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

State of the Art of

Patrick Firnkes

December 13, 2017

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

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

Contents

1 Motivation

The *Worldwide LHC Computing Grid (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 created data size steadily increases and reaches 50 Petabytes 2017, thus the grid has to be constantly extended and optimised to be able to analyse all data.

However, there is no model or simulation existing to evaluate the load balancing strategies of the grid, helping the operators to make the correct decisions in the optimization process.

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

1.1 Worldwide LHC Computing Grid

The WLCG processes 50 Petabytes of data each year created by the LHC at CERN [24] and with its help the *Higgs boson* was discovered in 2012 [27].

170 computing centres distributed in 42 countries are part of the WLCG and it combines 72.000 CPUs, 386 Petabytes online data space and 368 Petabytes nearline space (tape storage) [25]. Every day it runs 2 million jobs to analyse data of the CERN experiments.

At CERN there are four experiments, namely: Atlas, Alice, CMS, LHCb. They all use the grid in a similar, but not the same way [4]. Figure 1.1 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, do monte carlo production. 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 trending to use more 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.

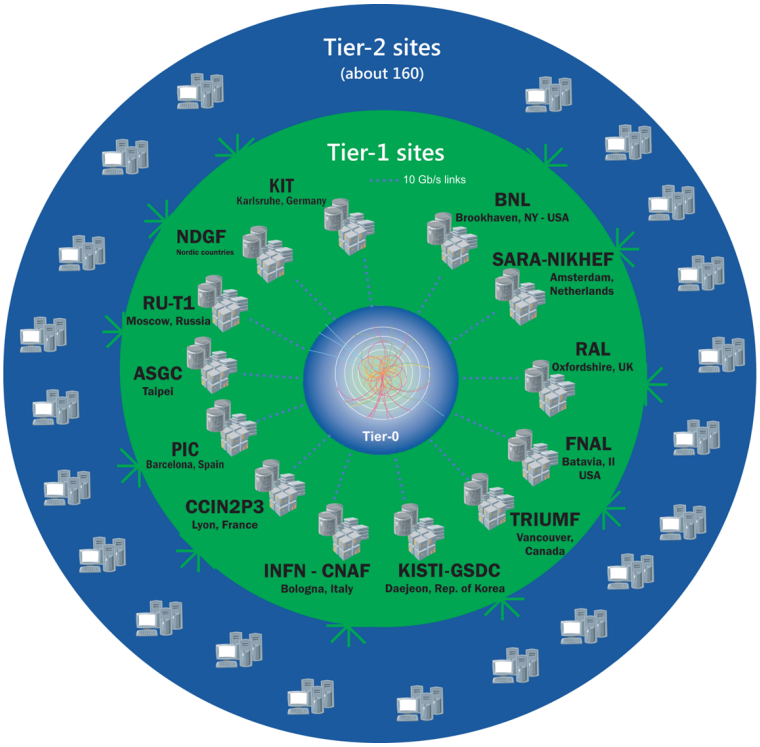


Figure 1.1: Tier structure of WLCG [26]

1.2 Current State

Currently, the load balancing strategy used at the *CMS computing model* is improvable, e.g. it does not account the nature of the jobs: *simulation jobs* require heavy I/O load and *analysis jobs* require heavy CPU time [33]. If this nature is not accounted 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 utilisation of I/O and CPU. For example, if too many simulation jobs are scheduled, the I/O is creating a bottleneck and the CPUs are idling.

HTCondor is the batch system used at the WLCG and is the central configuration point for load balancing [5]. It does not only decide on which site a job should run, but also on which node. This means it has full control over the load.

There are several other ideas how to improve the CMS computing model beside using a better load balancing strategy, but without a simulation it cannot be evaluated which one is the best in terms of speed-up, cost or utilization.

1.3 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 decide on a set of strategies which one is the best [23]. 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.

2 Foundations

In this chapter the *Palladio simulator* is described, which is intended to be used as the tool to model and simulate the load balancing strategies for the CMS computing model.

