Modelling and Simulation of Load Balancing Strategies for Particle Physically Experiments at CERN

Project Proposal of

Patrick Firnkes

April 30, 2018

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer: Jun.-Prof. Koziolek
Advisor: Jun.-Prof. Koziolek

Contents

1 Abstract

The amount of data created by high energy experiments at CERN will drastically increase in the next years and cannot be processed without an efficient resource usage. To achieve a better resource utilization the load balancing strategies have to be optimized. However, such optimizations are tedious and error prone for large and complex computing systems without the help of simulations.

To improve the load balancing of the World Wide LHC Computing Grid (WLCG) we model and simulate the performance of computing jobs executed at the Tier 1 centre GridKa. This model is validated using real world data and enables us to evaluate various load balancing strategies.

2 Motivation

The WLCG is one of the largest grids in the world and processes the data created by the Large Hadron Collider (LHC) at CERN [5]. The amount of data to be processed steadily increases, expected to be about 50 Petabytes in 2018. Therefore, the grid has to be constantly extended and optimized to be able to handle the increasing load.

Nevertheless, the amount of data will drastically increase in the coming years due the upgrade of the LHC to the High-Luminosity Large Hadron Collider (HL-LHC), creating major challenges for the community of HEP [1]. The goals of the HL-LHC upgrade include the search for dark matter, the investigation of matter-antimatter differences and exploiting the discovery of the Higgs boson as a precision tool. Figure 2.1 shows the estimated CPU resources for the ATLAS experiment in the next ten years. As can be seen there is a large gap between needed resources and estimated resources in the period from 2026 to 2028. This gap is created due the start of the HL-LHC era, which requires much more computing resources. One approach to close this gap is to improve the load balancing strategy.

However, there is currently no model or simulation in existence to evaluate the load balancing strategies of the grid, helping the operators to make the correct decisions in the optimization process. Optimizing load 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 centre 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 systems would not be financially affordable. A smaller test system could not deliver correct results because too much of the complexity 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 created by the HL-LHC upgrade. The first step to archive this goal is to model and simulate the performance of computing jobs at the Tier 1 centre GridKa. The model and simulation will be created using Palladio, which leads to the research question: which extensions to Palladio are required to achieve accurate results that can be used to evaluate load balancing strategies for GridKa?

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

The following sections describe the WLCG in detail, the current state of the load balancing, and the desired state in the future.

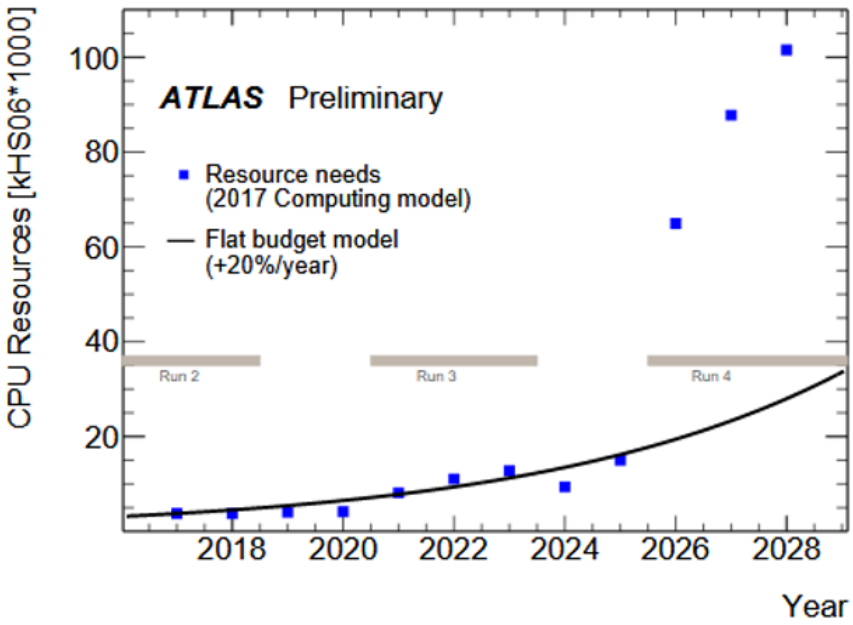


Figure 2.1: Estimated CPU resources of ATLAS [1]

2.1 Worldwide LHC Computing Grid

The WLCG processes 50 Petabytes of data each year created by the LHC at CERN [14] and with its help the Higgs boson was discovered in 2012 [17].

