MAGICIAN: Model-based design for optimizing the configuration of data-centers

Pablo C. Cañizares ★ Alberto Núñez ★ Juan de Lara ☆

★ Universidad Complutense de Madrid
Spain

☆ Universidad Autónoma de Madrd
Spain

Abstract

Designing data-centers that provide an acceptable cost-performance ratio is challenging. Generally, a wide spectrum of components must be previously analyzed, such as the kind of applications to be executed in the data-center, computing/storage requirements and the network topology, among others. Since each one of these components has a direct impact on the overall system performance, the design process is complex and difficult, which usually requires the intervention of an expert.

We propose a model-based approach to design data-centers. For this purpose, we have created a meta-model that describes the structure of data-center models. Then, a set of expert rules can be used to detect sub-optimal configurations, and (in some cases) correct the design. Data-center models can be simulated, to assess their performance and scalability, for which we use a code generator into the SIMCAN tool. We have implemented our approach as an Eclipse plugin, and illustrate the usefulness of some expert rules by showing the efficiency and scalability gains of the optimized model with respect to the original one.

1 Introduction

During the last decade, most efforts in scientific applications were focused in obtaining the best possible performance, exploiting the system resource usage both in supercomputers and commodity clusters.

Due to the high number of inter-related parameters that have a direct impact on the overall performance, building a system that provides the maximum performance for a given application is a very complex task. Designing and configuring a data-center that properly exploits the system resource usage may be a feasible task for an expert [1]. However, when the data-center is designed by a non-expert, it may provide an overall performance far from the expected one. Generally, a misconfiguration of the system architecture or a wrong choice of hardware resources, may lead to obtaining a poor performance.

Usually, the first step before deploying the data-center in a production environment consists in modelling and simulating its underlying architecture. Thus, the obtained results from the simulation are used to polish and improve the initial design. Unfortunately, the number of possible configurations is extremely large, making unpractical to model and simulate all of them.

In this paper we propose `MAGICIAN`, an approach that aids designers to optimize the configuration of data-centers. The main objective of `MAGICIAN` is to identify possible inconsistencies in the initial design of the data-center and to suggest feasible corrections. Thus, a reduced number of data-centers designs are generated, which can be simulated to analyze which one provides the best results.

The approach is based on model-driven engineering (MDE) [2]. We propose a meta-model for data-centers, so that data-center configurations are expressed as instances of such meta-model. We provide a library of expert knowledge rules to detect misconfigurations and suggest improvements on the design. Finally, we support the simulation of the data-center configuration to assess properties like scalability, and detect possible bottlenecks. The simulation is performed by generating code for the SIMCAN tool [3].

The rest of this paper is organized as follows. Section 2 provides an overview of the proposed approach. Section 3 describes in detail the principal components of `MAGICIAN`. Next, Section 4 presents performance experiments that show the usefulness of our approach. Section 5 presents related work, and Section 6 ends with the conclusions and future work.

2 Overview

In this section, we describe the `MAGICIAN` approach for optimizing the configuration of data-centers. The overall scheme is shown in Figure 1.

Our architecture represents data-center designs as models conformant to a meta-model, and includes two optimization loops. The first one is based on expert rules, which encode knowledge on typical good configurations. The second is based on simulation, with which one can analyse aspects like efficiency or scalability of the data-center models.

The structure of data-centers, and the integrity constraints that valid data-center models should obey, have been captured through a meta-model. The designer will be able to create models that *conform* to such meta-model. That is, these models use the types and relations defined in

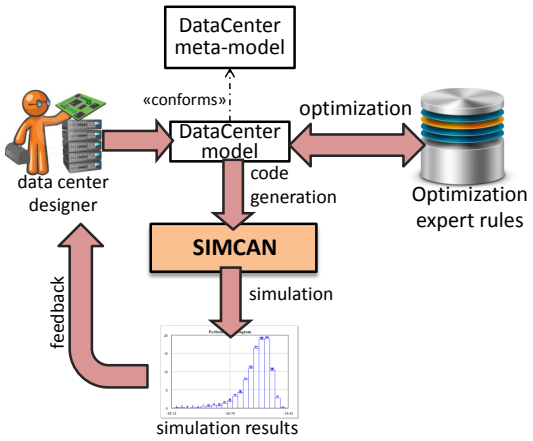

Figure 1. Working scheme of our approach

the meta-model, and satisfy the integrity constraints. This meta-model will be explained in Section 3.1.

