of analysis tasks over a cluster prior to implementation.

An established research area at the intersection of model-driven and component-based software engineering is using component models to conduct performance simulation and prediction. There are numerous approaches with varying focus in this area [3]. The key research that led to this paper was driven from experience with Palladio [4], however, we believe that most of our points hold for the broader field of model-based performance prediction of component-based systems.

In our research, we aim to apply model-based performance prediction to Big Data systems. Such systems typically use established infrastructures like Storm, Spark, etc. Our current focus is on performance models for near real-time computation for financial data streams [7]. As a consequence, our discussion of the approaches and their limitations is driven from this background. However, we believe that several of our points actually apply to a larger range of situations. As part of our discussion, we will make explicit which problems are particular to Big Data and which ones probably apply to a wider range.

The structure of the paper is: In Section II we describe the identified problems. This is the core of this contribution, as this is to our knowledge the first time these issues are made explicit. In Section III we discuss some initial ideas on how to address these issues. Finally, in Section IV we conclude.

II. Identified Problems

Our analysis was mainly driven from attempts to model the performance of Big Data systems for achieving adaptive behavior. Our modeling studies were performed with Palladio. Hence, in terms of experiences the points below must be regarded as Palladio-specific. However, as we will discuss, the points also apply to a broader range of modeling approaches and (partially) even beyond Big Data systems. First limitations of Palladio for modeling Big Data systems have been described in [7]. Our analysis goes beyond these limitations and is structured into 3+1 points: the first three issues refer to potential problems in modeling characteristics of Big Data systems, while the last point is of a more fundamental nature.

Flexible Component Distribution: In Big Data systems, it is often necessary to modify the distribution of worker implementations across different resources, e.g., switching from two servers responsible for a certain kind of processing to three servers hosting the same logical component. This can also be combined with adding servers (e.g., automatic scale-out). Such behavior is not specific to Big Data systems, as even web shops like Amazon do this. These distribution changes are triggered by changes in the workload of the system.

Today capabilities for describing arbitrarily complex dynamic component-resource binding do not exist in performance modeling and analysis approaches like Palladio. However, approaches like SimuLizar [1] and iObserve [2] are able to support some specific adaptations. SimuLizar considers deployment changes during analysis at design-time. iObserve focuses on reflecting observed changes in the running system in deployment like migration and (de-)replication.

Data-oriented Load Distribution: Big Data is centered on data processing and the data itself is used to control the applications, i.e., the processing component is selected based on the type of data (data-flow processing). In contrast, existing performance modeling approaches focus on the call-relation among components and do not consider data as first class entities. Thus, they do not provide capabilities for modeling data streams. Based on our experience, this makes it very hard to model Big Data applications. However, beyond ease and adequacy of modeling, it also leads to situations were specific, performance-relevant aspects cannot be modeled, e.g., if the amount of data stays the same over time, but the composition of data types change, this may lead to adaptations. It seems these issues are particularly relevant to Big Data applications.

Explicit Queuing Components: Big Data applications utilize queuing components for various purposes, but in particular, to organize the distribution of data across the application and to smooth peak loads. Thus, queuing has a very significant performance impact. The precise impact depends on specific

aspects of the queuing. Any performance models that omit such aspects are significantly insufficient, but current component-based modeling approaches like Palladio do not have modeling capabilities for queuing components. (Queuing exists, but is restricted to modeling resource usage). Thus, it is impossible to create adequate models of this aspect of system behavior. As internal queuing exists in other systems as well, we assume that this will be a problem for modeling these types of systems, too.

The previous three points describe three dimensions of modeling capabilities that are not – or not sufficiently – supported by existing performance modeling approaches like Palladio. However, there exists a broader and more fundamental problem to which we will turn now:

Blackbox Infrastructure: In Big Data applications, technologies like Storm, Spark, Hadoop, etc. play a central role, however, these are very large and complex infrastructures. There do not exist any models of their behavior and because of their size alone it would be a daunting task to construct one. The situation that large, unknown infrastructures are part of the models is not new. But, experience shows that for traditional systems, despite abstracting from classical infrastructures it was possible to get very adequate results [4]. This is different for Big Data, as the infrastructure operations may have a strong impact on system performance. This leads to the fundamental problem of how to derive models (at least of critical aspects) of such large infrastructures? Manual model construction seems out of scope, as the modeling alone would be often many times more complex than modeling the core application. Moreover, it would entail a significant reengineering project.

