Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

INNOVATIONS IN SIMULATION: EXPERIENCES WITH CLOUD-BASED SIMULATION EXPERIMENTATION

Simon J. E. Taylor
Anastasia Anagnostou
Nura Tijjani Abubakar

Modelling & Simulation Group
Department of Computer Science
Brunel University London
Uxbridge, UB8 3PH, UK

Tamas Kiss
James DesLauriers
Gabor Terstyanszky

Centre for Parallel Computing
School of Computer Science and Engineering
University of Westminster
London, W1W 6UW, UK

Peter Kacsuk
Jozsef Kovacs

Hungarian Academy of Science
Institute for Computer Science and Control
(MTA SZTAKI)
Budapest, 1111, HUNGARY

Shane Kite
Gary Pattison
James Petry

Saker Solutions Ltd
Upper Courtyard, Ragley Hall
Alcester, Warwickshire, B49 5NL, UK

ABSTRACT

The amount of simulation experimentation that can be performed in a project can be restricted by time, especially if a model takes a long time to simulate and many replications are required. Cloud Computing presents an attractive proposition to speeding up, or extending, simulation experimentation as computing resources can be hired on demand rather than having to invest in costly infrastructure. However, it is not common practice for simulation users to take advantage of this and, arguably, rather than speeding up simulation experimentation users tend to make compromises by using unnecessary model simplification techniques. This may be due to a lack of awareness of what Cloud Computing can offer. Based on several years' experience of innovation in this area, this article presents our experiences in developing Cloud Computing applications for simulation experimentation and discusses what future innovations might be created for the widespread benefit of our simulation community.

1 INTRODUCTION

In simulation we define a problem, build a model and then experiment with it to better understand the problem. For example, we might want to understand how throughput in a manufacturing system can be improved through changes in that system or how infection might be limited in a population through different lockdown strategies. We build a model of that system and then experiment with different parameters and configurations to see how these affect throughput or infection rate. In discrete-event simulation, each experiment typically has a number of replications associated with it. Let's say the model takes 10 minutes to simulate and we need 10 replications per experiment. In a simple case, if we have 10 parameters with 10 values each then we have 100 experiments. With 10 replications each experiment we have 1000 runs. 10 minutes per run gives us 10000 minutes – around 166 hours or a week's worth of continuous runs. If this is too long then one either tries to make the model run faster through compromises and simplifications or one does less experimentation. If one is attempting to optimize, especially with problems involving multiple or many objectives, the utility of these techniques can be limited significantly by this runtime.

The concept of speeding up, or extending, simulation experimentation by using multiple computing resources has been widely explored and sometimes termed distributed simulation (Taylor 2019). For example, by using 10 computers and sharing the runs between them we can potentially bring the runtime down to 16 hours – with 20 computers we can bring this down to 8 hours. In reality, due to complexities of scheduling and sending simulations across a communications network and sharing computations with other work on a computer, the speed up is never entirely linear. However, in general, one can make the claim of "many hands make light work!" There are many approaches to using the fixed computing resources of one or more enterprises (e.g. Computing Clusters, Desktop computer, etc.) to speed up an application. These can be complex and need local investment to set up the resource infrastructure.

A potentially attractive alternative is Cloud Computing. Instead of providing a local infrastructure, users "hire" computing resources from a Cloud to support the computing needs of their application. Cloud has other advantages such as remote access, quality of service, up-to-date technology, no need to implement costly enterprise computing infrastructure, etc. In terms of simulation these are also attractive, especially the idea of hiring many computing resources to speed up experimentation. Cloud-based simulation experimentation application can be complex and, for end user or commercial application development is not just "putting simulation on a cloud". For around seven years we have investigated how advances in Cloud Computing could be used to speed up simulation experimentation through several multinational projects. Initially, this work combined the multi-cloud Platform-as-a-Service middleware CloudBroker with the science gateway and workflow system WS-PGRADE to produce the CloudSME Simulation Platform (Taylor et al. 2018). This has been used to develop several commercial high performance simulation experimentation applications. This approach allows developers to choose from several deployment options and enables them to quickly implement their simulation solution in a way that allows different clouds to be chosen depending on price/performance. Users choose a cloud and the number of resources needed to execute their simulation experiments. However, experience showed that not all cloud resources might be used in experimentation. To automatically scale the cloud resources needed for an application we developed MiCADO and adapted it for simulation experimentation (Kiss et al. 2019a; Kiss et al. 2019b). This has produced more efficient cloud resource usage and is now being adopted for use to "cloud-enable" Saker Solution's SakerGrid, an enterprise desktop grid that supports high speed simulation experimentation (Kite at al. 2011). To bring these together to contribute to the state-of-the-art of cloud-based simulation experimentation, this paper describes each of these advances in turn. Section 2 gives a brief review of related work. Section 3 presents the CloudSME Simulation Platform and its approach to cloud-based application development. Section 4 describes MiCADO and its approach to autoscaling. Section 5 discusses how MiCADO was integrated with SakerGrid. Finally, Section 5 discusses the impact these innovations have made and concludes the paper.