We have created a library of expert rules, containing optimization patterns and idioms, typically followed by good data-center designs. These rules detect parts of the model that are amenable to optimization, signalling potential deficiencies in the model. Moreover, some of these rules contain quick fixes, which modify the design to improve some suboptimal aspect of the model. Technically, these rules have been implemented using the Epsilon Validation Language (EVL) [4]. They will be detailed in Section 3.2.

We also enabled the evaluation of the data-center model through simulation. This way, we have built a code generator that produces code to be executed by the SIMCAN simulation tool [3]. The results of the simulation can be used by the designer to find problems in the design, such as bottlenecks, to further improve it. The approach to code generation, and details on how the simulation is performed are given in Section 3.3.

3 Model-based simulation of data-centers

In this section we explain the three main building blocks of our approach: the data-center meta-model (see Section 3.1), the expert rules and quick fixes (see Section 3.2), and the code generation and simulation (see Section 3.3).

3.1 The data-center meta-model

Figure 2 shows a simplified version of the data-center meta-model. The *DataCenter* meta-class is the root class, which contains the main elements of a real data-center, such as those relating with both computational and networking aspects.

The computing elements are divided in two types. The first type corresponds to the *Node* meta-class, which represents a single computational node. The first 3 attributes define the CPU processor, where *CPU_Sockets*, *CPU_Cores* and *CPU_Speed* represent the number of CPUs, the number of cores of each CPU and the CPU speed (measured in MIPS), respectively. The next 3 attributes are related with memory features, where *RAM_slots*, *RAM_Size* and *RAM_Frequency* represent the number of memory modules, the total size of each module (measured in GBytes) and the frequency of the memory (measured in Mhz), respectively. The next 4 attributes refer to storage aspects, where *Disk_Slots*, *Disk_Size Disk_RBandwidth* and *Disk_WBandwidth* are the number of disks, the size of each disk (measured in GBytes) and the read and write bandwidth of the storage system (measured in Gbps), respectively. The last attribute, *isComputingNode*, denotes if a given node is a computing node or a storage node. The second type of computing elements corresponds to the *Rack* meta-class. A rack represents a structure that contains multiple computing elements. In this case, the rack consists of a set of *Boards*, where each board contains a number of nodes that is determined by the attribute *Nodes_per_board*.

The network is defined by 2 elements. The *Network* meta-class represents the communication network of the data-center, where *Bandwidth*, *Latency*, and *ErrorRatio* define the data transfer rate (measured in Gbps), the latency (measured in μs) and the error ratio of the network, respectively. The *Switch* meta-class represents a resource used to communicate the different computing elements of the data-center through the communication network, where *MTU* and *NumPorts* are the maximum transmission unit and the number of ports of the switch, respectively.

Finally, the *Repository* meta-class represents the data-center repository, which provides a wide collection of networking and computational components to model, with a high level of detail, a complete data-center.