Another important foundation is to explain the differences between the related concepts of grid and cloud computing in order to classify the CMS computing model.

2.1 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 [2].

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 2.1 shows the modelling and simulation process.

Lehrig and Becker [18] extended Palladio with the *Architectual Templates* to make it possible to model cloud computing environments efficiently.

Furthermore, Palladio's simulation approach *SimuLizar* was created to simulate self-adaptive systems [1]. It was later extended to support the metrics scalability, elasticity, and efficiency [18].

The first case studies using Palladio in cloud environments show that it delivers accurate results [18]. However, Palladio was never used to model and simulate such a large system as the CMS computing model.

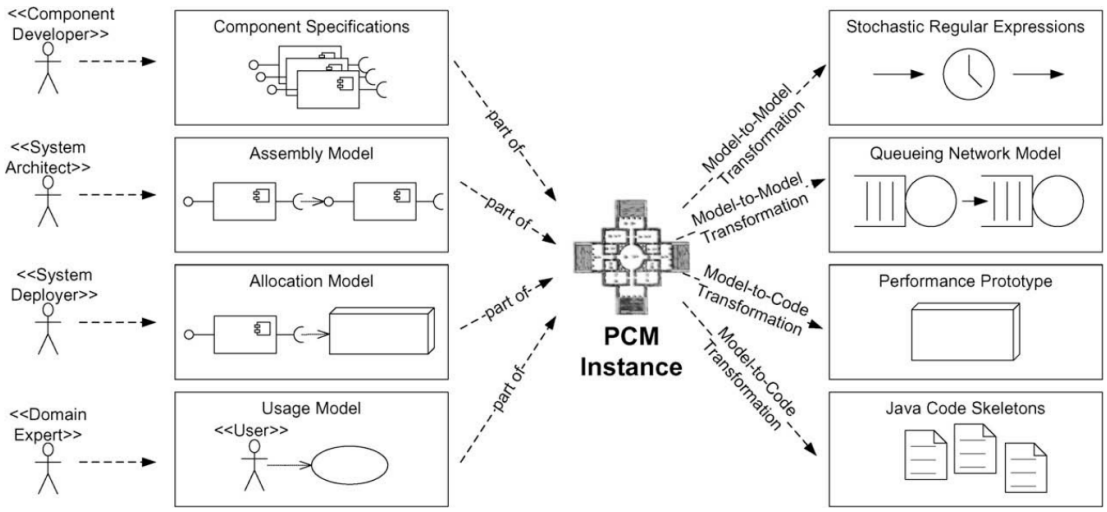


Figure 2.1: Process of Palladio [2]

2.2 Compare Grid and Cloud Computing

Grid computing are distributed systems working together to solve a problem [20]. *Cloud computing* is a large-scale distributed computing paradigm that is driven by economies of scale [12]. They have a lot in common, because cloud computing evolved from grid computing [12]. Both are scalable and involve multi tenancy. However, there are also some important differences.

Some of the key aspects are: *Resource pooling*, *on-demand resource provisioning*, and *elasticity* [12]. Firstly, cloud computing uses massively virtualization, this means that assigning application models to computing nodes do not model the abstraction correctly [7]. Instead the resources are pooled and different physical and virtual resources are dynamically assigned and reassigned depending to consumer demand, which makes auto scaling of applications possible [12]. Secondly, cloud computing allows on-demand resource provisioning [12]. Furthermore, clouds are elastic, they scale up or down according to the currently required resources.

From these aspects it follows that the CMS computing model is rather a grid than a cloud, because it does not do resource pooling, does not allow on-demand resource provision, and is not elastic. The given resources are assigned to a job and do not change till its completion, so there is no support of auto scaling [5]. The submitted jobs have to wait until resources are available, so there is no on-demand resource provision. Moreover, it is not elastic, because it does not create new sites if the load is high and does not shut down sites if the load is low [5].

3 State of the Art

There are a lot of other approaches beside Palladio that deal with the simulation of grids or clouds. In this chapter a selection of these is presented.