2 RELATED WORK

In terms of developing cloud-based environments to support simulation experimentation there has been other work to that described in this paper. For example, the distributed agent-based traffic simulator Megaffic uses adaptive resource provisioning to speed up the execution of a single simulation run (Hanai et al. 2015). The Scalable Electro-Mobility Simulation Cloud Service (SEMSim CS) is the cloud-based version of the Scalable Electro-Mobility Simulation (SEMSim) platform used to study the impact of large-scale electromobility on a city's infrastructure (i.e. the replacement of the majority of vehicles with electric ones) (Zehe et al. 2015). When an experiment is run, the two simulations are linked together as a distributed simulation that synchronizes simulation time advance between the discrete simulations. The D-Mason framework is a parallel version of the Mason library for writing and running distributed agent-based simulations and has been ported to cloud (Carillo et al. 2016). GridSpice (Anderson et al. 2014) is a cloud-based simulation platform for distributed smart power grid simulation that supports the development of models consisting of a transmission network, distribution networks and power generators. HPCCloud was developed as a cloud/web-based simulation environment platform to deliver a Software-as-a-Service that allows a simulation to be submitted to a cluster in a cloud and submits work to these via distributed task queues (OLeary et al. 2015).

There is less work associated with autoscaling of simulation experiments. Cai et al. (2017) demonstrated that with variable deadline-based workflow applications using cloud resources it can be difficult to meet the deadline constraint. Thai et al. (2018) noted that in a survey of requirements for large scale computing applications that include simulation experimentation (parameter sweeps) that the ability to set deadlines and minimize the cost of cloud resources is a popular requirement. Earlier work showed the feasibility of deadline-based autoscaling but with limited implementations (Mao et al. 2010; Vecchiola et al. 2012).

We now present three examples of innovative cloud-based simulation experimentation approaches.

3 CLOUD-BASED SIMULATION EXPERIMENTATION: THE CLOUDSME PLATFORM APPROACH

The architecture of the CloudSME Simulation Platform (CSSP) is shown in Figure 1. The goal of developing the platform was to support relatively quick deployment of cloud-based simulation applications and applications supporting high speed simulation experimentation that could easily switch between clouds and had several development options. The CSSP integrated three components: the WS-PGRADE/gUSE gateway framework (Balasko et al. 2013) and the CloudBroker Platform (Farkas et al. 2014) and the CloudSME AppCenter (cloudsme.eu/appcenter). The major objective of the CSSP is to ease developers' efforts when cloud-enabling existing simulation software and to speed up significantly the "cloudification" process for *commercial* applications. The CSSP consists of three layers:

- Simulation Applications Layer that allows software vendors deploying and presenting simulation products to end-users as Software as a Service (SaaS) in a wide range of scenarios and deployment models.
- Cloud Platform Layer that provides access to multiple heterogeneous cloud resources and supports the creation of complex application workflows - a Platform as a Service (PaaS) to create and execute cloud-based simulations.
- Cloud Resources Layer that represents the Infrastructure as a Service (IaaS) clouds connected to the platform.

The Simulation Applications Layer consists of the CloudSME AppCenter. This is a web-based frontend that enables software products and services to be offered by software vendors and service providers via a single interface. It offers billing functionality that includes price setting, payment integration and tracking of users' spending. Three main deployment models are supported: Directly Deployed Applications (used via the AppCenter), Web-based Applications (used via a Web Portal or Science Gateway) and Desktop Applications (download and install). To support this, the CSSP offers a diverse set of APIs to support developers that link directly to the other components of the platform (via REST APIs, for example).