Figure 3 shows an example of a data-center model which conforms the proposed meta-model. This model is inspired by a real IBM data-center configuration, which consists of 1 *IBM Flex* rack and 1 *IBM v7000 Storage* rack. These racks are interconnected using a 40/10 Gigabit Ethernet communication network and a *SAN42B-R extension switch*. The *IBM Flex* rack consists of 6 *Flex System Enterprise Chassis* boards, where each board contains 14 *IBM Flex System p460* computing nodes. These boards consist of 4 CPUs with 8 cores, reaching a speed of 317.900 MIPS. The memory system consists of 32 slots, which contain 64 GB of RAM. Finally, the storage consists of two disks of 2 TBytes. The *IBM Storage* rack consists of 6 *Flex System Chassis Storage* boards, where each board contains 14 *IBM v7000* storage nodes. Each node consists of 2 CPUs with 4 cores, reaching an speed of 200.000 MIPS, 16 GB of RAM and 16 hard disks with a total storage of 10 TBytes.

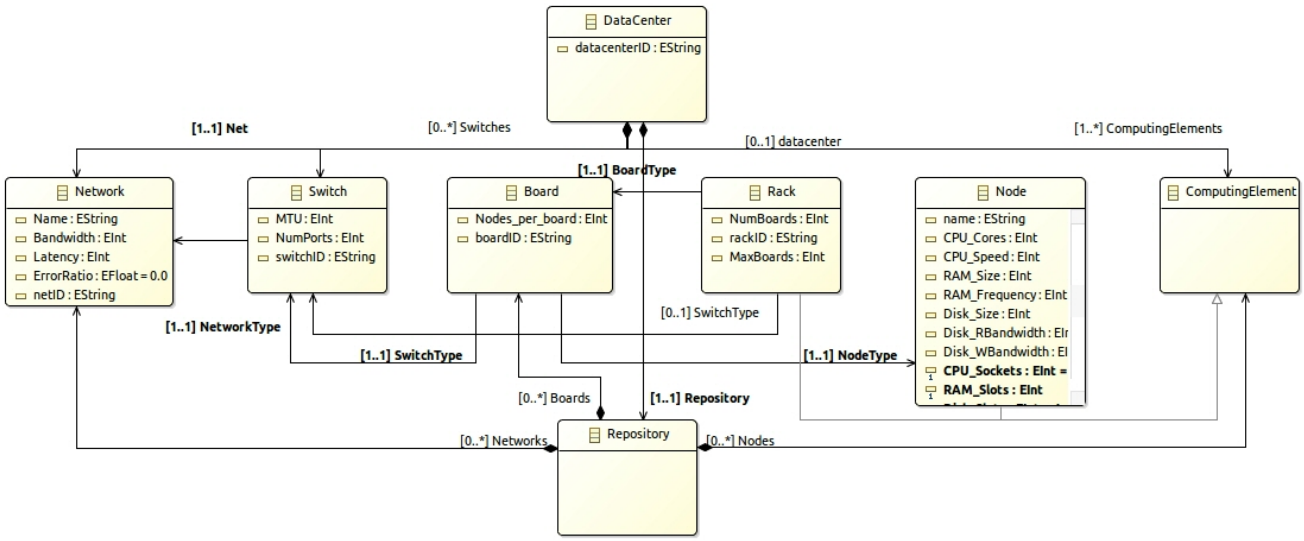

Figure 2. The data-center meta-model (excerpt)

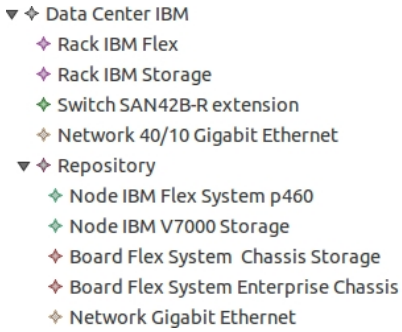

Figure 3. Example of a data-center model

3.2 Encoding expert rules and heuristic quick fixes

In order to support the user during the data-center design process, we have included a library of optimization rules based on data-center experts knowledge. These expert rules aid the user to solve possible design issues, which in most cases, hamper the overall data-center performance. However, expert rules must be designed and provided by an expert user, who must decide whether these are suitable to cover the requirements of the systems under study. The library consists of several rules focused on analysing different features of the data-center components, such as CPU processors, the memory system, storage and connectivity, among others. The main goal of these rules is to find inconsistencies in data-center models and to provide relevant information to fix them. For the sake of simplicity, in this paper we have described a sample of three rules from the complete library.

