Modeling Big Data Systems by Extending the Palladio Component Model

Johannes Kroß,
Andreas Brunnert
fortiss GmbH
Guerickestr. 25
80805 Munich, Germany
{kross,brunnert}@fortiss.org

Helmut Krcmar
Chair for Information Systems
Technische Universität München
Boltzmannstr. 3
85748 Garching, Germany
krcmar@in.tum.de

ABSTRACT

The growing availability of big data has induced new storing and processing techniques implemented in big data systems such as Apache Hadoop or Apache Spark. With increased implementations of these systems in organizations, simultaneously, the requirements regarding performance qualities such as response time, throughput, and resource utilization increase to create added value. Guaranteeing these performance requirements as well as efficiently planning needed capacities in advance is an enormous challenge. Performance models such as the Palladio component model (PCM) allow for addressing such problems. Therefore, we propose a meta-model extension for PCM to be able to model typical characteristics of big data systems. The extension consists of two parts. First, the meta-model is extended to support parallel computing by forking an operation multiple times on a computer cluster as intended by the single instruction, multiple data (SIMD) architecture. Second, modeling of computer clusters is integrated into the meta-model so operations can be properly scheduled on contained computing nodes.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques

Keywords

Palladio Component Model, Performance Model, Big Data

1. INTRODUCTION

Exponentially growing volumes of data of various formats—referred to as big data—and the necessity of organizations to gain benefits have led to the development of big data systems [6], [10]. These systems are specialized for storing and processing this data. A common example includes Apache Hadoop[1] consisting of a distributed file system called *HDFS*, a scheduler and cluster resource manager called *YARN* and the *MapReduce* model for parallel data processing [8].

Although Apache Hadoop is originally built for commodity hardware, other systems such as Apache Spark (Streaming)[2] and Apache Storm[3] have emerged that enable low latency results on big data by also using in-memory computing [13]. Therefore, big data systems are able to meet continuously increasing performance requirements and to serve several additional use cases. Consequently, up-front performance evaluations for these systems and capacity planning for building an appropriate cluster become not only inevitable, but also difficult and costly [4, 5].

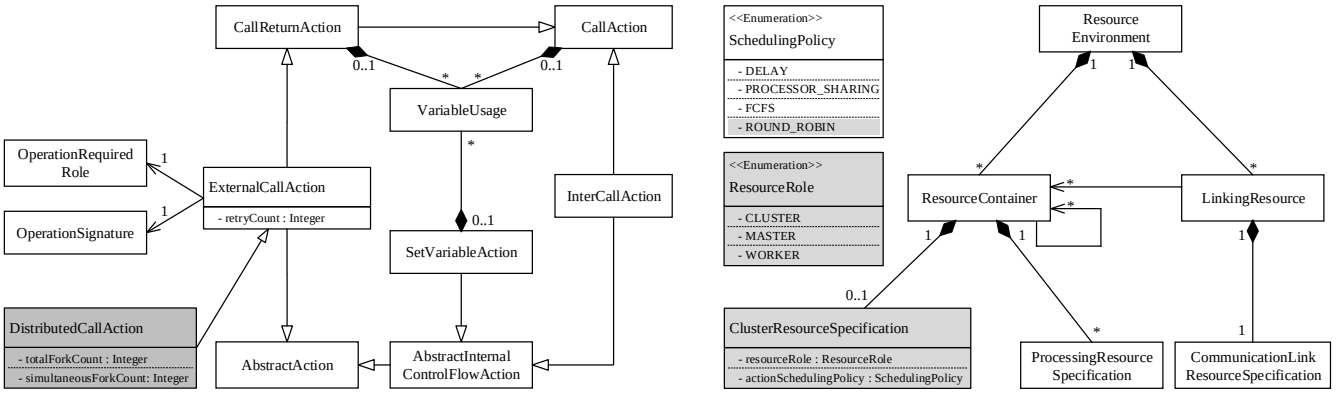
One way to approach these challenges are performance models such as the Palladio component model (PCM) that focuses on component-based software architectures [2]. It allows to model factors influencing system performance and predict performance metrics such as resource utilization, response time, and throughput by analytical solving or simulation [3]. As the PCM meta-model does not allow to model some specific requirements of big data systems yet, we propose and contribute a meta-model extension in this paper. This includes to specify an external call of an action to be executed multiple times in parallel while limiting the number of concurrent actions. It also includes to model a resource cluster consisting of several resource containers with different resource roles such as found in distributed computing architectures.