The Cloud Platform Layer consists of the cloud-based services from the CloudBroker Platform and the science gateway framework WS-PGRADE/gUSE. The CloudBroker Platform is a commercial PaaS that supports the management and execution of software on different cloud provider resources. CloudBroker uses IaaS clouds from resource providers and incorporates adapters both to public and private cloud infrastructures. The platform provides access to a wide range of resources including open source (e.g. OpenStack and OpenNebula) and proprietary (e.g. Amazon and CloudSigma) clouds, and also various HPC resources. Applications are deployed to CloudBroker. CloudBroker supports non-interactive serial and

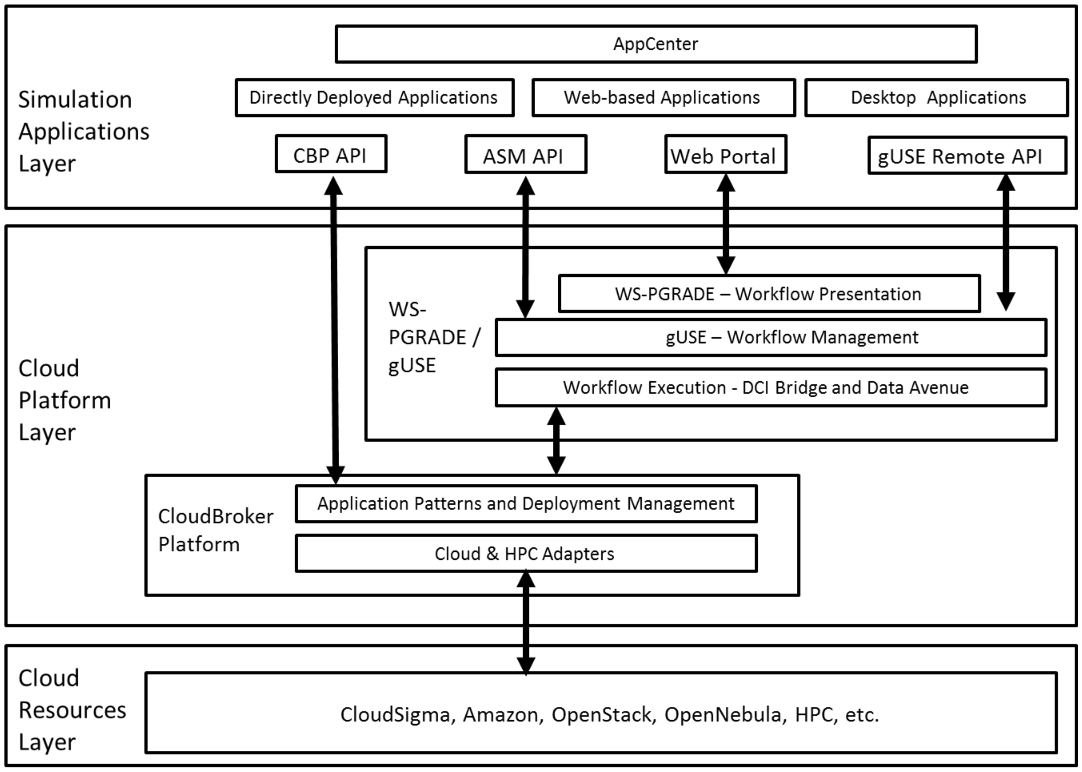
AppCenter
Simulation Applications Layer
Directly Deployed Applications
Web-based Applications
Desktop Applications
CBP API
ASM API
Web Portal
gUSE Remote API
Cloud Platform Layer
WS-PGRADE / gUSE
WS-PGRADE – Workflow Presentation
gUSE – Workflow Management
Workflow Execution - DCI Bridge and Data Avenue
CloudBroker Platform
Application Patterns and Deployment Management
Cloud & HPC Adapters
Cloud Resources Layer
CloudSigma, Amazon, OpenStack, OpenNebula, HPC, etc.