Listing 1 shows the expert rule *CoresVsStorageNodesRatio* encoded in EVL. This rule analyses the ratio between the number of storage nodes and the number of CPUs of a data-center. The main objective of this rule is to avoid system bottlenecks caused by a reduced number of storage nodes. In this case, if the available storage nodes are not able to provide the required performance, a message to modify the current design is shown. As can be seen, the rule is applied on the context of DataCenter objects (line 1 of the listing). It is made of a check section (lines 5–11), which evaluates a certain condition on the model, and a message part, which is presented to the designer if the check part returns true.

```
1  context DataCenter
2  {
3    critique CoresVsStorageNodesRatio
4    {
5      check{
6        var storageNodes: Integer;
7        var totalCores: Integer;
8        storageNodes = self.calculateStorageNodes();
9        totalCores = self.calculateTotalCores();
10       return storageNodes*40 >= totalCores;
11     }
12     message: 'The number of storage nodes must be increased, there
               exist a high number of cores in comparison with the number of
               storage node which can act as bottleneck'
13   }
14 }
```

Listing 1. Data-center topology optimization rule encoded in EVL

Listing 2 shows two expert rules based on the analysis of two network features, bandwidth and latency. In this case, these rules check that these features range in a determined interval. If some of these features is out of the range, the system provides a quick-fix method to alleviate the issue.

Quick fixes are specified in the fix section of the rules (lines 7–9 and 16–19). Figure 4 shows how such quick-fix is presented to the user.

```
1  context Network
2  {
3    critique NetBandwidth
4    {
5      check: self.Bandwidth>=10 and self.Bandwidth <=100
6      message: 'Network '+ self.Name + ': Bandwidth is usually ranged in
             [10−100]. Some of the most used configuration is 40'
7      fix {
8        title : "Set Bandwidth" + self.Name + " Bandwidth to 40"
9        do { self.Bandwidth = 40; }
10     }
11   }
12   critique NetLatency
13   {
14     check: self.Latency>=20 and self.Latency <=2000
15     message: 'Network '+ self.Name + ': Latency is usually ranged in
             [20−2000]. Some of the most used configuration is 200'
16     fix {
17       title : "Set Latency" + self.Name + " Latency to 40"
18       do { self.Latency = 200;}
19     }
20   }
21 }
```

Listing 2. Network optimization rules encoded in EVL

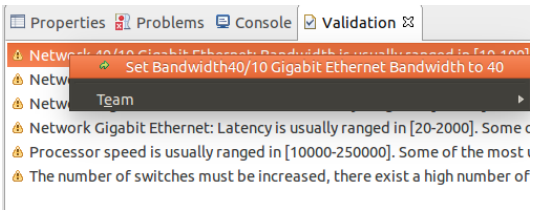

Figure 4. Example of network quick-fix

3.3 Code generation and Simulation

Once the data-center has been modelled, it can be simulated, to analyse its efficiency, in terms of scalability and performance. In this paper, we use the SIMCAN simulation platform to represent and simulate the behaviour of data-centers [3]. We have created a code generator that transforms the designed model into the required configuration files to perform the simulation.

Listing 3 shows an extract of the generated data-center topology, written in the NED language. The first line represents the name of the data-center, the next 3 lines refers to the different resources that compose the environment, such as switch, storage and computing nodes, respectively. Finally, the lines 6-8 show how the storage and computing elements are connected through the communication network, using the switch component.

```
1  network IBM{
2    switch_0:EtherSwitch;
3    rCmp1_IBM_Flex_Rack:Rack;
4    rSto0_StorageRack:Rack;
5
6    for i=0..5 {
7      rCmp1_IBM_Flex_Rack.ethg++ <−> Eth10M_channel <−>
           switch_0.ethg++;
8      rSto0_StorageRack.ethg++ <−> Eth10M_channel <−> switch_0.
           ethg++;
9    }
10 }
```