2. MODELING BIG DATA SYSTEMS

Comparing big data systems to ordinary component-based software systems (e.g., for web applications), they make use of specialized processing paradigms. Casado and Younas [6] list two main techniques that are common for big data systems, namely, batch and stream processing. They have in common that they utilize parallel and distributed computing on a distributed system in the form of a computer cluster. For this purpose, software developers implement components with operation signatures, for instance by using software libraries such as Apache Hadoop, and combine these components to build a task job that will be deployed on a computer cluster. In order to distribute this task job across linked resources, the components can be assembled in the form of a directed acyclic graph (DAG) [13]. For instance, the *MapReduce* paradigm consists of two vertices *map* and *reduce* that are linked by a directed edge. By this means, such systems are able to make use of all distributed computing resources and achieve horizontal scalability for increased workloads in terms of data volume or velocity.

Despite their shared characteristics, batch and stream processing adopt distinct approaches and are designed for different use cases. Batch processing is intended to be used on data sets with high volume [6]. In doing so, a specified operation is applied on splits of a non changing distributed data set multiple times in parallel. For instance, implemented Hadoop MapReduce operations are applied on distributed

[1]http://hadoop.apache.org/
[2]http://spark.apache.org/
[3]http://storm.apache.org/



(a) Service effect specification (SEFF) actions

(b) Resource environment

Figure 1: Meta-model extension for the Palladio component model (PCM, Version 3.4.1)

files on the HDFS. Implemented operations using Apache Spark are applied on so called resilient distributed datasets (RDD). The amount of parallelism for one specified operation is usually limited by the split rate of a dataset. The amount of simultaneously running parallel operations is usually limited by the amount of available resources or by specified user configurations.

Stream processing, on the other hand, is designated for handling high velocity data streams with low latency and is also referred to as real-time processing [6]. It distinguishes itself from batch processing by not operating on a data set, but rather operating on each data (e.g., Apache Storm) or a mini-batch (e.g., Apache Spark Streaming) that are kept in-memory. Therefore, data are continuously received from an unbounded data stream (e.g., in a message queue manner) and immediately processed by an operation. Similar to batch processing, the number of simultaneously running operations is limited by the amount of available resources or by specified configurations.

In previous work [9] we already modeled one *MapReduce* job on a single computer and predicted its response time. As we had to simplify several features and take limitations into account, we identified the need to extend PCM. Based on these findings and the above mentioned characteristics of batch and stream processing, we derive the following requirements of big data systems that we propose to extend PCM in order to allow for modeling typical big data systems:

1. Distribution and parallelization of operations
Component developers specify reusable software components consisting of operations using software frameworks like Apache Spark. In doing so, they may specify, but also may not know the definite number of simultaneous and/or total executions of an operation.

2. Clustering of resource containers
System deployers specify resource containers with resource roles (e.g., master or worker nodes), link them to a mutual network and logically group them to a computer cluster.

On this basis, we propose the following extensions for the PCM meta-model, which are shown in gray in Figure 1 (note that we only depict the relevant parts of the meta-model regarding our approach). The PCM meta-model consists of several partial models according to different developer roles [2]. Figure 1a shows the actions of the service effect specification (SEFF) model. A SEFF describes the behavior of an implemented operation. The element we propose to extend is the *ExternalCallAction* that is used to call a required service [2]. Therefore, we introduce a *DistributedCallAction*. It contains the two additional input parameters *simultaenousForkCount* and *totalForkCount* that can be used to specify the simultaneous and/or total number of executions of an external call as mentioned in the first requirement. Since these parameters depend on the workload and resource environment, component developers can describe the two input parameters as well as the resource demand of an operation as dependencies in parameterized form. In this way, domain experts are able to specify the usage of the component afterwards as proposed by Becker et al. [2].

Figure 1b shows the meta-model extension for the resource environment. Here, a *ResourceContainer* may or may not have several *ProcessingResourceSpecifications* to specify e.g., processors and hard disks. A *ResourceContainer* can also have a set of nested *ResourceContainers*. We propose to complement the *ResourceContainer* by a *ClusterResourceSpecification* which contains references to one *ResourceRole* as well as one *SchedulingPolicy*. These are both part of a *ResourceRepository*, that is intended to contain types of resources such as for middleware and operating system resources [2]. A *ResourceRole* is used to describe whether a *ResourceContainer* represents a cluster, a master or a worker. A *SchedulingPolicy* is used to describe how actions are distributed on a cluster.

An example for a modeled computer cluster is shown in Figure 2. An outer *ResourceContainer* is used to connect

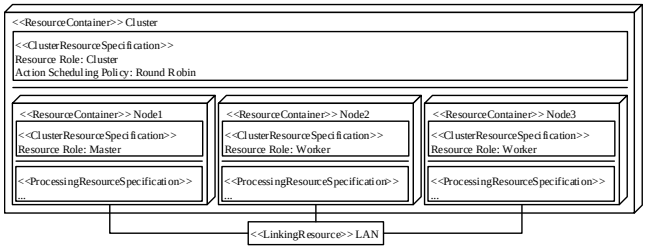


Figure 2: Example for a resource environment diagram

computing nodes to a cluster and includes a round robin strategy to schedule actions on its nested *ResourceContainer*. This enables system deployers to simply allocate components to a cluster. Furthermore, each nested *ResourceContainer* includes a *ResourceRole*. If only workers are specified, the cluster will represent a shared-nothing architecture. If one master is specified, it will be responsible for distributing actions. Therefore, its *ResourceContainer* operates usually special middleware with additional resource demands that can be modeled with *InfrastructureCalls* in PCM.

