EPJ Web of Conferences **214**, 03027 (2019) https://doi.org/10.1051/epjconf/201921403027
CHEP 2018

Modeling and Simulation of Load Balancing Strategies for Computing in High Energy Physics

René Caspart[1*], *Patrick* Firnkes[2**], *Manuel* Giffels[1***], *Anne* Koziolek[2****], *Günter* Quast[1†], *Ralf* Reussner[2‡], and *Maximilian* Stemmer-Grabow[2§]

[1]Institute for Experimental Particle Physics (ETP)
[2]Institute for Program Structures and Data Organization (IPD)
at Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Abstract. The amount of data to be processed by experiments in high energy physics (HEP) will increase tremendously in the coming years. To cope with this increasing load, most efficient usage of the resources is mandatory. Furthermore, the computing resources for user jobs in HEP will be increasingly distributed and heterogeneous, resulting in more difficult scheduling due to the increasing complexity of the system. We aim to create a simulation for the WLCG helping the HEP community to solve both challenges: a more efficient utilization of the grid and coping with the rising complexity of the system. There is currently no simulation in existence which helps the operators of the grid to make the correct decisions while optimizing the load balancing strategy. This paper presents a proof of concept in which the computing jobs at the Tier 1 center GridKa are modeled and simulated. To model the computing jobs we extended the Palladio simulator with a mechanism to simulate load balancing strategies. Furthermore, we implemented an automated model parameter analysis and model creation.
Finally, the simulation results are validated using real-word performance data. Our results suggest that simulating larger parts of the grid is feasible and can help to optimize the utilization of the grid.

1 Introduction

The amount of data processed by experiments in high energy physics (HEP) will drastically increase in the coming years due to the upgrade of the LHC to the High-Luminosity Large Hadron Collider (HL-LHC), creating major challenges for the HEP community [1]. Figure 1 shows the estimated required and provided CPU resources for the ATLAS experiment in the next ten years. Assuming a flat computing budget model, the estimated provided resources do not cover future needs for computing resource in the period from 2026 to 2028 due to the start of the HL-LHC era. One of the key aspects to deal with this increasing load is the more efficient usage of existing and future resources.

*e-mail: rene.caspart@cern.ch
**e-mail: patrick.firnkes@student.kit.edu
***e-mail: manuel.giffels@cern.ch
****e-mail: koziolek@kit.edu
†e-mail: g.quast@kit.edu
‡e-mail: reussner@kit.edu
§e-mail: mail@mxsg.de

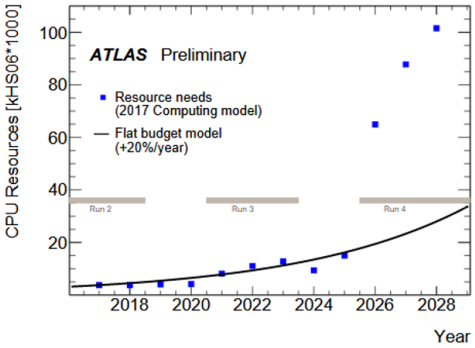

Figure 1. Estimated CPU resource needs of the ATLAS experiment[1]

Another challenge for the HEP community is the increasing heterogeneity and distribution of the computing resources for user jobs. Cloud resources, such as the Helix Nebula Science Cloud[1] and Amazon Web Services, or institute clusters and Tier 3 centers will be progressively introduced for the computation of these jobs. This results in a more complex scheduling, because each resource has different optimal job types and mixes. Furthermore, a more dynamic data placement, where fewer centers host data, leads to an increase in the number of remote data accesses.

We aim to create a simulation for the WLCG helping the HEP community to solve both challenges: a more efficient utilization of the grid and coping with the rising complexity of the system. Currently, there is no model or simulation in existence to evaluate the load balancing strategies for the grid, helping the operators to make the correct decisions while optimizing the load balancing strategy. Section 7 describes different attempts to simulate the WLCG or parts of it. However, they either showed scalability issues or the simulations do not have the necessary detail level to simulate effects on working nodes that is necessary to evaluate load balancing strategies.