3.1 Grid Simulators

Grid simulators exist since the late 1990s, but the most ones have been created in the mid 2000s. However, most simulators are not developed any more. In this section the most common ones are presented.

3.1.1 SimGrid

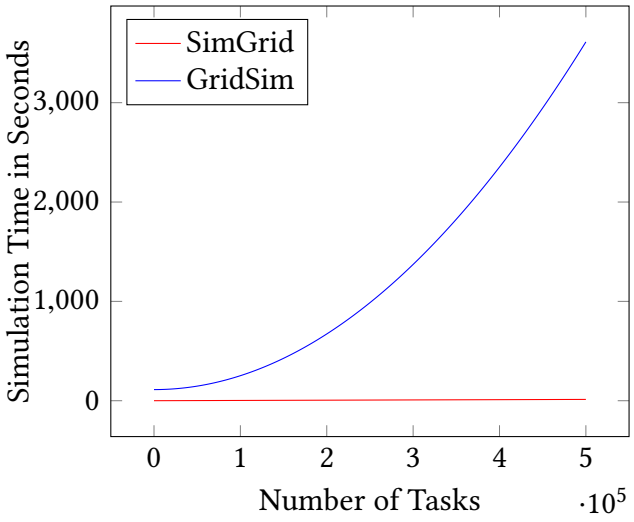
SimGrid is a framework for simulating large-scale distributed systems and one of the earliest created grid simulators [8]. Its first release was 1998 and is one of the few that is still developed today. Originally, it started as a grid simulator, but became more versatile, so that it can be used as a grid, P2P, MPI, or cloud simulator today. Moreover, it is one of the most used grid simulators [8].

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 [8].

It is more scalable than *GridSim*, which is another popular grid simulator [8]. Figure 3.1 shows the simulation times of SimGrid and GridSim for 2000 nodes and increasing tasks. SimGrid performs significantly better that GridSim, because GridSim's simulation time increases quadratic with the number of tasks, while SimGrid's increases linear. At 500.000 tasks GridSim requires more than one hour and 4.4 GiB of memory to simulate, while SimGrid needs less than 14 seconds and with only 165 MiB of memory.

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 [8].

Figure 3.1: Simulation Times of SimGrid and GridSim [8]



3.1.2 GridSim

GridSim was created 2001 at the University of Melbourne and was a popular simulator used by other members of the community until its development stopped 2010. It is focused on evaluating scheduling and resource brokering and supports simulation of CPU, I/O and network resources, heterogeneous workloads, and heterogeneous platforms [6]. GridSim is implemented in Java using SimJava's discrete event simulation [6]. Sulistio et al. [31] extended GridSim to make more realistic results regarding moving or the storage of data, which the original version struggled with [23]. Furthermore, they implemented the simulation of locality of data to be able to evaluate data replication strategies using GridSim.

However, as mentioned in section 3.1.1, GridSim does not scale very well.

3.1.3 GangSim

GangSim derived from an enhancement of the *Ganglia Monitoring Toolkit* in 2005 [10], but was not further developed after its initial release. It is used to handle usage constraints for sites and virtual organizations. GangSim evaluates allocation of individual resources from the interactions between allocation policies across virtual organisations [10].

It supports simulation of CPU, I/O and network resources, heterogeneous workloads, and heterogeneous platforms [10].

In difference to most other simulators, it is a discrete time-stepped simulator [15]. This means that it simulates every time slice, even if there is no activity in it. This results in a worse performance and scalability than most discrete event simulators.

3.1.4 OptorSim

OptorSim was created 2003 as part of the European Data Grid project and is used to evaluate data replication strategies in grids [3]. It simulates the time needed to copy or access the required files for each job and simulates the replication of files according to a given strategy.

OptorSim is not developed any more and does not consider job scheduling optimization. Because of its little scope of only evaluating data replication strategies, it does not support the simulation of CPU and I/O resources, or heterogeneous platforms [3].