Listing 3. Example of data-center topology in SIMCAN written in NED language (excerpt)

Listing 4 shows an excerpt of a generated data-center configuration file. This portion of the configuration file configures the computing rack illustrated in Figure 3. It is important to remark that the symbol $*$ refers to a wildcard that represents all the elements in the referenced structure. For example, the lines 3 and 11 refer to the configuration of the network for all the boards in the rack.

```
1  IBM.rCmp1_IBM_Flex_Rack.numBoards = 6
2  IBM.rCmp1_IBM_Flex_Rack.nodesPerBoard = 14
3  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].channelType = "Eth10M"
4  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].cpuModule.CPUcore
       [*].speed = 79475
5  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].bsModule[*].disk.
       readBandwidth = 650.0Mbps
6  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].bsModule[*].disk.
       writeBandwidth = 420.0Mbps
7  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].osModule.memory.
       size = 2.0GiB
8
9  IBM.rSto0_StorageRack.numBoards = 1
10 IBM.rSto0_StorageRack.nodesPerBoard = 1
11 IBM.rSto0_StorageRack.nodeBoard[*].channelType = "Eth10M"
12 IBM.rSto0_StorageRack.nodeBoard[*].node[*].bsModule[*].disk.
       readBandwidth = 650.0Mbps
13 IBM.rSto0_StorageRack.nodeBoard[*].node[*].bsModule[*].disk.
       writeBandwidth = 420.0Mbps
14 IBM_ScenarioA_1server.rSto0_StorageRack.nodeBoard[*].node[*].
       osModule.memory.size = 2.0GiB
```

Listing 4. Data-center configuration in SIMCAN

4 Evaluation

This section presents a experimental study that shows the applicability of our our proposed approach. In order to carry out these experiments, a data-center inspired by IBM Flex system has been modelled (see Figure 2). In this case, the target system contains two racks and one main switch, that is, one computing rack for processing purposes and one storage rack for managing data. The computing rack consists of 6 board nodes, where each board contains 14 p460 blades with 4 CPUs, 64 GB of RAM memory and a local disk drive of 1TB. Hence, the modelled system provides a total of 336 CPUs. The storage rack has been modelled with one blade consisting of 2 CPUs, 32GB of RAM memory

and a high performance disk drive of 2TB. Each rack uses an Ethernet 10/100 network to interconnect the blades. The main switch is connected to each rack through an Ethernet 10-Gigabit network.

This data-center has been modelled using MAGICIAN, and the alternative designs have been simulated using SIMCAN[3], using the code generator explained in Section 3.3. In order to analyze the overall system performance, a Map-Reduce application has been used [5]. This application processes a 2.5GB data-set. This data-set is divided into small data portions, called domains, which are delivered among the different processes. In these experiments, 336 processes are executed in the available CPUs of the system. It is important to remark that each process has a dedicated CPU. The size of each domain is 4MB and the size of generated data, after processing each domain, is 2MB. Each process requires 1,875,000 MIs to process a single domain.

Once the data-center has been modelled, MAGICIAN detects 2 possible inconsistencies in the data-center configuration. The first inconsistency targets the infrastructure of the data-center (the rule detecting this issue is the one in Listing 1), while the second is related to a possible misconfiguration of a single parameter (see Listing 2).

In the first case, the storage rack has been configured to use 1 blade only, that is, 1 storage server. Generally, when the proportion between the number of processes and the storage resources is not properly balanced, the storage system acts as a system bottleneck, slowing down the overall system performance. Consequently, MAGICIAN suggests to increase the number of storage servers by modelling each board in the rack with different storage blades. In order to show the usefulness of this rule, different alternative configurations, using the expert rule described in Listing 1, have been generated. In this case, these simulations have been executed using 1, 2, 4, 8, 16 and 32 storage servers. Figure 5 shows the results obtained from these simulations, where the x-axis represents the number of storage servers and the y-axis represents the speed-up with respect to the initial configuration using 1 storage server. This chart shows that the overall system performance slightly increases when the number of storage servers increases as well. However, this chart also shows that there should be another bottleneck in the system, because the increase is not high.