Figure 1: The CloudSME Simulation Platform.

parallel batch processing applications on both Linux and Windows operating systems. The platform itself consists of a set of modules that manage processes, applications, users, finance (accounting, billing and payment), and runtime issues (process monitoring, queuing, resources, storage and images). A scalability and fault handler layer supervises scalability requirements and failure issues. Cloud Provider Access Management oversees the connection to each Cloud technology and can control the number of virtual machines (VMs) started for a given application on a given cloud. gUSE (Grid and Cloud User Support Environment) is an open source scientific gateway framework providing users with easy access to cloud and grid infrastructures. gUSE provides with WS-PGRADE, a Liferay based portal to create and execute scientific workflows in various distributed computing infrastructures (DCIs) including clusters, grids and clouds. This enables the use of workflows on the CSSP. A specific workflow component enables parameter sweeps to be defined (that implement simulation experimentation). In more recent versions of the CSSP a simpler alternative to the workflow environment has been created that supports this function. A full description of WS-PGRADE/gUSE gateway framework is available in Kacsuk et al. (2012).

The Cloud Resources Layer that consists of a range of clouds and HPC resources accessible via the CloudBroker Platform.

Many cloud-based simulation applications have been developed using the CSSP - see the CloudSME website and (Taylor et al. 2018) for examples.

4 CLOUD AUTOSCALING FOR SIMULATION EXPERIMENTATION: THE MICADO APPROACH

The previous section describe an architecture that supports the rapid deployment of commercial cloud-based simulation applications. These have the facilities that allow vendors or users to specify which cloud to use and how many of which cloud resource/instance type. This fixed use of cloud resources is effective

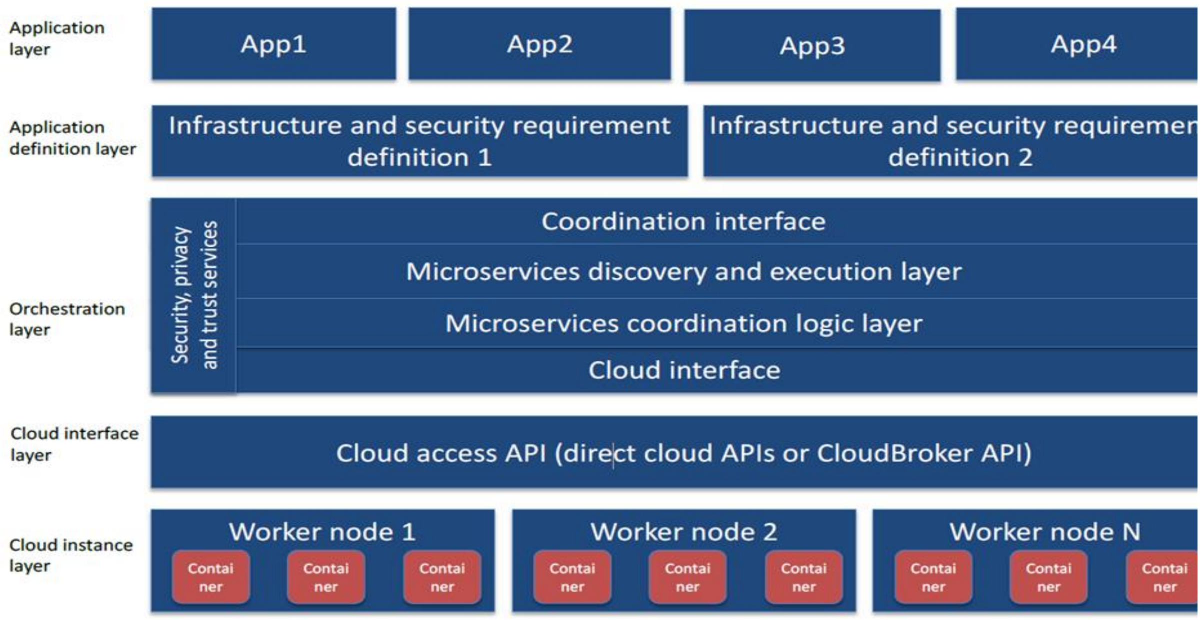
Application layer
App1
App2
App3
App4
Application definition layer
Infrastructure and security requirement definition 1
Infrastructure and security requiremen definition 2
Orchestration layer
Security, privacy and trust services
Coordination interface
Microservices discovery and execution layer
Microservices coordination logic layer
Cloud interface
Cloud interface layer
Cloud access API (direct cloud APIs or CloudBroker API)
Cloud instance layer
Worker node 1
Worker node 2
Worker node N
Contai ner

Figure 2: MiCADO Architecture.

but, as experience has shown in simulation experimentation, not all cloud resources will be used all the time (due to overestimating the amount of work, queuing and scheduling of jobs on and off the cloud, etc.) to see if we could develop a better approach we investigated how autoscaling approaches could be applied to cloud.