Over 170 computing centres distributed over 42 countries are part of the WLCG and it combines over 72.000 CPUs, 386 Petabytes online data space and 368 Petabytes nearline space (tape storage) [15]. Every day it runs about 2 million jobs to analyse data of the CERN experiments.

At CERN there are four experiments taking place, namely: Atlas, Alice, CMS, and LHCb. They all use the grid in a similar, but not the same, way [4]. Figure 2.2 shows the hierarchical three tier structure of the WLCG. Tier 0 is the CERN compute centre, which stores all raw data of the experiments, makes the first pass reconstruction and distributes raw data to Tier 1 compute centres. Tier 1 consists of 13 sites, which store a share of raw and reconstructed data, creating a second copy. They also run analysis and simulation jobs. Tier 2 sites, which are mostly located at universities or other scientific institutes, conduct monte carlo simulations. In contrast to Tier 1 sites, the Tier 2 sites do not have much storage [4].

The WLCG is heterogeneous, meaning that the resources of one site are different to the resources of another site, the network connections between sites vary, and even the nodes in a single site have different resources. It is becoming more common to use virtualization, resulting in even more heterogeneity [5].

Our work will focus on modelling and simulating load balancing strategies for the CMS experiment. Yet the results of our work can be presumably transferred to the other experiments, because they all use the grid in a similar way.