In the second case, MAGICIAN suggests to change the configuration of the network. Since the original model uses a 10/100 Ethernet network in each rack, it may lead to slowing down the system significantly. Similarly, in this case we have simulated the data-center using a different number of storage servers and a 10-Gigabit Ethernet for communicating the blades in the racks. Figure 6 shows the results obtained from the simulation of the alternative data-center designs. This chart shows a significant increment of performance when the number of storage servers are increased.

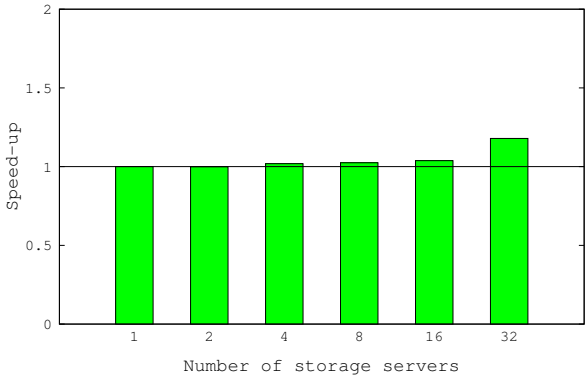

Figure 5. Proposed data-center designs by the expert-rule shown in Listing 1

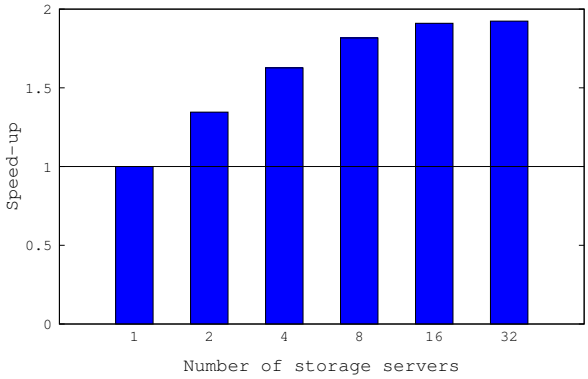

Figure 6. Proposed data-center designs by the expert-rule shown in Listing 2

As a conclusion, the initial configuration of the data-center had two drawbacks. First, using only 1 storage server limits the parallelism for accessing data in the system. In this case, MAGICIAN detects an inconsistency in the ratio between the number of CPUs and the number of storage servers. The expert rule suggests to increase the number of storage servers. In particular, this rule recommends to use 8 storage servers for this data-center configuration. Second, the network used in the racks acts as a system bottleneck. In this case, the issue is easily fixed by using the corresponding quick-fix.

5 Related work

The correct design of distributed systems is a process that requires years of expertise. In order to alleviate the inconveniences of this complex and costly task

the scientific community has performed a constant effort [6, 7]. Alshahrani and Peyravi presented a theoretical model to design and evaluate communication networks in data-centers [8]. This proposal includes an experimental analysis where the three major DCN architectures have been deployed by using simulation techniques.

In the field of modelling and simulation several contributions can be found. Son et. al presented CloudSimSDN [9], a simulation framework for software-defined cloud infrastructures. This framework incorporates a graphical interface to design the data-center topology. Meisner et. al presented [10], a simulation infrastructure for data-center systems. This approach is based on a higher level of abstraction, and uses a combination of queuing theory and stochastic modeling, which reduce the overall simulation time. Although these works allow to model several infrastructures in a fast and easy way, some of their main weakness are related with the low level of detail of the resultant infrastructures models. In order to alleviate these inconveniences, Nuñez et. al presented SIMCAN [3], a simulation platform designed to analyse and test parallel and distributed architectures and applications. In addition, a graphical user interface to help users without specific knowledge with the task of modelling new architectures is included.