3.1.5 DGSim

To improve simulation environments the *DGSim framework* focuses on automating and optimizing the overall simulation process [15]. It helps setting up the experiment by generating realistically grid infrastructure, grid dynamics and evolution events. Moreover, it either generates realistic grid workloads or uses traces of existing ones. It also enables grouped or batches of simulations [15].

DGSim was created in 2007 at Delft University of Technology and is not public available, thus nobody else beside the authors could use it.

3.1.6 ChicagoSim

Data locality is an important aspect in grid computing, thus *ChicagoSim* was created in 2003 to evaluate different scheduling and data replication algorithms [28]. In contrast to OptorSim, ChicagoSim does not only evaluate data replication algorithms, but also scheduling algorithms. One of the insights gained is that scheduling jobs to sites containing the data and simultaneously replicating popular data-sets to other sites is a good solution and performs better than scheduling jobs to idle sites and moving the data to them.

ChicagoSim was not further developed and does not support simulating CPU or I/O resources, instead simple constant waiting times are used without any concurrence of the underlying resources [28].

3.2 Cloud Simulators

In the recent years the research focus migrated from grid to cloud simulators, because clouds are more commonly used and grid simulators cannot sufficiently model the cloud infrastructure [19]. Although, as in section 2.2 shown, the CMS computing model is a grid and therefore a grid simulator is more suitable than a cloud simulator, it is useful to know about the latest developments in cloud simulators.

Hence, this sections presents a brief overview about the most common cloud simulators.

3.2.1 CloudSim

CloudSim was created 2010 at the University of Melbourne as part of the cloudbus project and is the most popular and sophisticated cloud simulator [19]. It is based on GridSim and is a foundation for a lot of other cloud simulators. It shows a good scalability, because its able to simulate 10.000 machines [7]. Though, this experiment did only create 50 VMs resulting in only 50 machines of the 10.000 being used to process the submitted tasks. This and the fact that the results have not been validated against measured data makes the scalability questionable.

CloudSim supports both system and behaviour modelling of cloud system components such as data centres, virtual machines and resource provisioning policies.

It was later extended with a network simulation component to evaluate applications that communicate with each other [13].

3.2.2 CDOSim

CDOSim was developed 2012 at Kiel University and is a simulator based on CloudSim to evaluate competing cloud deployment options (CDOs) [11]. It represents more the user than the provider perspective and hides fine fine-grained internals of a cloud platform. CDOSim helps to find the best ratio between high performance and low costs, by supporting the selection of a cloud provider, the mapping between service and virtual machines, and adoption strategies.

3.2.3 LocalitySim

Another simulator based on CloudSim is *LocalitySim*, which was created 2016 at the University of Cairo [14]. Data locality affects the performance of any scheduling algorithm. If the scheduler places the job far away from its data, extra time is needed to transfer data. Therefore, LocalitySim was created, because most other cloud simulators are not supporting the locality of data.

3.2.4 GreenCloud

GreenCloud focuses on the energy efficiency of clouds [16]. It was created 2012 at the University of Luxembourg and focuses specifically on the measurement of energy consumption of severs, switches and links. It simulates the network on a packet level, resulting in a very accurate network model.

However, it assumes that every server only has a single core and virtualization, storage area networks, and resource management are not considered. Therefore, the correctness of evaluating the trade-off between performance and energy consumption are questionable [29].

3.2.5 Cloud Autoscaling Simulation

Researchers at the Czech Technical University tried to evaluate predictive autoscaler using CloudSim, but found out that CloudSim had accuracy and speed issues [32]. Thus, they implemented a new simulation method based on *queueing models*, delivering realistic results.

One of their results was that autscaling is very stable and cost-efficient if a strategy is used that scales up based on latency and scales down based on utilization.

3.3 MONARC

The *MONARC project* was started 1998 to help with the initial design of the WLCG at CERN [9]. 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 [21]. 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 [9]. However, the project was not continued in 2000.

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 [17]. 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.