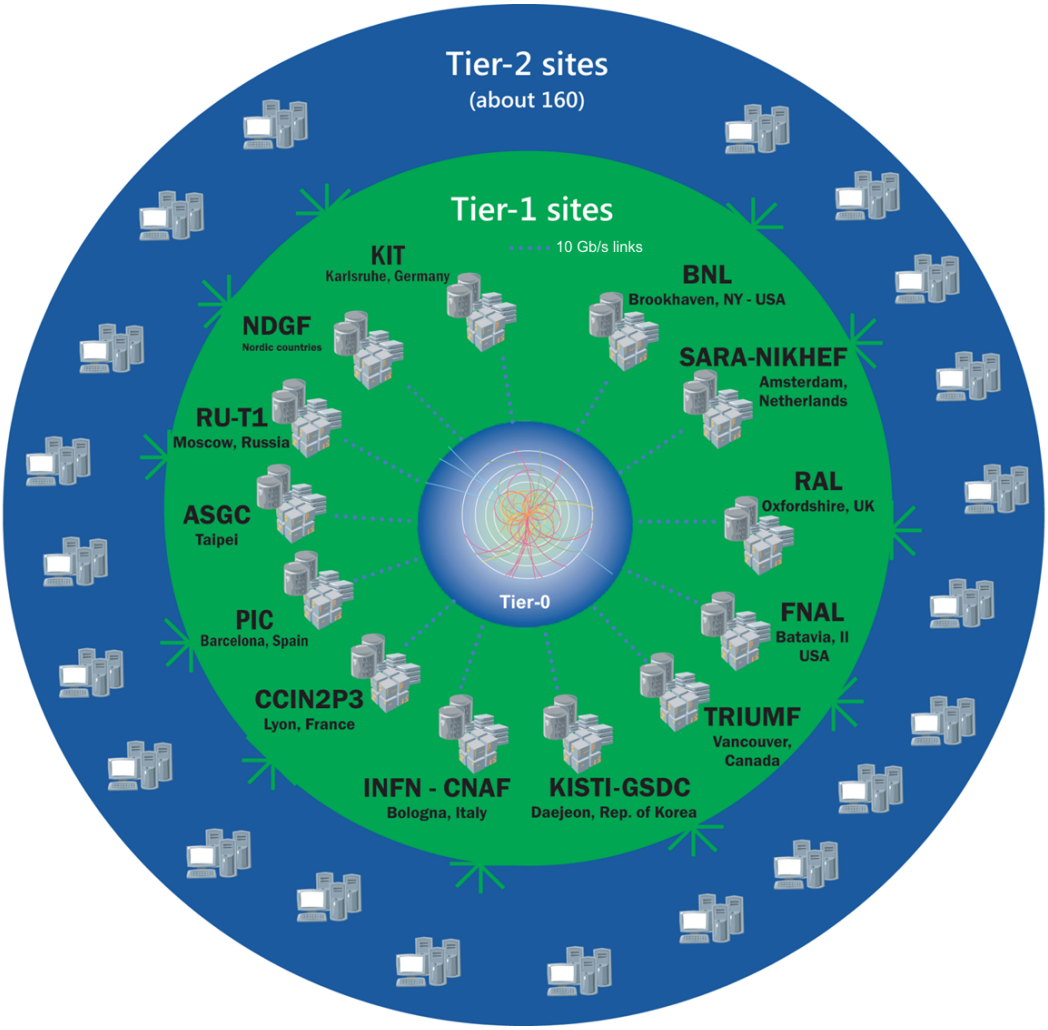


Figure 2.2: Tier structure of the WLCG [16]

2.2 Current State

Currently, the load balancing strategy used at the CMS computing model is improvable. One aspect is that it does not account for the nature of the jobs: simulation jobs require heavy I/O load and analysis jobs require heavy CPU time [19]. If this nature is not accounted for in the scheduling decision, it leads to a bad utilization of the nodes. A good strategy would schedule the right amount of simulation and analysis jobs to a site to maximize the utilization of I/O and CPU. For example, if too many simulation jobs were scheduled, the I/O would create a bottleneck and the CPUs would be idle. There are several other ideas how the load balancing could be improved, but they cannot be evaluated without a simulation.

Another approach to improve the CMS computing model is to upgrade the grid infrastructure such as integration of cloud computing resources or creation of better links between the sites. A simulation could help to find the best improvement in terms of speed-up, cost or utilization.

2.3 Envisioned Solution

The amount of data produced by the LHC is steadily increasing, thus either more computing resources have to be provided or the existing ones have to be used in a more efficient way. However, the computing resources will not increase in the same magnitude as the produced data, hence a better utilization of the given resources is required. Our approach will allow to simulate the effects of different load balancing strategies and decide on these results what the best strategy is. The scheduling problem itself is np complete, so the optimal solution will not be found, but we can determine, from a set of strategies, which one is the best [13]. Evaluating load balancing strategies using the productive system is not accurate, because the workload constantly changes. This makes the performance results of different load balancing strategies incomparable. Though, this problem will be solved by using simulations.

Furthermore, our approach can be used to evaluate the effects of changing the infrastructure of the grid. Therefore, we can find the best approach to improve the CMS computing model, e.g. adding cache, create a new network connection or allow dynamic appearing nodes. It is difficult to find the best approach without using a simulation, because one cannot consider all side-effects of a change. For example increasing the CPU resources may be a bad decision, because then the network would become the bottleneck. Only the increasing of both, the network and the CPU resources, would lead to an improvement.

3 State of the Art

In this chapter four simulators are presented which could be used to model and simulate GridKa, from which we chose Palladio to use in this project. There are several other simulators, but these are the most relevant ones.

It should be noted that there is no indication that a similarly large system like GridKa has been simulated before. Our work will determine if it is possible to model and simulate such large systems with the current state of simulators.

3.1 Palladio

Palladio is a model driven software architecture simulator developed by Karlsruhe Institute of Technology (KIT), FZI Research Center for Information Technology, and University of Paderborn. The development started in 2003 and it is still actively developed today. Palladio predicts quality of software properties (e.g. performance) using several models of a system [3].

These models are the component model, the assembly model, the resource model, the allocation model, and the usage model. The component model specifies the structure and behaviour of the components independently from their later usage. This allows reuse of the components, but requires its parametrization. To represent how the components are connected to model the software architecture, the assembly model has to be created. 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. Allocation models represent the mapping of the different components to the resource containers. Finally, the usage model defines how a system is used regarding workload, user behaviour, and parameters.

Once these models are created, Palladio can be used to simulate the system choosing one of several supported simulators. Figure 3.1 shows the modelling and simulation process.

Lehrig and Becker [9] 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 [2]. It was later extended to support the metrics scalability, elasticity, and efficiency [9].

The first case studies using Palladio in cloud environments show that it delivers accurate results [9]. However, Palladio was never used to model and simulate such a large system as the CMS computing model.

