An AnyLogic Simulation Model for Power and Performance Analysis of Data Centres

Björn F. Postema(✉) and Boudewijn R. Haverkort

Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands
{b.f.postema,b.r.h.m.haverkort}@utwente.nl
http://www.utwente.nl/ewi/dacs/

Abstract. In this paper we propose a simulation framework that allows for the analysis of power and performance trade-offs for data centres that save energy via power management. The models are cooperating discrete-event and agent-based models, which enable a variety of data centre configurations, including various infrastructural choices, workload models, (heterogeneous) servers and power management strategies. The capabilities of our modelling and simulation approach is shown with an example of a 200-server cluster. A validation that compares our results, for a restricted model with a previously published numerical model is also provided.

Keywords: Data centres · Simulation · Discrete-event models · Agent-based models · Power management · Performance analysis · Power-performance trade-off · Cascading effect · Transient analysis · Steady-state analysis

1 Introduction

In 2012-2013, the global power consumption of *data centres* (DCs) was approximately 40 GW; this number is still increasing [7]. Hence, being able to evaluate the effect of energy-savings measures is valuable. One such energy-savings measure is *power management* (PM), which tries to lower the power state of servers, while performance is kept intact. Moreover, the so-called cascade effect (to be discussed later; cf. [8]) on energy consumption in infrastructure, strengthens the effects of PM strategies.

This paper aims to obtain insight in power usage and system performance (measured in terms of throughput and response times) in early DC design phases. It presents high-level models to estimate DC power consumption and performance. We will present and simulate cooperating models for (a) IT

B.F. Postema—The work in this paper has been supported by the Dutch national STW project Cooperative Networked Systems (CNS), as part of the program "Robust Design of Cyber- Physical Systems" (CPS).

B.R. Haverkort—The work in this paper has been supported by the EU FP7 project Self Energy-supporting Autonomous Computions (SENSATION; grant no. 318490).

M. Beltrán et al. (Eds.): EPEW 2015, LNCS 9272, pp. 258–272, 2015.
DOI: 10.1007/978-3-319-23267-6_17

equipment, (b) the cascade effect, (c) the system workload, and (d) power management. The value of our models is shown through the analysis and simulation of an example DC. Our models combine *discrete-event* models and *agent-based* models. Simulating these models sheds light on the above-mentioned power-performance trade-off. For the construction of our models, the multi-method simulation tool AnyLogic [1] is used. AnyLogic supports a mixture of three common methodologies to build simulation models: (a) *system dynamics*, (b) *process-centric/discrete-event* modelling, and (c) *agent-based* modelling. In this paper, we do not use system dynamics. Discrete-event modelling is a suitable approach for the analysis of systems that encompass a continuous process, that can be divided into discrete parts. Each part is characterised by triggering an event. As [15, p.6] states about discrete-event simulation:

> Discrete-event simulation concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. These points in time are the ones at which an event occurs, where an event is defined as an instantaneous occurence that may change the state of the system.

Agent-based modelling allows to model individual behaviour to obtain global behaviour with so-called communicating agents. It allows to easily specify heterogeneous populations. As [15, p.694] states about agent-based simulation:

> We define an agent-based simulation to be a DES where entities (agents) do, in fact, interact with other entities and their environment in a major way.

This paper contributes by taking the first steps towards accurate insight in both power and performance by presenting simple queueing models of IT equipment that are easy to extend and allow heterogeneity. Also, a model for the cascading effect is taken into account, and workloads can be based on general probability distributions or on measurement data. Moreover, the insight in power and performance has strong visual support for transient and steady-state analysis. Next steps that follow from this research involve refining and validation of models for more realistic case studies based on measurements and knowledge obtained from cooperation with the project partner Target Holding that allocated their IT equipment in the Centrum voor Informatie Technologie (CIT) data centre in Groningen, the Netherlands.

Over the last few years, various authors have proposed models for the analysis of the power-performance trade-off in data centres. Numerical solutions to compute power and performance for DCs based on Markov models have been proposed in [14], [9], [11], fluid analysis has been proposed in [17] and stochastic Petri nets in [16], [5], [12]. All these numerical approaches allow for the rapid computation of trade-offs, but are often limited in their modelling capabilities, thus leaving them useful for only few metrics under limiting assumptions. Simulation using AnyLogic, as we propose here, might be slower, however, it can handle a wider variety of DCs than numerical analysis and scales well to larger systems (as we will see).

The paper is further organised as follows. First, the DC and its context are described in Section 2. Section 3 continues from this system description by introducing all models, metrics and visualisation. A case study with a 200-server example and model validation are presented in Section 4, followed by Section 5 with the conclusions and future work.

2 System Description

In [2], important customer demands for DCs are distinguished, that direct choices on the system architecture, namely: *availability*, *scalability*, *flexibility*, *security* and *performance*. The minimum requirements for a server are *location*, *space*, *power supply*, *network accessibility* and *healthy environment conditions*. The demands from the customer and server requirements drive the choice of the most relevant components in a typical DC. Therefore, a data centre consist of various components, as described in [3], which are typically: Automatic Transfer Switches (ATSs), Uninterruptible Power Supplies (UPSs), Power Distribution Units (PDUs), servers, chillers, coolers, network equipment and devices for monitoring and control.