Optimizing load balancing strategies without the help of simulations requires stable load in order to preserve the comparability of the different strategies. This is neither possible to achieve for the WLCG nor for a single center like GridKa, because the nature of the load is highly dynamical. Another option would be to use an isolated test system for the evaluation. However, the system under inspection is too large and thus a similar test system would not be financially affordable. A smaller test system could not deliver the correct results because too much of the complexity of the system would be taken away. These reasons lead to the conclusion that a model and simulation of the performance of computing jobs is required to evaluate different load balancing strategies.

Our long term goal is to evaluate different load balancing strategies for the WLCG and additionally evaluate different design decisions affecting the performance of the grid. These gained insights will help the HEP community to solve the challenges caused by the HL-LHC upgrade.

The contribution of this paper is the modeling and simulating of CMS computing jobs at the Tier 1 center GridKa as a proof of concept. The model and simulation are created using the Palladio simulator, see Section 2 for details on Palladio. We chose Palladio to achieve a scalable simulation because it is a comparably abstract simulator (Section 7 gives an overview on other simulators and approaches). As the simulation of load balancing strategies is a new use case for Palladio, we created an extension for the simulator for this use case. Furthermore, we created an automated model parameter analysis and model creation in order to increase the usability of the simulation. This model makes it possible to evaluate different load balancing

[1] http://www.helix-nebula.eu/

strategies for GridKa and enables us to draw conclusions about the best optimizations of the load balancing for the WLCG. Also, it can be used as a foundation for models and simulations of larger parts of the WLCG.

This paper is organized as follows: Section 2 presents the Palladio simulator as a foundation of the simulation. Section 3 gives a overview on the simulation process. Section 4 describes how different elements of the computing jobs and the computing resources are represented in our model. Section 5 explains how we determine the model parameters and the automated model creation works. The validation is presented in Section 6 and Section 7 puts our work into context with the related work. In Section 8, possible design decisions and how they could be modeled and simulated, are presented. Finally, Section 9 gives a conclusion on this paper.

2 Foundations: Palladio

Palladio is a model driven software architecture simulator, which is implemented as a plugin for the Eclipse IDE and developed by Karlsruhe Institute of Technology (KIT), FZI Research Center for Information Technology, and University of Paderborn. Palladio predicts quality of software properties (e.g. performance) using the following models of a system [2]:

- The component model: Specifies the structure and behavior of the components independently from their later usage. This allows reuse of the components, but requires its parametrization.
- The assembly model: Represents how the components are connected to model the software architecture.
- The resource model: Describes the resource environment. It contains the number and characteristics of the resource containers on which the components could be deployed and the topology of the network connecting these containers.
- The allocation model: Represents the mapping of the different components to the resource containers.
- The usage model: Defines the usage of a system regarding workload, user behavior, and parameters.

Once these models are created, Palladio can be used to simulate the system choosing one of several supported simulators.

Lehrig and Becker [3] extended Palladio with Architectural Templates to make it possible to model cloud computing environments efficiently. Furthermore, Palladio's simulation approach SimuLizar was created to simulate self-adaptive systems [4]. It was later extended to support the metrics scalability, elasticity, and efficiency.

Palladio has been successfully used to solve several industrial problems and was used to optimize cloud infrastructure in the context of chemical computing as part of the CACTOS project [5].

3 Simulation Process

To receive a valid model of a site and its jobs, we created the simulation process depicted in Figure 2. This process consists of five phases: Matching of data sets, grouping the data for job types, analysing the model parameters, generating a blueprint model, and finally injecting the parameters into this model. Note, only a part of the data is used to calibrate the model, the other part is used for its validation.