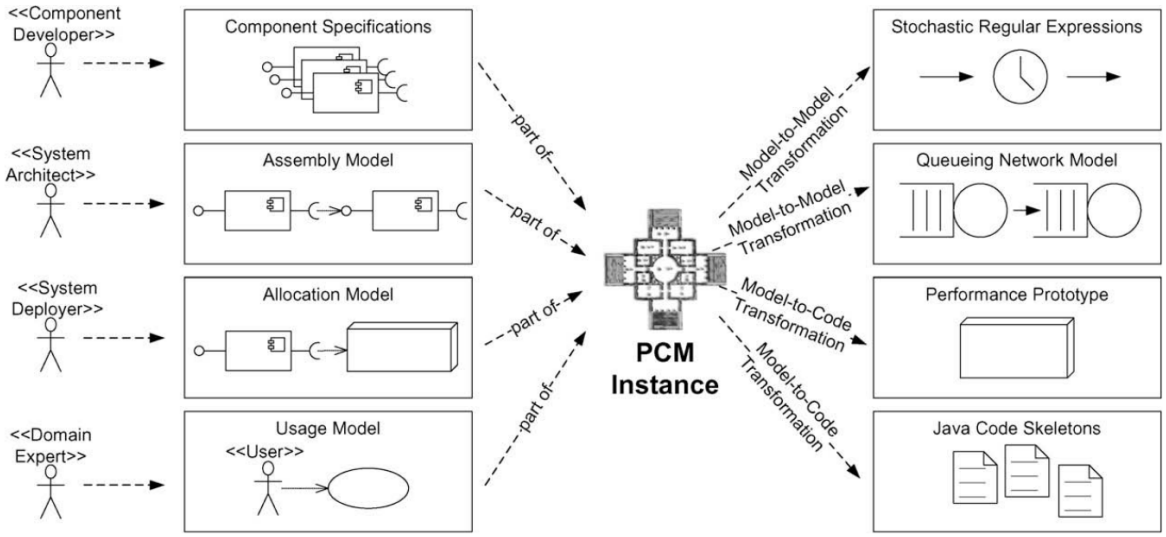


Figure 3.1: Process of Palladio [3]

3.2 MONARC

The MONARC project was started 1998 to help develop the initial design of the WLCG at CERN [7]. To simulate possible structures of the WLCG they created the MONARC simulator. The results of the simulations indicated that a hierarchical structure with regional centres performs best, resulting in the three Tier structure of the WLCG today [11]. After its initial results, the MONARC project proposed to model, prototype, and optimise the design of the overall distributed computing and data handling systems for the LHC experiments [7]. However, the project was discontinued in 2000.

In 2004 a second version of the MONARC simulator was released with several improvements, such as simulation of data replication. MONARC2 focuses on simulating experiments at CERN, but can also be used to simulate general large scale distributed computing systems [8]. A major difference to its first version is that a discrete-event approach instead of a discrete time-stepped is used, which results in a better performance.

In 2010 Zach et al. [19] used MONARC2 to simulate the ALICE Tier-2 site in Prague. The simulation included 3000 CPUs, and the CPU as well as the network and I/O resources were simulated. MONARC2 delivered accurate simulation results, however two problems occurred. Firstly, they were not able to simulate the network load correctly and secondly MONARC2 revealed performance issues, making it not usable for even larger systems.

3.3 CACTOS

CACTOS is an approach to cloud infrastructure automation and optimization, which started in 2014 as a project funded by the European Union [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 centre deployment scenarios and resource management algorithms.

The Prediction Toolkit is built on Palladio and SimuLizar, see section 3.1 for details on Palladio. The models used by the Prediction Toolkit regarding resource or application behaviour are updated using monitoring data to handle the highly dynamic behaviour of clouds. The idea to couple monitoring and grid simulation tools has been around for a while, suggested by Pop et al. [13] in 2006.

CACTOS supports the rapid testing of resource management algorithms due the ability to simulate them without the need to reimplement the algorithms against the Prediction Toolkit. Most other simulators require the reimplementation of a resource management algorithm to its specific interface. Furthermore, it allows the automatic creation of workload models using runtime monitoring data [18].

First case studies show that the Prediction Toolkit has a high accuracy, but it was not tested on large systems yet [18]. Nevertheless, the aim of CACTOS is to simulate cloud infrastructures, which leads to many features that are not needed to simulate GridKa. This feature richness would lead to big overhead when implementing load balancing strategies and worse performance regarding simulation time. Thus, CACTOS is not the best choice to model and simulate GridKa.