Typically, autoscaling of cloud resources is based on some performance-related metrics such as CPU, memory and/or bandwidth usage. However, in simulation experimentation a more appropriate metric is time. This would enable a deadline to be set and cloud resources scaled up/down as experiments are carried out in an attempt to meet the deadline (with appropriate restrictions on cost). The Microservice-based Cloud Application-level Dynamic Orchestrator (MiCADO) framework was developed to support autoscaling and the optimal and secure deployment and runtime orchestration of cloud applications (MiCADO website) (Kiss et al. 2019a). MiCADO has been extended to support deadline-based simulation experimentation (Anagnostou et al. 2019; Kiss et al. 2019b). An outline of MiCADO is given below.

As shown in Figure 2, MiCADO's architecture is organized in layers using microservices. The layers are:

- Application layer: the application (simulation) code with an application description template that defines the application requirements in terms of infrastructure, security and connectivity.
- Orchestration layer: this has a Coordination interface API and a Cloud interface API and is responsible for overall scaling.
- Microservices discovery and execution sublayer: starting and stopping microservices execution and keeping reachability information at runtime such as IP address and port number.
- Microservices coordination logic sublayer: operations of running infrastructure such as starting and stopping instances and microservices migration.
- Security, privacy and trust services vertical layer: security management across the Orchestration layer - security services are defined in the Application layer.

- Cloud interface layer: support access to cloud resources (different cloud middleware APIs for direct access to cloud resources/interface with Platform as a Service (PaaS) APIs so as additional services, such as billing and account management, can be provided.
- Cloud instance layer: actual cloud instances provided by Infrastructure as a Service (IaaS) providers and can be private or public infrastructures.

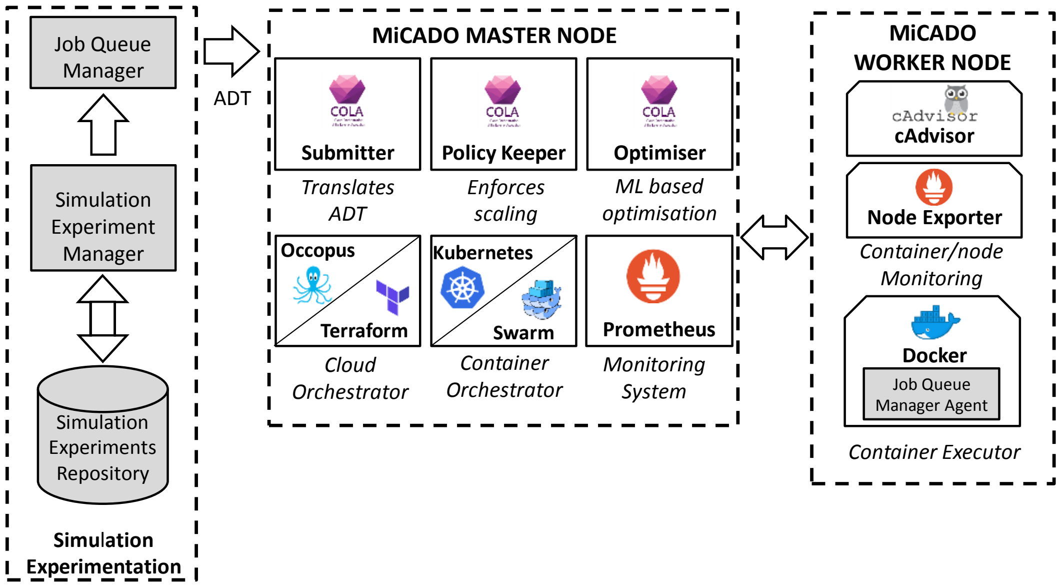

Figure 3: MiCADO for Simulation Experimentation

Figure 3 shows the version of MiCADO developed for autoscaling cloud-based simulation experimentation. All components are implemented using open source tools as indicated. The figure shows the MiCADO core and the deadline-based simulation experimentation application specific components. Functions denoted with clear boxes belong to MiCADO core, while the ones in grey boxes are extensions required to support simulation experimentation.