2010 Zach et al. [33] 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.

3.4 CACTOS

CACTOS is an approach to cloud infrastructure automation and optimization, started in 2014 as a project funded by the European Union [22]. 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 2.1 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. [23] 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 [30].

First case studies show that the Prediction Toolkit has a high accuracy, but it was not tested on large systems yet [30].

4 Discussion

In chapter 3 several grid and cloud simulators have been presented. Grid simulators fit our needs for modelling and simulating the CMS computing model better than cloud simulators, see section 2.2 for the differences of grid and cloud computing. Although it may be possible to use a cloud simulator, the additional functionalities to simulate clouds may create an overhead, affecting performance and comprehensibility of the simulation due to the required workarounds. Nevertheless, it seems that CloudSim is the most evolved cloud simulator.

Table 4.1 compares the presented grid simulators based on our requirements. As one can see only SimGrid and Palladio fulfil them, but it is not clear if Palladio is as scalable as required. Though, Palladio is made ready for the cloud as part of the CACTOS project, so we expect that it will fulfil our scalability requirement. All other simulators are either not developed any more or have never been intended to be used as a general grid simulator. Instead, they only have been created for the research of a small scope in grid computing. Most simulators support the simulation of CPU, I/O, and network resources. Also heterogeneous platforms and tasks are mostly supported.

As mentioned in chapter 3, GridSim, OptorSim, ChicagoSim, and LocalitySim are the only simulators that consider data locality. However, this feature is not required for simulating load balancing strategies, because we only need to simulate the effects of a data replication strategy, e.g. reduce the data miss rate by 30%. The data locality feature is needed to evaluate better data replication strategies.

Another use case that requires data locality is to use the simulation to evaluate concrete load balancing decisions instead of evaluating load balancing strategies. E.g. compare if a job should be submitted to Site A or B when Site A contains the required data. However, comparing concrete load balancing decisions using the simulation is out of scope and should be done in future work.

There was no indication that a similar large grid or cloud as the CMS computing grid was simulated yet, so our work will evaluate if the existing tools make it possible to simulate such large systems.

Table 4.1: Comparison of Grid Simulators

Simulator	Active Project	General usable	Job Scheduling	CPU	I/O	Network	Heterogeneous Platforms	Heterogeneous Tasks	Scalable
SimGrid	✓	✓	✓	✓	✓	✓	✓	✓	✓
GridSim	✗	✓	✓	✓	✓	✓	✓	✓	✗
GangSim	✗	✗	✓	✓	✓	✓	✓	✓	✗
OptorSim	✗	✗	✗	✗	✗	✓	✗	✓	?
DGSim	✗	✗	✓	✗	✓	✓	✓	✓	✓
ChicagoSim	✗	✗	✓	✗	✗	✓	✓	✓	?
MONARC	✗	✓	✓	✓	✓	✓	✓	✓	✗
Palladio	✓	✓	✓	✓	✓	✓	✓	✓	?

5 Conclusion

It was shown that Palladio is a decent choice for modelling and simulating the load balancing strategies of the CMS experiment. Another good candidate is SimGrid. However, if it shows up that Palladio is not scalable enough, model-to-text transformations can be used to transform Palladio models to SimGrid input files and use SimGrid to simulate the load balancing strategies. Also, we keep CloudSim in mind, because it is the most sophisticated cloud simulator.

Another insight gained is that the need for a simulation to optimize the WLCG was already present in the early 2000s. Nevertheless, the MONARC project did not continue its work, so no simulation is existing yet.

Furthermore, it was shown that neither the CMS computing experiment nor any other similar large grid or cloud system was simulated yet. Our work will create a model and a simulation magnitudes larger than the existing ones and will show if the simulators of today can be used to accurately model such large systems.

6 Next Steps

The first step will be to use Palladio to model and simulate the Tier 1 centre *GridKA* of the CMS computing model. To achieve this, performance logs have to be analysed to find out the different types of executed jobs, their frequency and resource requirements. Also, the resources of GridKA have to be determined. Once the model is created and calibrated with the measured performance values, the simulation can be executed and the results can be validated. At this stage, load balancing strategies of a single site could be simulated by adapting the workload of the site according to the expected effects of changing the global load balancing strategy.