Through the network the DC becomes accessible from the outside world. The **workload** of a DC is the amount of work that is expected to be done by the DC. The workload of a DC is an important indication for functionality and efficiency. An indication of the workload in a DC is the number of *jobs* per time unit that arrive via the network, together with the length (distribution) of the jobs. Jobs sent through the network arrive in a buffer of a load balancer, that schedules the jobs. We assume that storage and network equipment guarantee negligible job losses in this buffer.

Energy consumption can be reduced in DCs in several ways [8]. One way is **power management** (PM), that aims to switch servers into a lower power state to reduce power consumption, while performance is kept intact. The challenge is to minimise the number of idle servers but prevent unacceptable performance degradation. Sometimes energy consumption reduces at the cost of performance, resulting in a trade-off. We will illustrate such trade-offs later in the paper.

3 Data Center Models

Section 3.1 presents an overview of all implemented agent-based models based on Section 2. These agent-based models are built from underlying queueing models, state-chart models and functions for analysis, which are detailed in Sections 3.2-3.5. Finally, power and performance metrics are presented in Section 3.6.

3.1 Model Overview

All relevant entities are modelled as agents, which enables easy extension towards heterogeneous entities. An overview of all agents is given in the UML diagram in Figure 1.

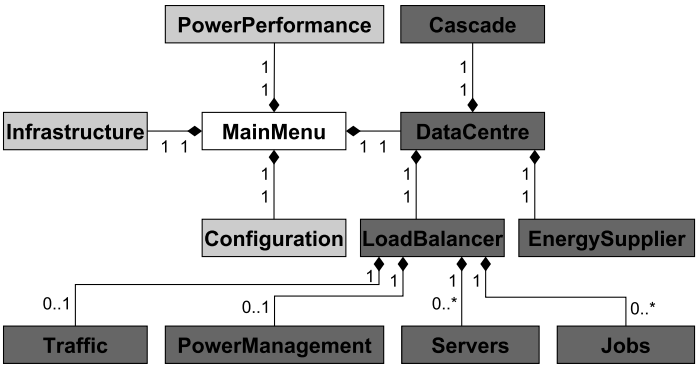

Fig. 1. All implemented agents in one UML diagram.

The `MainMenu` agent links to the agents `PowerPerformance`, `Infrastructure` and `Configuration` with visual representation of the results (light grey). The other agents, i.e., `DataCentre`, `Cascade`, `LoadBalancer`, `EnergySupplier`, `Traffic`, `Power Management`, `Servers` and `Jobs` are the DC models, including a visual representation (dark grey). In the upcoming subsections, the models inside these agents are discussed. The models inside the agent-based models are queueing models, state-chart models and functions for analysis.

3.2 IT Equipment Model

Jobs arrive in a queue in a load balancer. The load balancer decides to which server the jobs should be dispatched depending on the state information.

Figure 2 shows an $G|G|1|\infty|\infty$ queue of the load balancer. Jobs arrive in a FIFO buffer in the load balancer according to a general arrival process (left-most queue) and are served (big circle) in one of the M servers after injection of the job in one of the server queues and waiting for service there.

In order to compute response times, the `LoadBalancer` agent flags a job with a time stamp before it enters the load balancer queue. When a job is finished it compares the time stamp with its current time stamp to compute a response time sample.

Each `Server` agent comprises a $G|G|1|\infty|\infty$ queue with FIFO buffer. The jobs from the load balancer are injected and arrive at the server queue. At most one job at a time is served with a generally distributed service time (with mean value $1/\mu$). If a server has been switched off, then no jobs are routed to it.

The main reason for this modelling approach, instead of directly using an $G|G|M|\infty|\infty$ queue, is that any scheduling algorithm based on the state information of the server can be implemented in this framework, and it also allows for heterogeneous servers.

The *power state* of a server indicates how the server is used and how much power is consumed for that use. The server state can be described with

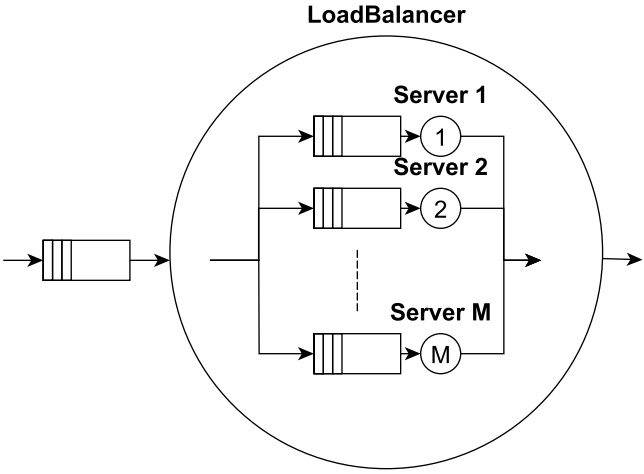

Fig. 2. Load balancer and servers queueing models.