The starting point of the simulation process is matching the data sets from different data sources as a prerequisite of the analysis. In our proof of concept we use data from three

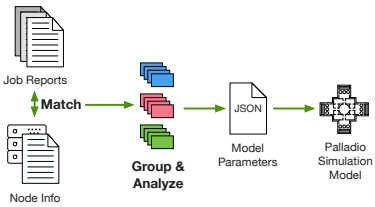

Figure 2. The Simulation Process

data sources: The CMS WMArchive, CMS GridKa job monitoring data and GridKa node performance data. These data sets have to be matched because each set contains different information about an executed job.

After matching and grouping the data for job types, the analysis of the job and working node data to extract performance parameters for the model takes place. For example, these are the CPU demand of jobs, the frequency of the job types or the CPU power of the nodes. This analysis and the data set matching are automated and described in Section 5.

The next step is the generation of the model of GridKa and the injection of the previously extracted performance parameters. Neither step requires manually modeling. At this point the users have a valid and calibrated model of GridKa with its resource and the workload. The properties of this model are described in Section 4.

Users can then either run the simulation or make changes to the model to simulate the effect of a design change. Finally, after running the simulation, they can inspect the results, such as core utilization of a node or throughput of jobs, using the integrated visualization tool or they can export the results and use a third party software to do the evaluation.

4 Modeling Computing Jobs and Resources

This section describes how the computing jobs and working nodes are modeled using Palladio. The component model describes each type of CMS computing job, such as analysis or reprocessing jobs, with their resource demands. These demands are the *CPU demand*, the *I/O demand*, the *number of threads*, the *required job slots*, and the *number of demand rounds*.

Figure 3 shows the behaviour of the job during the simulation. After a job is scheduled on a resource container, it acquires the specified number of job slots or waits if the node does not have sufficient free slots. Afterwards, the CPU and I/O demands of the job are processed. To avoid one big I/O operation at the start of a job and to make the simulated processing of I/O more continuous, it is modeled using iterations that reflect the number of processed events. Thus, for each iteration there are I/O and CPU resources needed. Rather than simulating each event, we simulate a fraction of the processed events, because the simulation of each event would harm the scalability of the simulation. Whilst processing, the job uses the specified number of threads to process the CPU demand. Once the job is finished, it releases its acquired job slots. All of these parameters are specified as a probability function and are extracted from measured performance data as described in Section 5.

Each type of working node in GridKa is modeled in the resource model. Each node type has a *CPU and I/O processing rate*, *number of cores*, *number of job slots*, and *number of instances* of this node type. The types of computing nodes are duplicated according to their specified number of instances as a preprocessing step of the simulation, resulting in the simulation of each working node.

At GridKa every type of computing job can run on every node, thus each job type is allocated to each computing node in the allocation model.

The usage model defines which and how many jobs are scheduled. To make sure that there is always a high load on the system, we simulate a closed workload with enough jobs

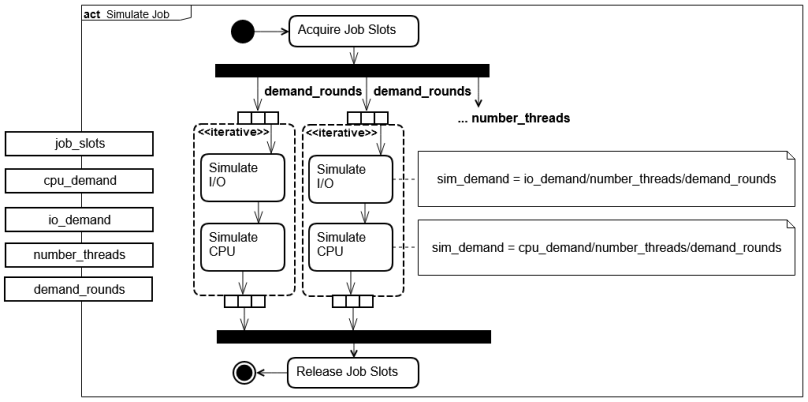

Figure 3. Model of Job Behavior

that the system is fully utilized. Each job type has a configurable *job share* of the overall load.