More recently, Palyart et al. presented MDE4HPC [11], an model-based approach to describe and generate scientific knowledge for diverse architectures. This work presents a methodology to generate HPC applications independently from the platform by using Archi-MDE. Hence, to the best of our knowledge, there is no proposal to design data-center infrastructures that combines expert rules and simulations. Although there exist several simulation platforms, none of them includes users assistance during the modelling process. For this, our approach complements some of the existing simulation platform with expert-rules. In this case, we have selected SIMCAN due to its high level of detail and flexibility. In addition, this SIMCAN simulation platform is based on the OMNeT++, one of the most extended and adopted simulation platforms in the scientific community. Moreover, expert rules are expressed in EVL, which in its turn is based on OCL, a widely used standard for expressing model queries and constraints.

6 Conclusions and Future work

In this work we have presented MAGICIAN, a model-based approach for the design and analysis of data-center configurations. The metodology relies on expert rules to detect and fix suboptimal decisions, and on simulation to analyse performance and scalability of the configurations.

We have performed several experiments by modelling a real data-center using MAGICIAN. The proposed data-center designs, after applying the suggestion made by MAGICIAN, show that the existent inconsistencies in the initial design are fixed. Also, the new designs provide an overall system performance higher than the initial model.

In the future, we would like to support semi-automatic tuning configuration to reach a specific performance goal. We are also planning to identify recurring architectural patterns, which can be expressed as configurable templates.

Acknowledgements

Research partially supported by the Spanish MINECO projects DArDOS (TIN2015-65845-C3-1-R) and FLEXOR (TIN2014-52129-R), and the Comunidad de Madrid project SICOMORo-CM (S2013/ICE-3006).

References

[1] S. Tarapore, C. Smullen, and S. Gurumurthi, "Midas: An execution-driven simulator for active storage architectures," in *Workshop on Modeling, Benchmarking, and Simulation*, Beijing, 2008, pp. 1–10.

[2] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, ser. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2012.

[3] A. Nuñez, J. Fernández, R. Filgueira, F. García, and J. Carretero, "SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications," *Simulation Modelling Practice and Theory*, vol. 20, no. 1, pp. 12–32, 2012.

[4] S. Kolovos, R. Paige, and F. Polack, "Rigorous methods for software construction and analysis." Springer, 2009, ch. On the Evolution of OCL for Capturing Structural Constraints in Modelling Languages, pp. 204–218.

[5] A. Núñez, C. Andrés, and M. G. Merayo, "Optimizing the Trade-offs Between Cost and Performance in Scientific Computing," in *International Conference on Computational Science*, 2012, pp. 498–507.

[6] A. Sarkar, J. Guo, N. Siegmund, S. Apel, and K. Czarnecki, "Cost-efficient sampling for performance prediction of configurable systems (t)," in *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 342–352.

[7] N. Siegmund, A. Grebhahn, S. Apel, and C. Kästner, "Performance-influence models for highly configurable systems," in *10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015, 2015, pp. 284–294.

[8] R. Alshahrani and H. Peyravi, "Modeling and simulation of data center networks," in *Conference on Principles of Advanced Discrete Simulation*. ACM, 2014, pp. 75–82.

[9] J. Son, A. V. D., R. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2015, pp. 475–484.

[10] D. Meisner, J. Wu, and R. Wenisch, "Bighouse: A simulation infrastructure for data center systems," in *Int. Symp. Performance Analysis of Systems and Software*. IEEE Comp. Soc., 2012, pp. 35–45.

[11] M. Palyart, D. Lugato, I. Ober, and J. Bruel, "MDE4HPC: an approach for using model-driven engineering in high-performance computing," in *Integrating System and Software Modeling*, vol. 7083. Springer, 2011, pp. 247–261.