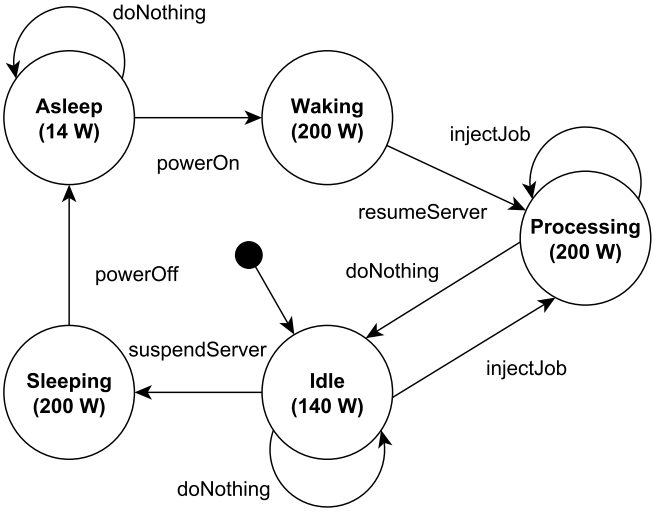

Fig. 3. State-chart model of server with sleep power states.

a state-chart model that switches between the low power consuming inactive `Asleep` state and the high power consuming active states `Idle` and `Processing`, that is controlled by external agents via messages; as depicted in Figure 3. Initially, the server is idle, i.e., the initial state is `Idle`. When the server is active, it can switch between the power state `Processing` (200 W) and `Idle` (140 W). When a server receives a sleep message, it first needs time to suspend the system

in power state `Sleeping` (200 W). After a generally distributed time with mean $1/\alpha_{\text{sl}}$, the server is in power state `Asleep` (14 W). Power state `Waking` (200 W), which takes extra time before the server starts processing the first job, i.e., after a generally distributed time with mean $1/\alpha_{\text{wk}}$ the server is back on. The cycle to shut down and boot a server follows the following sequence of power states: `Idle` (140 W) → `Shutting Down` (200 W) → `Off` (0 W) → `Booting` (200 W)→ `Processing` (200 W). The servers leave the power state `Booting` after a generally distributed time with mean $1/\alpha_{\text{bt}}$ and the power state `Shutting Down` after a generally distributed time with mean $1/\alpha_{\text{sd}}$. The power consumption values as used here are taken from [10].

The used power state model is highly abstract and could be refined, e.g., based on recent results for CPU-intensive workloads [13].

The currently implemented *job scheduling* depends on the power state of servers. Initially, a random idle server is selected. If no idle server is present, an off server is selected. In case only active servers are available, a random server is selected. Another variant of a scheduling mechanism is to inject a job in the server with the shortest queue. In case there are multiple shortest queues, a random server is chosen; such (and other) variants can all be easily implemented in our framework.

3.3 Cascade Model

The cascade effect, as elaborated on before, occurs in many DC infrastructure components that consume power based on server power consumption.

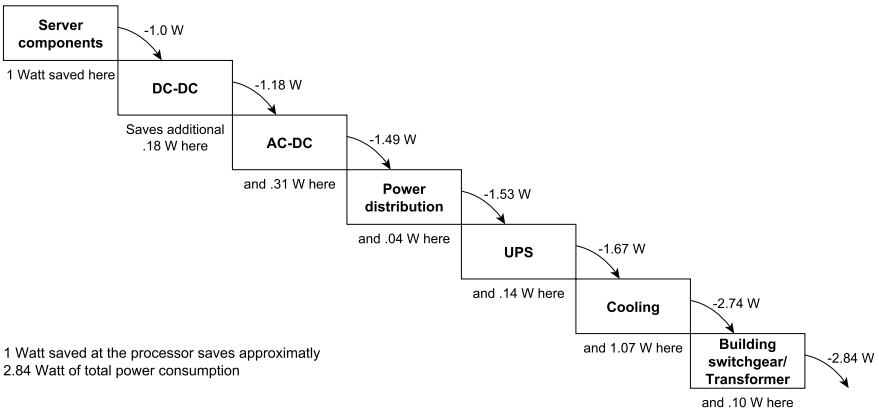

Fig. 4. EnergyLogic's cascade effect model.

The model for the cascade effect in DCs from [8], as depicted in Figure 4, is used in the `Cascade` agent. For each unit of power used by the servers, other DC infrastructure components, e.g., DC-DC, AC-DC, Power distribution, UPS, cooling, building switchgear/transformer "waste" power in a linear relation.

Hence, energy savings at the level of the server has great impact on the overall energy usage. The `Cascade` agent computes the power consumption metrics via simple linear functions.