We extended Palladio with a *load balancing strategy action*[2], so that we can simulate load balancing strategies. This load balancing strategy action is contained in a load balancing component. In the assembly model all job components are connected with the load balancing component, which is the entry point of the system. This action decides to which node a job should be scheduled, based on the currently free job slots of each working node. The load balancing strategies can be extended to evaluate different strategies.

5 Model Parameter Extraction and Model Creation

To obtain a comprehensive view both of the resource usage of CMS jobs executed at a WLCG site as well as the resources available there to process them, combining information from multiple data sources is required:

- We acquire job monitoring information from several global CMS monitoring systems: Job reports from JobMonitoring and Workflow and Data Management framework job reports (FWJR documents) are archived on the *analytix* Hadoop cluster. The latter are archived on the cluster as part of the WMArchive project [6]. We collect job information using the *CMSSpark* framework by Kuznetsov et al. [7].

  For this proof of concept, only jobs executed at the GridKa site were considered, but these data sources can be used to collect job reports from arbitrary WLCG sites. As these data sets contain different metrics for each executed job and not every job is contained in each data set, they are matched to each other to identify records that describe the same job execution prior to analysis.

- Site-specific performance data sets include the available nodes at the site and their performance information (including node benchmarks such as HS06 scores), which are available from local site monitoring systems. The simulation currently only includes computing jobs from CMS. As GridKa also processes jobs for other experiments, the number of available nodes at the site is scaled down to match the share of CMS workloads at the site in the time frame used for calibration. This share of resources is also collected from site-local monitoring information, as well as CPU efficiency reference data for the workloads executed in the same time frame.

[2]https://doi.org/10.5281/zenodo.1464569

To allow quick extraction of model parameters from these data sets, we implemented data set matching as well as the following analysis steps in an automated, configurable model parameter extraction tool written in Python. Matching job information to local node performance information allows their resource usage to be decoupled from the hardware used to execute them, e.g. by combining the CPU time they used with node benchmarks of the machine they were executed on.

For the purposes of this proof of concept, jobs were then grouped according to their type as indicated in the job reports, but the extraction framework allows to be configured for arbitrary groupings of the extracted jobs. Job resource requirement distributions for the different job types are extracted as distributions based on the empirical distributions in the data sets used for calibration, in the form of discrete probability density functions.

The parameter extraction tool[3] exports model parameter sets as JSON files. To simplify calibrating simulation models with these parameter sets, we created a Palladio plugin[4] that can be used to automatically complete a blueprint of the simulation model (which defines the structure of the model as described in section 4) with arbitrary parameter sets. This ensures an efficient and flexible model creation process.

6 Validation

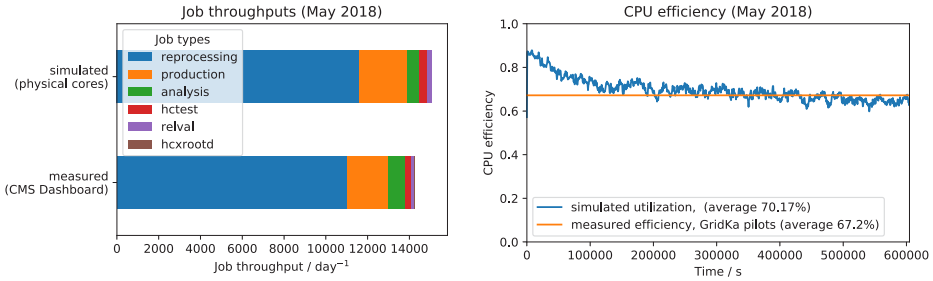

Figure 4. Measured and simulated throughputs and CPU efficiency job data from May 2018

To validate the model and simulation we simulated CMS jobs at GridKa in May 2018. The job performance data was split into two equal sized calibration and test datasets by randomly sampling reports from the full data set. For the validation we considered the following metrics: Overall throughput of jobs, shares of the job types, and the overall CPU efficiency of the site. These metrics describe how well the simulation matches the measured performance data. Comparisons between simulation results and measurements from the system are shown in Figure 4. One result is that simulated throughput matches the reference with a difference of about 5.6%. Furthermore, the shares of the simulated and measured job types also match closely.