3.4 SimGrid

SimGrid is a framework for simulating large-scale distributed systems and one of the earliest created grid simulators [6]. Its first release was in 1998 and is one of the few that is still developed today. Originally, it started as a grid simulator, but it became more versatile. Therefore it can be used as a grid, P2P, MPI, or cloud simulator today. Moreover, it is one of the most used grid simulators [6].

In SimGrid the platforms and the deployment are defined in XML, and the application behaviour and scheduling is implemented using C. It supports simulation of CPU, I/O and network resources, heterogeneous workloads, and heterogeneous platforms [6].

It is more scalable than GridSim, which is another popular grid simulator [6]. This performance difference is created by several optimizations, including implementing light-weight execution contexts, using lazy activity updates, and using trace integration for resource management [6].

However, without the help of graphical model editors the models of the computing jobs are time-consuming to create. The support of graphical editors is a big advantage of Palladio.

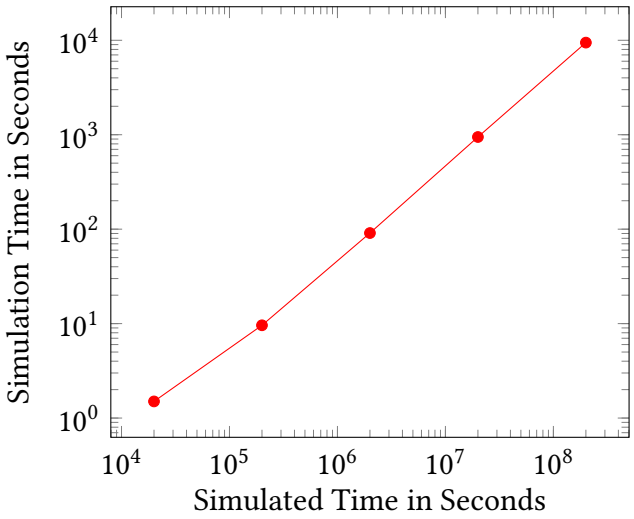
4 Exploratory Work

We chose Palladio as the tool to model and simulate the performance of computing jobs executed at the Tier 1 centre GridKa. Therefore, the first step was to learn how to work with Palladio and explore its possibilities. As a result of this process, the initial prototype of the model was created and the required improvements of Palladio were found. This initial model will be refined during the project and the different planned improvements are described in chapter 6.

Furthermore, we gathered performance data about the computing jobs and the resource environment at GridKa. First of all, these data was analysed to assess its usefulness and verify that they contain all required information. The analysis showed that all required information about the resource environment is available (CPU, I/O and network performance). In addition, the computing job data contain nearly all required information (CPU usage, job type, job start time and job run time), unfortunately the I/O usage was missing. Because the data contained nearly all required information we were able to calibrate the model and are working on receiving the I/O usage of the jobs, making the calibration more accurate.

One key question is if Palladio and SimuLizar are scalable enough to simulate such large systems in a satisfying time. Figure 4.1 shows the required simulation time for the initial model. It can be seen that even for long simulated times the required real time is acceptable. These results suggest that Palladio will be scalable enough to simulate GridKa and even bigger parts of the WLCG, thus we will continue to use Palladio as the tool to model and simulate the computing jobs at GridKa.

Figure 4.1: Simulation Times of Palladio



5 Goals and Methodology

The goal of our work is to model and simulate the performance of computing jobs executed at the Tier 1 centre GridKa for the CMS computing model. The simulation should be accurate enough to allow the evaluation of different load balancing strategies and design decisions. Finally, the model will be validated using real world performance data.

While creating the initial model several short comings of Palladio have been detected, which must be remedied to create accurate simulation results.

This leads to the following sub goals that are described in detail in chapter 6:

- Model heterogeneous resource environment: Currently it is not possible to model different types of resource containers. This is not accurately enough, instead each type of resource container at GridKa should have an own representation in the model.
- Duplicate measurement points: For each resource container the measurement points to measure the resource usage should be generated. It is not practical to create these measurement points manually for about 1000 resource containers.
- Model job slot concept: Each resource container can only run a certain amount of computing jobs in parallel, this is the amount of job slots a resource container has. We need to model this concept, because it is a foundation of the load balancing algorithm.
- Implement more realistic load balancing: Currently the load balancing algorithm randomly chooses the resource container. This does not reflect a realistic load balancing algorithm, therefore a load balancing algorithm based on available job slots should be implemented.
- Enable replay of traces: To be able to simulate and replay real world situations, Palladio should be able to replay traces of computing jobs at GridKa. This feature enables us to make the validation more accurate.
- Implement optional improvements: This sub goal contains several ideas on how the model could be improved. In contrast to the other sub goals these improvements are optional because their effects are expected to be not as big as the effects of the other sub goals.
- Validate model: The last goal is to validate the model. The validation consists of two parts. In the first part, the model is validated using real world data. In the second part, a change of the load balancing strategy will be evaluated.