3.4 Workload

Based on the description from Section 3.2, jobs enter the load balancer in a $G|G|1|\infty|\infty$ queue following a generally distributed inter-arrival time. In AnyLogic, the most common probability distributions are pre-implemented functions, e.g., exponential, normal, uniform and Erlang. The agent `Job` is added to the buffer after an inter-arrival time based on a function call that generates a random variable for the specified probability distribution. Additionally, in combination with the `Traffic` agent, custom discrete and continuous probability distributions can be defined using, e.g., frequency tables or observed samples. In this paper, we only discuss generally distributed times with time-constant means and jobs with fixed mean lengths, yet our simulation does allow time-varying means in order to support realistic time-varying workload with heterogeneous jobs obtained from measurements in data centres.

3.5 Power Management Strategies

Without application of PM, all servers in the DC are either processing or idle. PM, however, aims to switch servers into lower power states to reduce power consumption when the workload is low, while performance is kept intact. The `PowerManagement` agent has functions to decide when servers need to be put to sleep or even switched off, and when servers need to be switched on.

In order to demonstrate the capability of implementing strategies in our framework, two of the functions are illustrated here. Customers of DCs often demand a certain performance with a Service Level Agreement (SLA), e.g., the response time in a DC should never exceed 25 ms ($R_{\text{thres}} = 0.025\,\text{s}$).

The *threshold strategy* tries to stay as close to this response time as possible by putting servers to sleep until it gets too close to the threshold and servers are again woken. In more detail, the response time gets too close to the threshold when the latest observed sample exceeds 80 % of R_{thres}. Servers are put to sleep when the latest observed sample is lower than 60 % of R_{thres}. In future work, we will investigate more advanced threshold strategies, e.g., including hysteresis.

The aim of the *shut-down strategy* is to achieve a workload of all active servers that is equal to a pre-defined percentage, e.g., a server workload of 20 % means a server spends on average 20 % of the time processing, when jobs are equally scheduled among all servers. As a consequence, servers are shut down to achieve that goal. The only exception to this rule is when there are not enough servers in the DC.

3.6 Power-Performance Metrics

Quantitative metrics are used to provide insight into power and performance in DCs.

Power Consumption. An infrastructure component c has power consumption $P_c(t)$ (in Watt) at time t (in seconds). Power consumption $P_{\text{server}_i}(t)$ of server i depends on the server's power state. The total power consumption of K servers $P_{\text{servers}}(t)$ at time t:

$$P_{\text{servers}}(t) = \sum_{i=1}^{K} P_{\text{server}_i}(t). \tag{1}$$

The power consumption of other system components (like infrastructure), $P_{\text{other}}(t) = \sum_j P_j(t)$, where $j \neq \text{server}_i$ from all other components is computed through the cascade model. The total power consumption then equals the sum of power consumption by all components, i.e., $P_{\text{total}}(t) = P_{\text{other}}(t) + P_{\text{servers}}(t)$. The mean power consumption up to time t is computed as:

$$E[P_{\text{total}}(t)] = \frac{1}{t} \int_{x=0}^{t} P_{\text{total}}(x)dx. \tag{2}$$

Note that this integral is not explicitly computed, but that an efficient discretisation takes place. This discretisation takes full advantage of the fact that events trigger changes in the power consumption, i.e., there is a piecewise linear function for the power consumption over time. The mean power consumption up to time t, where k events occur at time $e_0, e_1, \ldots, e_k$ within the interval $[0, t]$ with a fixed first event $e_0 = 0$ and a fixed last event $e_k = t$, is computed as:

$$E[P_{\text{total}}(t)] = \frac{1}{e_k - e_0} \sum_{i=0}^{k} \int_{x=e_i}^{e_{i+1}} P_{\text{total}}(x)dx \tag{3}$$

$$= \frac{1}{e_k - e_0} \sum_{i=0}^{k} (e_i - e_{i-1}) P_{\text{total}}(e_i) \tag{4}$$

Response Time. This is the delay R_i (in ms) from the moment a job i enters until the moment it leaves the DC. So, each job will report its response time R_i. Given m observations, the mean response time is computed as:

$$E[R] = \frac{1}{m} \sum_{i=1}^{m} R_i. \tag{5}$$

Power State Utilisation. The power state utilisation $\rho_i(t)$ is the percentage of servers in a particular power state i at time t, with $\rho_i(t) \in [0, 1]$. The sum of all power state utilisations at time t is exactly 100 %, i.e., $\sum_i \rho_i(t) = 1$.

The mean power state utilisation up to time t is computed as:

$$E[\rho_i(t)] = \frac{1}{t} \int_{x=0}^{t} \rho_i(x)dx. \tag{6}$$

In practice, the integral is not explicitly computed, but an efficient discretisation takes place, similar as done for the mean power consumption. The mean power

state utilisation up to time t, where k events occur at time $e_0, e_1, \ldots, e_k$ within the interval $[0, t]$ with a fixed first event $e_0 = 0$ and a fixed last event $e_k = t$, is computed as:

$$E[\rho_i(t)] = \frac{1}{e_k - e_0} \sum_{i=0}^{k} \int_{x=e_i}^{e_{i+1}} \rho_i(x) dx \tag{7}$$

$$= \frac{1}{e_k - e_0} \sum_{i=0}^{k} (e_i - e_{i-1}) \rho_i(e_i) \tag{8}$$

3.7 Visualisation

The `PowerPerformance` and `Infrastructure` agents are implemented to show visuals and "live" values obtained from the simulation runs.