The CPU utilization in the simulation and a comparison with the measured average CPU efficiency are shown in Figure 4, which also makes the initial ramp-up of the simulation in the region up to 200000 s visible. The overall average utilization in the simulation deviates from the reference by about 3%, showing that even the current simplified model is capable of reproducing the metrics introduced above. Additionally, the runtimes of the simulated jobs were measured: The average runtime of simulated jobs deviates by approximately 8% from the measured runtimes in the test data set.

Simulating one week of CMS jobs at GridKa takes about 24 minutes on a laptop. Thus, it is feasible to simulate several sites at once in a reasonable time, however simulating larger parts of the grid is left as future work.

[3]https://doi.org/10.5281/zenodo.1466003

[4]https://doi.org/10.5281/zenodo.1466006

7 Related Work

The MONARC project was started in 1998 to help with the development of the initial design of the WLCG at CERN [8]. However, after the initial design was developed, the project was discontinued in 2000 and thus no model and simulation of the WLCG was created. In 2004 a second version of the MONARC simulator was released with several improvements, such as simulation of data replication. However, MONARC2 showed scalability issues when simulating a Tier-2 site and accuracy issues regarding network load [9].

There are a variety of grid and cloud simulators in existence today, of which SimGrid and CloudSim are two popular examples. However, both of them are implemented as libraries and not as standalone simulators. Thus, both are missing graphical model editors which makes the modeling of the computing jobs time-consuming [10, 11]. The support of graphical editors is a big advantage of Palladio and increases its usability and efficiency. Another advantage of Palladio is the separation of the system into different models, which enables one to change one part of the system without modifying others. Furthermore, both lack the support of an integrated visualization tool to display the results.

CACTOS is an approach to cloud infrastructure automation and optimization [12]. One part of the approach is the CACTOS Runtime Toolkit for monitoring and resource management and another part is the CACTOS Prediction Toolkit for evaluation of alternative data center deployment scenarios and resource management algorithms. The Prediction Toolkit is built on Palladio and SimuLizar. A first case study in the context of chemical computing showed that the Prediction Toolkit has high accuracy, but has not been tested on large systems yet [5]. However, due to the required Runtime Toolkit CACTOS was not applicable in our use case.

Korenkov et. al simulated the JINR Tier1 site in 2016 [13] and Hushchyn et al. created a simulation for the LHCb Grid in 2017 [14]. Both simulations focused on the effects of the location of the required job data on the throughput of the sites. However, they neither modeled each working node nor modeled details of the scheduling process like job slots, thus both simulations would not be usable to simulate load balancing strategies on working node level.

8 Outlook

Our next steps are the evaluation of different design decisions. In this section a selection of possible design decisions that could be simulated are presented.

One possible design decision is a more sophisticated load balancing strategy that aims for a better resource utilization of the working nodes by scheduling to it a balanced share of CPU and I/O heavy jobs. To simulate this, a new scheduling strategy must be implemented.

Another design decision would be to experiment with different sizes of pilot jobs, assuming that smaller pilot jobs result in a better utilization due to less wasted CPU capacity in the draining phase of a pilot. For this use case the model must be extended with the explicit modeling of pilots jobs.

Additionally, the effects of integrating cloud resources could be simulated. The model must be extended with the cloud resources and the required network load to access the files from a site. Then a good load balancing strategy for the integration of cloud resources could be evaluated, such as only scheduling CPU heavy jobs to cloud resources and keeping I/O heavy jobs near the data.

Also the effects of data caches could be simulated. Firstly, several sites must be modeled to simulate their interactions. Secondly, the jobs must be parametrized with the possibility of remote access. Thirdly, the data cache with its hit rate should be modeled. Finally, the

performance improvements of the data cache achieved by transferring data from the cache, and not from sites, can be simulated.