III. Solution Ideas

Based on the problems identified in Section II we will now discuss some potential solution approaches.

Flexible Component Distribution: Performance analysis of systems with flexible component distribution requires the inclusion of adequate modeling primitives to support this. An approach that already heads in this direction is SimuLizar [1]. It supports the modeling of self-adaptation rules, e.g. for load balancing. However, the expressiveness of its rules is not sufficient to support all relevant distribution adaptations.

In order to improve the capabilities of performance modeling approaches, we assume it is necessary to significantly enhance the capabilities for describing runtime adaptation rules and further enhance analysis approaches so the adaptation effects are taken into account in the analysis.

Data-oriented Load Distribution: As discussed earlier, the key issue is that the modeling of component-oriented approaches relies on call-relations, while in Big Data systems, the key relation is the data flow. Hence, we assume a natural, but necessary extension, will be to extend the modeling approaches with explicit data-flow modeling primitives.

However, we are not the first to propose this. Seifermann proposed such an extension for the Palladio approach [6]. His motivation was completely different, i.e., it focused on analyzing systems for privacy or SLAs violations. We imagine that an appropriate extension for dataflow modeling could actually simultaneously serve both purposes.

Explicit Queuing Components: To the best of our knowledge queuing components are not yet considered as predefined model elements on architecture level in existing approaches to architecture analysis. They may be constructed manually using formalisms like Layered Queuing Networks (LQNs) and Queuing Petri Nets (QPNs). However, to support an integrated performance analysis in a component-based paradigm, it would be necessary to integrate these capabilities into the underlying component models. We regard this as a difficult, but mandatory challenge for the performance analysis of Big Data systems.

Blackbox Infrastructure: Even if the above extensions would be sufficient to model the characteristics of complex Big Data systems, the problem would remain that the underlying infrastructures would need to be modeled as well. Given existing performance-oriented reengineering approaches, this would require significant reengineering efforts that seem rather unrealistic. Hence, the challenge here is: how can we (semi-) automatically derive sufficient model information from such infrastructures? If these infrastructures would be once comprehensively modeled, these models could be reused as a single component or as a parameterizable pattern.

IV. Conclusion

In this paper, we provided an initial discussion of current shortcomings in model-driven performance engineering. While it was based on modeling Big Data systems with Palladio, we believe the experiences hold at least for the broader range of performance modeling of big data systems and some of them may hold for a much wider range of cases were complex off-the-shelf infrastructures are used in system development.

It is the goal of our ongoing research to address the identified shortcomings by augmenting modeling approaches and providing novel methods for model construction.

V. References

[1] M. Becker, M. Luckey, and S. Becker. *Performance analysis of self-adaptive systems for requirements validation at design-time.* Quality of Software Architectures, pp. 43-52, ACM, 2013.

[2] R. Heinrich. *Architectural run-time models for performance and privacy analysis in dynamic cloud applications.* Performance Evaluation Review, 43(4):13-22, ACM, 2016.

[3] H. Koziolek. *Performance evaluation of component-based software systems: A survey.* Performance Evaluation, 67(8):634–658, Elsevier, 2010.

[4] J. Kroß, A. Brunnert, and H. Krcmar. Modeling Big Data Systems by Extending the Palladio Component Model. Softwaretechnik-Trends 35(3). GI. 2015.

[5] R. Reussner et al., (Ed.). *Modeling and Simulating Software Architectures – The Palladio Approach.* MIT Press, 2016. ISBN: 978-0-262-03476-0.

[6] S. Seifermann. *Architectural data flow analysis.* IEEE/IFIP Working Conference on Software Architecture, pp. 270-271, IEEE, 2016.

[7] C. Qin and H. Eichelberger. *Impact-minimizing Runtime Switching of Distributed Stream Processing Algorithms*. Big Data Processing - Reloaded, Joint Conference, CEUR-WS.org, 2016.