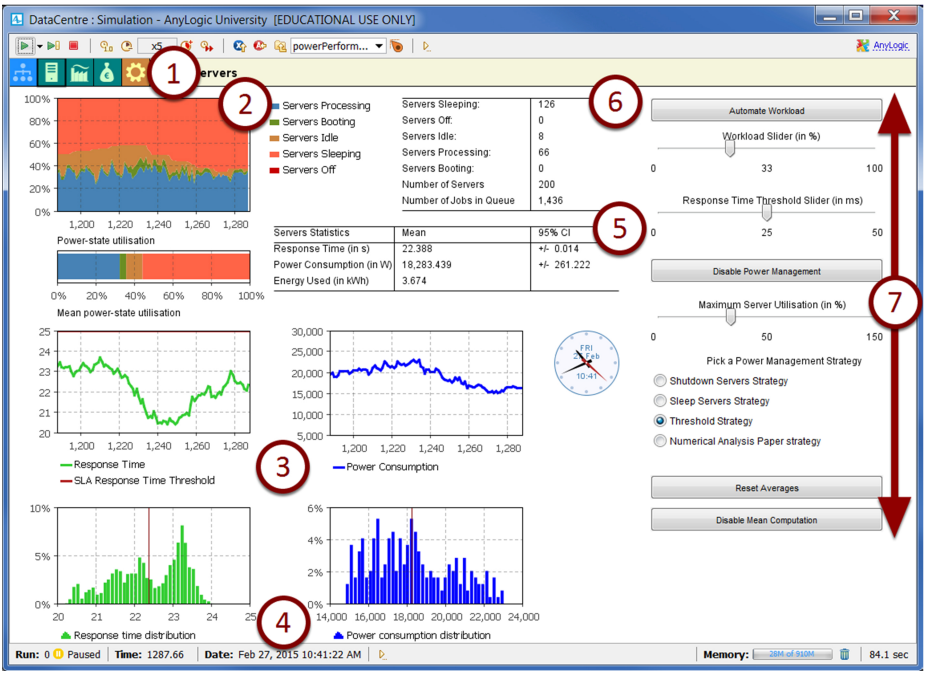

Fig. 5. The dashboard for the IT equipment.

Figure 5 shows an intuitive dashboard with results and configuration parameters of the DC model. The top line shows a menu bar with **(1)** links to the model, visuals and configuration. A cumulative utilisation plot **(2)** shows "live" how many servers are in each power state. A stack chart below this plot shows the mean cumulative utilisation, i.e., how many servers are in each power state. Furthermore, two time plots **(3)** show "live" power consumption (left) and live

response time (right) of the simulation. Two histogram plots **(4)** show the distribution of samples used to compute the means of power consumption (left) and response time (right). The values of the means are displayed in a small table including confidence intervals **(5)**; the exact way how these confidence intervals are computed is not clear (to us) from the documentation, hence, these should be handled with care. Table **(6)** shows the exact number of servers in each power state, the total number of servers in the DC and the total number of jobs in the queue(s). Configuration options **(7)** can be used to change the behaviour of the simulation on the fly: adjusting the server workload, the PM strategies, reset the averages and disable averages are the main configuration options. Additional configuration options are available in the `Configuration` agent, like changing the arrival, service, and booting time distributions.

4 Results

First, an example of a data centre with a 200-server computational cluster is elaborated to illustrate the capabilities of the simulation models in Section 4.1. Next, steps are taken for model validation by comparison of the results obtained from simulation to results obtained from models that are solved numerically in Section 4.2.

4.1 Case Study: Computational Cluster

We address a DC that needs to be installed with 200 servers. A *Service Level Agreement* (SLA) permits a response time of at most 25 s. Jobs are served, and, require on average 1 s service time. Furthermore, we require that at most 33 % of all servers are processing, which is not unusual [4]. Booting and shutting down of servers require exactly 100 s and going to sleep and waking up need only 10 s. The *Power Usage Efficiency* (PUE) of the DC is 1.5, i.e., 1 W saved at server level corresponds to 1.5 W saved in total; this is in line with the cascade effect model of Section 3.3. Furthermore, all the other IT equipment (that is, the non-servers) consume 1000 W, in total.

Table 1 shows an overview of workload (λ), service time distribution (μ), IT equipment specifications (mean booting time α_{bt}, mean shutting down time α_{sd}, mean sleeping time α_{sl} and mean waking time α_{wk} of servers), number of servers (n), PUE and power consumption by other IT equipment (P_{otherIT}). Figure 6 shows the power consumption in each power state, combined with a legend for time-cumulative utilisation plot for the shut-down strategy.