6 Work Plan

Starting from the created initial model of the computing jobs at GridKa, Palladio and the model are improved. The initial model describes the CPU usage and the arrival frequency of the computing jobs. In addition, it also models the resource containers on a coarse-grained level.

This project is carried out in collaboration with Maximilian Stemmer-Grabow's bachelor's thesis *Calibrating Performance Models for Particle Physics Workloads*. In his bachelor's thesis, Stemmer-Grabow focuses on data extraction and model calibration, whereas this project focuses on remedying the short comings of Palladio.

In the following sections the different subtasks of this project are described. The time plan of the project is depicted in fig. 6.1.

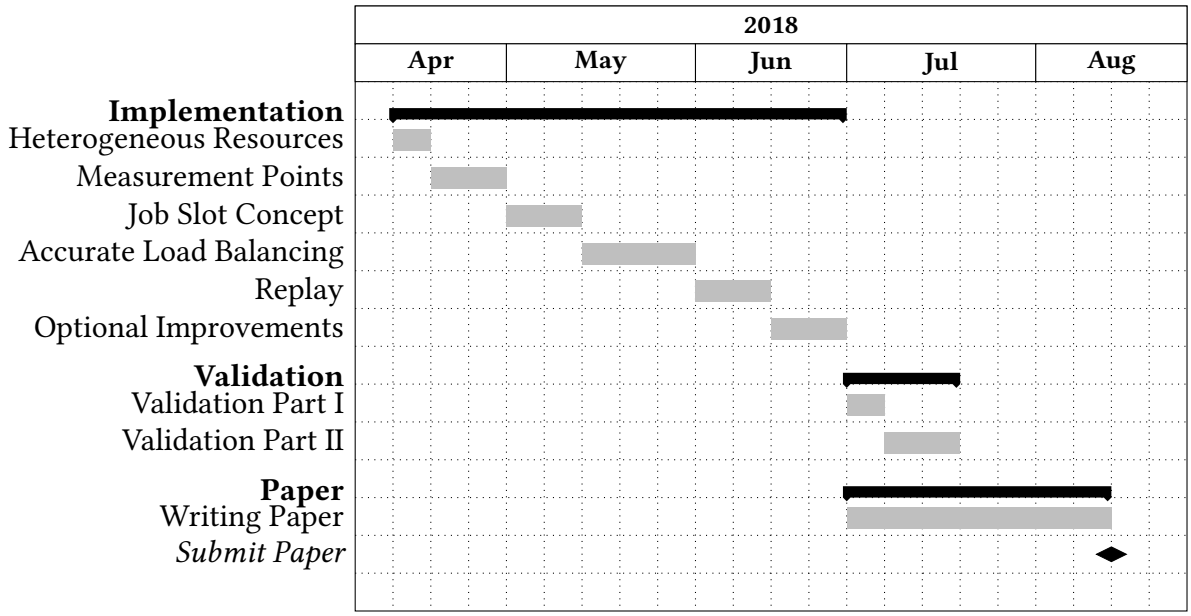


Figure 6.1: Time Plan

6.1 Heterogeneous Resource Environment

To represent the heterogeneous resource environment more accurately, the applied Architectural Template has to be modified. This template is applied to a resource container and specifies how many instances of this resource container should be generated in the completion phase. Currently the template can only be applied to one resource container, hence we can only model one type of resource containers. Our approach is to extend the template so it can be applied to multiple types of resource containers.

The result of this subtask is that each type of resource container at GridKa can be modelled as a resource container in Palladio and that the Architectural Template can be applied to each of these resource containers, resulting in a more accurate simulation. The expected amount of work for this subtask is about one week.

6.2 Measurement Points Duplication

Measurement points are needed to evaluate the resource usage of each resource container. To create them automatically the Architectural Template plugin should be extended. The developer specifies the measurement points for each type of resource container that should be duplicated. The effect of this feature should be that the measurement points for the duplicated resource containers are created due the completion phase.