9 Conclusion

Computing resources for user analyses will become increasingly distributed and heterogeneous in the future and scheduling will be a key component for the efficient usage of the grid. We are confident that our approach helps to improve utilization of the grid and to be able to cope with the rising complexity of the system, thus serves the HEP community as a tool to solve the challenges created by the HL-LHC update.

As a proof of concept we successfully simulated CMS computing jobs at GridKa. To enable easy usage of the simulation we implemented an automated model parameter extraction and model creation. These methods can not only be used for CMS monitoring data, but are also applicable for computing jobs from any other experiment providing similar monitoring information. Furthermore, we extended Palladio to be able to simulate load balancing strategies. Our first results are promising and suggest that this approach can be used to simulate load balancing strategies accurately. Regarding scalability, we are confident that several computing centers can be simulated in a sufficient time, however this is part of future work.

This proof of concept showed that an automated model parameter extraction and model creation is possible and that Palladio can be used to evaluate load balancing strategies for computing in HEP. In contrast to the existing simulations our simulation models a site in a more detailed way, enabling the evaluation of load balancing strategies on working node level. The next step is the evaluation of design decisions mentioned in Section 8.

References

[1] A.A. Alves Jr, G. Amadio, N. Anh-Ky, L. Aphecetche, J. Apostolakis, M. Asai, L. Atzori, M. Babik, G. Bagliesi, M. Bandieramonte et al., arXiv preprint arXiv:1712.06982 (2017)

[2] S. Becker, H. Koziolek, R. Reussner, Journal of Systems and Software **82**, 3 (2009), special Issue: Software Performance - Modeling and Analysis

[3] S. Lehrig, M. Becker, *Approaching the cloud: Using palladio for scalability, elasticity, and efficiency analyses*, in *Proceedings of the Symposium on Software Performance* (2014), pp. 26–28

[4] M. Becker, S. Becker, J. Meyer, Software Engineering **213**, 71 (2013)

[5] C. Stier, J. Domaschka, A. Koziolek, S. Krach, J. Krzywda, R. Reussner, *Rapid Testing of IaaS Resource Management Algorithms via Cloud Middleware Simulation*, in *ACM / SPEC International Conference on Performance Engineering (ICPE'18)* (ICPE, 2018)

[6] *WMArchive: Workload Archive project*, `https://github.com/dmwm/WMArchive`

[7] V. Kuznetsov, J. Rumševičius, D. Lange, dciangot, *Cmsspark* (2018), `https://doi.org/10.5281/zenodo.1401228`

[8] M. Collaboration et al., Tech. rep., Technical Report CERN/LCB-001, CERN, 2000. http://www.cern.ch/MONARC (2000)

[9] Zach, Betev, Adamová, the ALICE Collaboration, Journal of Physics: Conference Series **331**, 072038 (2011)

[10] H. Casanova, A. Giersch, A. Legrand, M. Quinson, F. Suter, Journal of Parallel and Distributed Computing **74**, 2899 (2014)

[11] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, Software: Practice and experience **41**, 23 (2011)

[12] P.O. Östberg, H. Groenda, S. Wesner, J. Byrne, D.S. Nikolopoulos, C. Sheridan, J. Krzywda, A. Ali-Eldin, J. Tordsson, E. Elmroth et al., *The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation*, in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science* (2014), pp. 26–31

[13] V. Korenkov, A. Nechaevskiy, G. Ososkov, D. Pryahina, V. Trofimov, A. Uzhinskiy, N. Voytishin, *The JINR Tier1 Site Simulation for Research and Development Purposes*, in *EPJ Web of Conferences* (EDP Sciences, 2016), Vol. 108, p. 02033

[14] M. Hushchyn, A. Ustyuzhanin, K. Arzymatov, S. Roiser, A. Baranov, *The LHCb Grid Simulation: Proof of Concept*, in *Journal of Physics: Conference Series* (IOP Publishing, 2017), Vol. 898, p. 052020