First assume that the exact workload is known at all times, and the shut-down strategy (as described in Section 3.5) is applied. Figure 7 shows transient behaviour in a time-cumulative utilisation plot. The x-axis represents the model time t (in s) and the y-axis shows the percentage of servers in each of the power states. The workload without PM is around 33 %. With PM switched on, 50 % of all servers is shut down, such that 66 % of all active servers are processing jobs.

Table 1. DC configuration and workload.

λ	exp(33.0)	μ	exp(1.0)
α_{bt}	det(100)	α_{sd}	det(100)
α_{sl}	det(10)	α_{wk}	det(10)
n	200 servers	PUE	1.5
$P_{otherIT}$	1000 W		

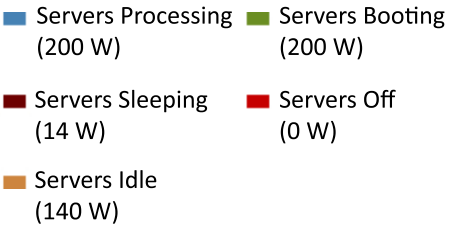

Fig. 6. Legend and power consumption in power-states.

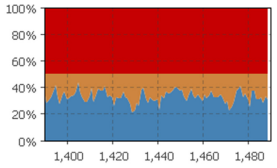

Fig. 7. Time-cumulative utilisation plot with shutdown strategy.

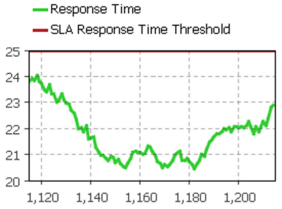

Fig. 8. Time-response time plot with threshold strategy.

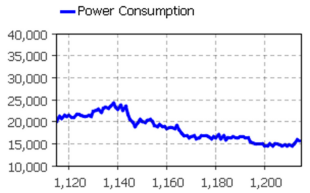

Fig. 9. Time-power consumption plot with threshold strategy.

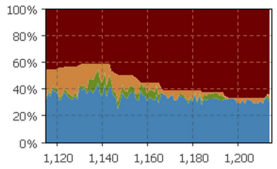

Fig. 10. Time-cumulative utilisation plot with threshold strategy.

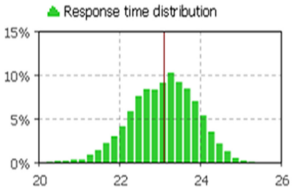

Fig. 11. Response time samples distribution with threshold strategy.

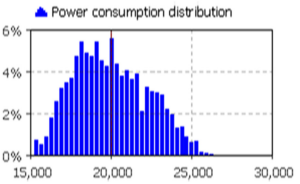

Fig. 12. Power consumption samples distribution with threshold strategy.

Furthermore, the mean power consumption is $\approx 18\,\text{kW}$ and the mean response time is $\approx 1\,\text{s}$.

In practice, the future workload is not exactly known. If workload prediction is inaccurate, late response of the PM strategy can dramatically increase the number of jobs in the system. Such situations have lead to worse performance, either by dropped jobs or large queues.

The threshold strategy (as described in Section 3.5) is based on response times rather than on the workload to control the power state of servers. For this strategy, the mean values are computed and time plots are generated (as can be seen from Figure 8–10). The mean response time $E[R] \approx 23\,\text{s}$ and mean power consumption $E[P_{\text{servers}}] \approx 20\,\text{kW}$.

Figure 8 shows a time-response time plot with again on the x-axis the model time t and on the y-axis a green line interpolating between the response time samples. A horizontal red line is drawn to indicate the response time threshold $R_{\text{thres}} = 25\,\text{s}$. Moreover, Figure 9 depicts a time-power consumption plot with model time t on the x-axis and a blue line that interpolates between power consumption $P_{\text{servers}}(t)$ samples on the y-axis. Furthermore, Figure 10 shows a time-cumulative utilisation plot. The x-axis represents the model time t (in s) and the y-axis shows the percentage of servers in each of the power states.

As seen in Figure 8–10, servers wake (for $t \in [1120, 1140]$), because the observed response times are approaching the threshold. Therefore, power consumption increases from $\approx 20\,\text{kW}$ to $\approx 25\,\text{kW}$ and the response time decreases from $\approx 24\,\text{s}$ to $\approx 21\,\text{s}$. The next step is to put servers to sleep again (for $t \in [1140, 1220]$), because the perceived response time is fine. As a consequence, response times increase again from $\approx 21\,\text{s}$ to $\approx 23\,\text{s}$, but power consumption decreases from $\approx 25\,\text{kW}$ to $\approx 15\,\text{kW}$.

4.2 Model Validation

For a simpler but very similar model, numerical solutions using stochastic Petri net (SPN) models have been presented in [16], also to compute mean response time and mean power consumption, again to analyse the power-performance trade-offs caused by PM (but no response time and power consumption distributions).

Table 2. DC configuration and workload.