Without this feature the validation cannot take place, because it is not practical to manually create the measurement points for about 1000 resource containers. This feature requires about two weeks of work.

6.3 Job Slot Concept

In the CMS computing model one resource container can only run a limited amount of computing jobs in parallel. This is called the amount of job slots of a resource container. The job slot concept is a foundation for the load balancing strategy, because the load balancing looks for available job slots to deploy the computing job. To realize this concept, a new Architectural Template has to be created that wraps an allocation context. When applying the template the amount of job slots is specified. Architectural Templates then generate a component that acquires a job slot before the wrapped component is executed and releases it after the execution is finished.

After modelling the job slots, a more realistic load balancing strategy can be implemented. It is expected to take two weeks to implement the job slot concept.

6.4 Accurate Load Balancing

The decision on which resource container a computing job runs highly affects the quality of the simulation. In the initial model the jobs are randomly deployed. This behaviour is not close to reality, instead a load balancing strategy should be implemented which considers the available job slots of a resource container.

This requires the implementation of a new Palladio plugin that realizes the load balancing mechanism and its configurable strategies. It is expected that this subtask requires the most work, with about 3 weeks.

6.5 Replay

The ability to replay traces of computing jobs is valuable because it enables one to replay concrete time-spans and situations. Furthermore, this makes the validation more accurate. To simulate the used resources of the resource containers, the input trace file contains the job type, the arrival time, and the required resources of a job.

To implement the feature the plugin mentioned in section 6.4 should be extended. The amount of work is estimated to be about two weeks.

6.6 Optional Improvements

This subtask contains several optional improvements which are expected to have not such a big impact on the quality of the model as the other subtasks, and thus are not as critical. This improvement includes the modelling of resource usage of other experiments at GridKa, data transfer from and to GridKa, modelling of individual phases of each job and modelling of pilot jobs. This list of optional improvements is not complete and could be extended due the project.

The method to approach this subtask is depicted in fig. 6.2. After implementing all other subtasks the model is compared with measured performance data. Based on these results the most promising improvement should be implemented. It is an iterative process, so after implementing one improvement the model is again compared with measured data to decide with which feature to continue.

The estimated time for this subtask is two weeks, however it will only be realised if there is enough time left, otherwise the time is used as a buffer.

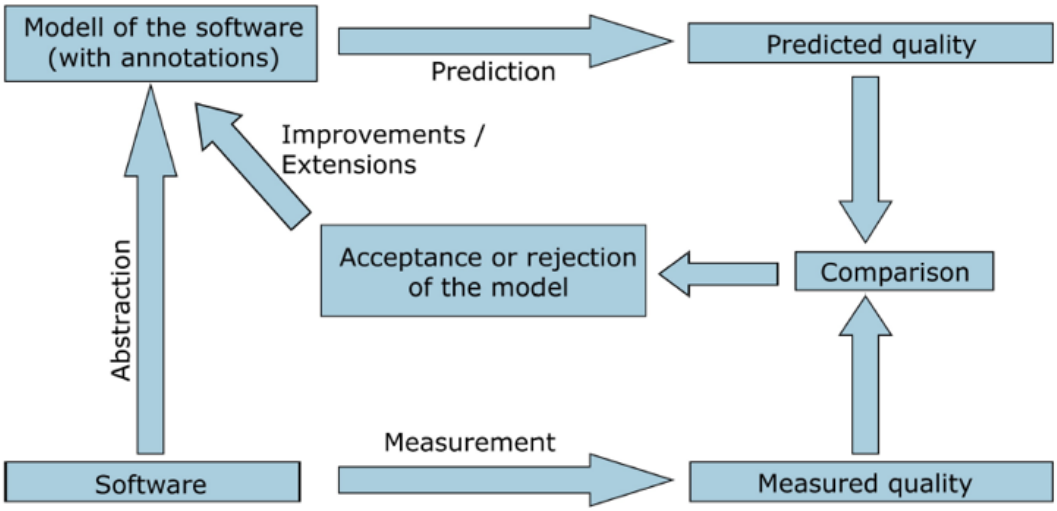


Figure 6.2: Scientific method: theory-driven empiricism [10]

6.7 Validation

The last subtask is the validation of the model to assess its quality. The validation consists of two parts; in the first part the model is compared with real world performance data to determine the differences of response times and resource usage. Thereby, it is determined, how well the model reflects the performance of the computing jobs and the current load balancing.