3. RELATED WORK

Most of the existing performance modeling approaches for big data systems concentrate only on one technology, namely Apache Hadoop. Barbierato et al. [1] introduce a performance modeling language to evaluate the performance of queries using Apache Hive which is a data warehouse software on top of Apache Hadoop with a SQL-like language. Vianna et al. [12] propose an analytical model, which combines a precedence graph model and a queuing network model, to model MapReduce workloads concentrating on the pipeline parallelism between *map* and *reduce* operations. Verma et al. [11] propose a framework consisting of micro benchmarks and a regression-based model to predict and evaluate response times of MapReduce processes for different cluster resource choices.

A more general approach regarding big data technologies is, for instance, introduced by Castiglione et al. [7] which use Markovian agents and mean field analysis to model big data batch applications and to provide information about performance of cloud-based data processing architectures. However, there is no approach available to the best of our knowledge that tries to enable modeling of general batch and stream processes, and to predict the response time and cluster resource utilization for their concurrent execution.

4. CONCLUSION AND FUTURE WORK

In this paper we introduced a generic performance modeling formalism to model essential characteristics of data processing as found in big data systems. For this purpose, we presented two meta-model extensions for PCM that enable performance engineers to model a computer cluster and to apply distributed and parallel operations on this cluster. This allows to model general stream processing as well as batch processing techniques independent of their technology and to realize up-front performance evaluations for response times, throughputs, and resource utilizations of CPU and memory of big data systems.

We already implemented the meta-model extensions, graphical modeling editors, model-2-code transformations and basic functionalities of the associated simulation framework SimuCom [2] to support our extensions. Although we do not consider network traffic between resource containers yet, first experimental results already look promising. In future, we plan to complete the SimuCom extension as well as integrate network traffic. Afterwards, we intend to comprehensively evaluate our meta-model extension in controlled experiments. This includes up- and downscaling scenarios regarding workload as well as resource capacities. Our long-term goal is to automatically derive performance models for batch and stream processes based on measurement data from middleware like Apache YARN.

5. REFERENCES

[1] E. Barbierato, M. Gribaudo, and M. Iacono. Performance evaluation of NoSQL big-data applications using multi-formalism models. *Future Generation Computer Systems*, 37(0):345 – 353, 2014.

[2] S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *The Journal of Systems and Software*, 82(1):3–22, 2009.

[3] F. Brosig, P. Meier, S. Becker, A. Koziolek, H. Koziolek, and S. Kounev. Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures. *IEEE Transactions on Software Engineering*, 41(2):157–175, 2015.

[4] A. Brunnert and H. Krcmar. Continuous performance evaluation and capacity planning using resource profiles for enterprise applications. *Journal of Systems and Software*, 2015. doi: 10.1016/j.jss.2015.08.030.

[5] A. Brunnert, C. Vögele, A. Danciu, M. Pfaff, M. Mayer, and H. Krcmar. Performance management work. *Business & Information Systems Engineering*, 6(3):177–179, 2014.

[6] R. Casado and M. Younas. Emerging trends and technologies in big data processing. *Concurrency and Computation: Practice and Experience*, 27(8):2078–2091, 2015.

[7] A. Castiglione, M. Gribaudo, M. Iacono, and F. Palmieri. Modeling performances of concurrent big data applications. *Software: Practice and Experience*, 2014.

[8] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[9] J. Kroß, A. Brunnert, C. Prehofer, T. Runkler, and H. Krcmar. Stream processing on demand for lambda architectures. In M. Beltrán, W. Knottenbelt, and J. Bradley, editors, *Computer Performance Engineering*, volume 9272 of *Lecture Notes in Computer Science*, pages 243–257. Springer International Publishing, 2015.

[10] M. Schermann, H. Hemsen, C. Buchmüller, T. Bitter, H. Krcmar, V. Markl, and T. Hoeren. Big data - an interdisciplinary opportunity for information systems research. *Business & Information Systems Engineering*, 6(5):261–266, 2014.

[11] A. Verma, L. Cherkasova, and R. H. Campbell. Profiling and evaluating hardware choices for MapReduce environments: An application-aware approach. *Performance Evaluation*, 79:328 – 344, 2014.

[12] E. Vianna, G. Comarela, T. Pontes, J. Almeida, V. Almeida, K. Wilkinson, H. Kuno, and U. Dayal. Analytical performance models for MapReduce workloads. *International Journal of Parallel Programming*, 41(4):495–525, 2013.

[13] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.