λ	exp(1.0)	μ	exp(1.0)
α_{bt}	exp(0.01)	α_{sd}	n.a.
n	2-10 servers	β	exp(0.005)

In this paper, we compare the power-performance metrics obtained from our simulation DC models to similar metrics found in the numerical approach, that was presented in [16]. Therefore, the DC model is configured to exactly the same rates, power management strategy, number of servers and job scheduling as with the numerical solution. While this validation covers only a few scenarios, this comparison does show the feasibility of expressing models with the exact same data centre scenario that approach the same power and performance values.

Table 2 shows the configuration and workload. The Poissonian arrival rate $\lambda = 1.0\,\text{jobs/s}$, $\alpha_{\text{bt}} = 0.01\,\text{servers/s}$, and $\mu = 1.0\,\text{jobs/s}$. A special PM strategy is implemented with an exponentially distributed release time with rate $\beta = 0.005$ servers/s that determines the number of servers shutting down per second when idle; note that deterministic time-outs are not allowed in stochastic Petri nets, which explains why the time-out has been chosen like this with the numerical approach. The number of servers is scaled from 2 to 10. Time spend on shutting down a server is ignored.

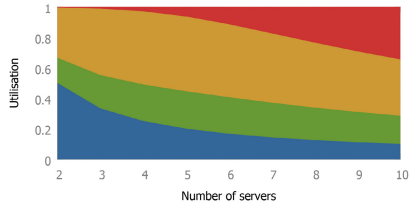

Fig. 13. Cumulative utilisation plot when scaling the number of servers for numerical analysis.

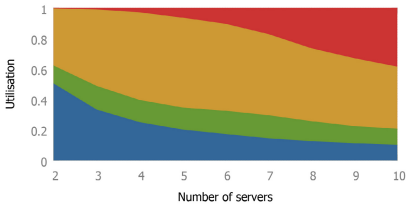

Fig. 14. Servers-cumulative utilisation plot when scaling the number of servers for simulation.

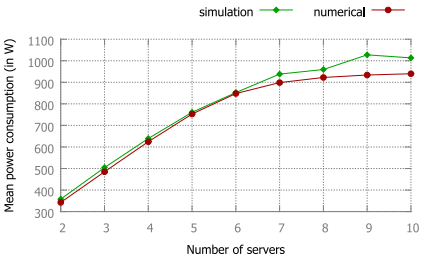

Fig. 15. Mean power consumption for various number of servers for simulation and numerical analysis.

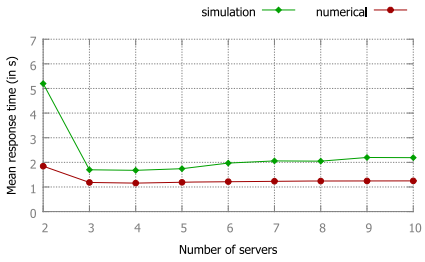

Fig. 16. Mean response time for various number of servers for simulation and numerical analysis.

Figure 13 and Figure 14 show cumulative power state utilisation plots for the servers with the PM strategy, for respectively the SPN-based numerical analysis and our simulation. The x-axis represents the number of servers n and the y-axis shows the percentage of servers in each of the power states (from top to bottom: red = off, orange = idle, green = booting and blue = processing). The plots confirm each other as the plots approach similar shape, but different values; the plots are not completely the same, which is partly the case due to the fact that we run a stochastic simulation, which, in essence, is a statistical experiment. Another reason for the observed difference lies in the implementation of the job scheduling: the SPN-based models use only one general buffer, whereas our simulation models use a separate buffer per server.

Figure 15 shows the mean power consumption for various servers in a DC. The x-axis represents the number of servers n and the y-axis shows mean power consumption (in W). The curves for simulation and numerical analysis have similar shape, but with different values, which range, respectively, from 353 W and 342 W for 2 servers to 982 W and 939 W for 10 servers.

Figure 16 shows the mean response times for various servers in a DC with the PM strategy. The shape of both curves are again very similar, but with different values, which are for simulation and numerical analysis respectively from 5.2 s and 1.84 s for 2 servers to 2.19 s and 1.25 s for 10 servers.

Another reason, for the different values in the curves, is that numerical analysis has no load balancer, but an implicit way for scheduling jobs. First, jobs are scheduled to a random idle server with both numerical analysis and simulation. Otherwise, the jobs are scheduled to a random off server. If all servers are booting or processing, numerical analysis keeps the jobs in the buffer and simulation inject the job in a random server.

5 Conclusions and Future Work

The contribution of this paper is the presentation of a new AnyLogic-based tool with an intuitive dashboard, effective for obtaining quick insights in transient and steady-state behaviour of heterogeneous DC with any possible workload and PM strategies. Furthermore, the AnyLogic environment enables to easily extend, refine and adapt DC models to many other scenarios.

Insight is obtained in the power and performance in DCs with varying number of servers, PM strategies and workloads. Relevant metrics are derived from the qualitative DC demands, including power consumption, response time and power state utilisation. These metrics are estimated by gathering samples from a mixture of discrete-event and agent-based models for IT equipment, PM and workload, implemented in AnyLogic. Furthermore, a cascade model enables the computation of total power consumption. Our approach is illustrated with a 200-server case study.