In the second part the load balancing strategy is changed and this new load balancing strategy is evaluated. The first part is expected to take one week and the second to take two weeks.

Bibliography

[1] Antonio Augusto Alves Jr et al. "A Roadmap for HEP Software and Computing R&D for the 2020s". In: *arXiv preprint arXiv:1712.06982* (2017).

[2] Matthias Becker, Steffen Becker, and Joachim Meyer. "SimuLizar: Design-Time Modeling and Performance Analysis of Self-Adaptive Systems." In: *Software Engineering* 213 (2013), pp. 71–84.

[3] Steffen Becker, Heiko Koziolek, and Ralf Reussner. "The Palladio component model for model-driven performance prediction". In: *Journal of Systems and Software* 82.1 (2009). Special Issue: Software Performance - Modeling and Analysis, pp. 3–22. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2008.03.066. URL: http://www.sciencedirect.com/science/article/pii/S0164121208001015.

[4] Ian Bird. "Computing for the Large Hadron Collider". In: *Annual Review of Nuclear and Particle Science* 61.1 (2011), pp. 99–118. DOI: 10.1146/annurev-nucl-102010-130059. eprint: https://doi.org/10.1146/annurev-nucl-102010-130059. URL: https://doi.org/10.1146/annurev-nucl-102010-130059.

[5] Ian Bird et al. *Update of the Computing Models of the WLCG and the LHC Experiments*. Tech. rep. 2014.

[6] Henri Casanova et al. "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms". In: *Journal of Parallel and Distributed Computing* 74.10 (June 2014), pp. 2899–2917. URL: http://hal.inria.fr/hal-01017319.

[7] Monarc Collaboration et al. *Models of Networked Analysis at Regional Centres for LHC experiments: Phase 2 report*. Tech. rep. Technical Report CERN/LCB-001, CERN, 2000. http://www.cern.ch/MONARC, 2000.

[8] Iosif C Legrand et al. "Monarc simulation framework". In: *International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Tsukuba, Japan*. 2003.

[9] Sebastian Lehrig and Matthias Becker. "Approaching the cloud: Using palladio for scalability, elasticity, and efficiency analyses". In: *Proceedings of the Symposium on Software Performance*. 2014, pp. 26–28.

[10] Anne Martens. *Exposé zum Promotionsvorhaben*. 2008.

[11] Youhei Morita, Monarc Collaboration, et al. "Validation of the MONARC simulation tools". In: *Computer Physics Communications* 140.1-2 (2001), pp. 153–161.

[12] P. O. Östberg 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*. Dec. 2014, pp. 26–31. DOI: 10.1109/CloudCom.2014.62.

[13] F. Pop et al. “A Simulation Model for Grid Scheduling Analysis and Optimization”. In: *International Symposium on Parallel Computing in Electrical Engineering (PARELEC’06)*. Sept. 2006, pp. 133–138. DOI: 10.1109/PARELEC.2006.8.

[14] María Alandes Pradillo et al. “Optimising costs in WLCG operations”. In: *Journal of Physics: Conference Series* 664.3 (2015), p. 032025. URL: http://stacks.iop.org/1742-6596/664/i=3/a=032025.

[15] WLCG Project. *WLCG REsource, Balance & USage*. 2017. URL: https://wlcg-rebus.cern.ch/apps/capacities/federations/ (visited on 11/2017).

[16] WLCG Project. *WLCG Tier Centers*. URL: http://wlcg-public.web.cern.ch/ (visited on 11/2017).

[17] WLCG Project. *WLCG Worldwide LHC Computing Grid*. 2017. URL: http://wlcg-public.web.cern.ch/ (visited on 11/2017).

[18] Christian Stier et al. “Rapid Testing of IaaS Resource Management Algorithms via Cloud Middleware Simulation”. In: *ACM / SPEC International Conference on Performance Engineering (ICPE’18)*. ICPE. 2018.

[19] Č Zach et al. “Simulation of the job processing performance at an ALICE Tier-2 site with MONARC”. In: *Journal of Physics: Conference Series* 331.7 (2011), p. 072038. URL: http://stacks.iop.org/1742-6596/331/i=7/a=072038.