The model of GridKA is then used as the foundation to model the whole CMS computing model. Finally, the load balancing of the CMS computing model can be simulated with all side effects between the sites.

Bibliography

[1] 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.

[2] 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.

[3] William H. Bell et al. "Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies". In: *The International Journal of High Performance Computing Applications* 17.4 (2003), pp. 403–416. DOI: 10.1177/10943420030174005. eprint: https://doi.org/10.1177/10943420030174005. URL: https://doi.org/10.1177/10943420030174005.

[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] Rajkumar Buyya and Manzur Murshed. "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing". In: *Concurrency and Computation: Practice and Experience* 14.13-15 (2002), pp. 1175–1220. ISSN: 1532-0634. DOI: 10.1002/cpe.710. URL: http://dx.doi.org/10.1002/cpe.710.

[7] Rodrigo N Calheiros et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". In: *Software: Practice and experience* 41.1 (2011), pp. 23–50.

[8] 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.

[9] 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.

[10] Catalin L Dumitrescu and Ian Foster. "GangSim: a simulator for grid scheduling studies". In: *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*. Vol. 2. IEEE. 2005, pp. 1151–1158.

[11] Florian Fittkau, Sören Frey, and Wilhelm Hasselbring. "CDOSim: Simulating cloud deployment options for software migration support". In: *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the*. IEEE. 2012, pp. 37–46.

[12] Ian Foster et al. "Cloud computing and grid computing 360-degree compared". In: *Grid Computing Environments Workshop, 2008. GCE'08*. Ieee. 2008, pp. 1–10.

[13] S. K. Garg and R. Buyya. "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations". In: *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. Dec. 2011, pp. 105–113. DOI: 10.1109/UCC.2011.24.

[14] Ahmed Hassan Abase, Mohamed Khafagy, and Fatma Omara. "Locality Sim : Cloud Simulator with Data Locality". In: 6 (Dec. 2016), pp. 17–31.

[15] Alexandru Iosup, Ozan Sonmez, and Dick Epema. "DGSim: Comparing grid resource management architectures through trace-based simulation". In: *European Conference on Parallel Processing*. Springer. 2008, pp. 13–25.

[16] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers". In: *The Journal of Supercomputing* 62.3 (2012), pp. 1263–1283.

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

[18] 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.

[19] Rahul Malhotra and Prince Jain. "Study and comparison of various cloud simulators available in the cloud computing". In: *International Journal* 3.9 (2013).

[20] Mahdi Mollamotalebi, Raheleh Maghami, and Abdul Samad Ismail. "Grid and Cloud Computing Simulation Tools". In: *International Journal of Networks and Communications* 3.2 (2013), pp. 45–52.

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

[22] 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.

[23] 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.

[24] 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.

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

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

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

[28] Kavitha Ranganathan and Ian Foster. "Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids". In: *Journal of Grid Computing* 1.1 (Mar. 2003), pp. 53–62. ISSN: 1572-9184. DOI: 10.1023/A:1024035627870. URL: https://doi.org/10.1023/A:1024035627870.

[29] Georgia Sakellari and George Loukas. "A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing". In: *Simulation Modelling Practice and Theory* 39.Supplement C (2013). S.I.Energy efficiency in grids and clouds, pp. 92–103. ISSN: 1569-190X. DOI: https://doi.org/10.10/j.simpat.2013.04.002. URL: http://www.sciencedirect.com/science/article/pii/S1569190X13000658.

[30] 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.

[31] Anthony Sulistio et al. "A toolkit for modelling and simulating data Grids: an extension to GridSim". In: *Concurrency and Computation: Practice and Experience* 20.13 (2008), pp. 1591–1609.

[32] T Vondra and J Šedivy. "Cloud autoscaling simulation based on queueing network model". In: *Simulation Modelling Practice and Theory* 70 (2017), pp. 83–100.

[33] 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.