A well known open-source toolkit CloudSim [6] allows to simulate cloud computing scenarios and allows to specify (textually) DC models with virtual machines, applications, users, scheduling and provisioning. This tool obtains utilisation, response times, execution times and energy consumption metrics from simulation runs. It is future work to to investigate the capabilities of CloudSim in comparison to our AnyLogic-based simulation models.

Foreseen future extensions to the presented models are, among others, (i) the analyses of other PM strategies, e.g., based on number of jobs in the system and on hysteresis based strategies; (ii) energy-efficiency measures based on dynamic voltage and frequency scaling; (iii) power consumption of scaling workload; (iv) large-scale DC setting with heterogeneous servers, and a mixture of job sizes and inter-arrival times; (v) virtualisation; and (vi) thermal-aware DCs. Other future work includes the validation of the models with actual measurements from a DC.

References

1. AnyLogic: AnyLogic: Multimethod Simulation Software (2000). http://www.anylogic.com/
2. Arregoces, M., Portolani, M.: Data Center Fundamentals. Cisco Press, Indianapolis (2003)
3. Barroso, L.A., Hölzle, U.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Synthesis Lectures on Computer Architecture 8(3), 1–154 (2009). http://www.valleytalk.org/wp-content/uploads/2013/10/WSC_2.4_Final-Draft.pdf

4. Birke, R., Chen, L.Y., Smirni, E.: Data Centers in the Wild: A Large Performance Study. Tech. rep., IBM Research (2012). http://domino.research.ibm.com/library/cyberdig.nsf/papers/0C306B31CF0D3861852579E40045F17F/File/rz3820.pdf
5. Bruneo, D., Lhoas, A., Longo, F., Puliafito, A.: Analytical Evaluation of Resource Allocation Policies in Green IaaS Clouds. In: Proc. of 3rd Int. Conf. on Cloud and Green Computing, pp. 84–91. IEEE Computer Society Washington, DC (2013)
6. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience **41**(1), 23–50 (2011)
7. Datacenter Dynamics Intelligence: Global Data Center Power 2013. Tech. rep., Datacenter Dynamics (2013). http://www.dcd-intelligence.com/Products-Services/Open-Research/Global-Data-Center-Power-2013
8. Emerson Network Power: Energy Logic: Reducing Data Center Energy Consumption by Creating Savings that Cascade Across Systems. White Paper of Emerson Electric Co (2009)
9. Gandhi, A.: Dynamic Server Provisioning for Data Center Power Management. Phd thesis, Carnegie Mellon University (2013). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.376.4361
10. Gandhi, A., Doroudi, S., Harchol-Balter, M., Scheller-Wolf, A.: Exact Analysis of the M/M/k/setup Class of Markov Chains via Recursive Renewal Reward. In: Proc. of the ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems, vol. 41, pp. 153–166. ACM, New York (2013)
11. Ghosh, R., Naik, V.K., Trivedi, K.S.: Power-Performance Trade-offs in IaaS Cloud: A Scalable Analytic Approach. In: Proc. of 41st Int. Conf. on Dependable Systems and Networks Workshops, pp. 152–157. IEEE Computer Society Washington, DC (2011). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5958802
12. Katoen, J.P., Noll, T., Santen, T., Seifert, D., Wu, H.: Performance Analysis of Computing Servers using Stochastic Petri Nets and Markov Automata. Technical report, RWTH Aachen University (2013)
13. von Kistowski, J., Block, H., Beckett, J., Lange, K.D., Arnold, J.A., Kounev, S.: Analysis of the Influences on Server Power Consumption and Energy Efficiency for CPU-Intensive Workloads. In: Proc. of the 6th ACM/SPEC International Conference on Performance Engineering, pp. 223–234. ACM Press, New York (January 2015). http://dl.acm.org/citation.cfm?id=2668930.2688057
14. Kuhn, P.J., Mashaly, M.: Performance of Self-Adapting Power-Saving Algorithms for ICT Systems. In: Int. Symposium IFIP/IEEE on Integrated Network Management, pp. 720–723. IEEE (2013)
15. Law, A.M.: Simulation Modeling and Analysis, 5th edn. McGraw-Hill (2015)
16. Postema, B.F., Haverkort, B.R.: Stochastic Petri Net Models for the Analysis of Trade-Offs in Data Centres with Power Management. In: Klingert, S., Chinnici, M., Rey Porto, M. (eds.) E2DC 2014. LNCS, vol. 8945, pp. 52–67. Springer, Heidelberg (2015)
17. Stefanek, A., Hayden, R.A., Bradley, J.T.: Fluid Analysis of Energy Consumption using Rewards in Massively Parallel Markov Models. In: Proc. of the 2nd ACM/SPEC International Conference on Performance Engineering (ICPE), pp. 121–132 (2011)