MiCADO consists of a MiCADO Master node and MiCADO Worker nodes. Inside the MiCADO Master node the Submitter receives the Application Description Template (ADT, e.g. a TOSCA specification (TOSCA 2016)) that includes a description of the application, the container and virtual machine topology, and the policy description (in case of simulation experimentation the parameters related to the deadline by which the experiment needs to be completed). The Cloud Orchestrator allocates and releases virtual machines by communicating with the cloud APIs and starting/stopping Worker nodes. The Container Orchestrator scales at the container level and allocates applications to containers in the Worker nodes, track their execution and destroys them when needed. The Monitoring System collects information on resources utilization from the Worker nodes and creates alerts if appropriate. The Policy Keeper implements the application policies and makes decisions to start/stop cloud resources and to schedule containers on Worker nodes. It also keeps the Cloud and Container Orchestrators synchronized. The Optimizer always runs in the background and performs optimization calculations on demand; it informs Policy Keeper on the optimized setup of cloud resources and container infrastructure. MiCADO Worker Nodes contain the Container/node monitor that is responsible for measuring the load of the resources and the resource usage of the container services. The measured attributes are then given to the Monitoring System running on the Master Node. The Container Executor starts, executes and destroys containers upon request from the Container Orchestrator. Container components realise the user services defined in the (container) infrastructure description submitted through the MiCADO Submitter on the Master Node. In the case of simulation experimentation there is also a specific component running in the containers, the Job Queue Manager Agent that communicates with the external Simulation Experimentation services (left hand side of Figure 3).

The components supporting simulation experimentation are external to MiCADO. The Job Queue Manager is responsible for communicating with MiCADO and generating and passing on the Application Description Template (ADT). This description includes the simulation application to be executed and the necessary parameters to enforce deadline-based scaling (e.g. the deadline, the estimated time for one simulation run, URLs for simulation executables, the maximum number of virtual machines/containers that can be deployed, etc.) The Simulation Experiment Manager, that typically also includes the user interface, is responsible for uploading experiment details (e.g. simulation model, input data, etc.) to the Simulation Experiments Repository, downloading the results from the repository and sending the experiment description (provided by the user for example in JSON format) to the Job Queue Manager. The Simulation Experiments Repository keeps all experiments data and files (or URLs to data and files).

MiCADO is being adopted by several scientific user communities and commercial companies, for example by Saker Solutions, in the development of a cloud-based extension to their simulation experiment SakerGrid. This is presented in the next section.

5 CLOUD-BASED AUTOSCALING IN THE SAKERGRID SIMULATION EXPERIMENTATION ENVIRONMENT

Saker Solutions Ltd is an independent supplier of simulation solutions from the United Kingdom that underpins its simulation offers with the provision of innovative technologies which support users to gain the most from simulation projects. They have produced the SakerGrid Platform, an enterprise desktop grid that significantly reduces simulation experimentation time by using available enterprise computing resources (Kite et al. 2011). The capacity of SakerGrid is restricted by the number of available desktop grid workers at any time and adopting the on-demand computational resources of a cloud, especially combined with efficient use through autoscaling, is very attractive. The following presents how MiCADO has been integrated with SakerGrid.

The new SakerGrid-MiCADO architecture is shown in Figure 4. This integration shows changes made in both architectures, particularly in terms of the Worker nodes and the delivery of several dynamically changing parameters to MiCADO for the deadline-based scaling policy. Changes were needed to associate a Worker Service to a given simulation experiment, i.e. when worker nodes are created MiCADO must ensure sure that the new Worker Service does not process runs belonging to another experiment. The Worker node now takes the (SakerGrid) experiment ID and the Worker Service instructs the Manager Service to send jobs belonging only to the given experiment. This modification was needed as Saker runs multiple experiments from multiple projects.

The SakerGrid Manager Service was also modified to give the necessary information to MiCADO for deadline-based scaling of jobs. This involved extending the Manager Service to make some parts of its internal database visible (see DB in Figure 4). An SQL table is created and maintained by the Manager Service to provide information to MiCADO so that it has information on the number of running and waiting run, the average execution time of a run, the deadline of the experiment, and the current number of idle and busy workers. These are continuously monitored by a Prometheus SQL exporter (see SQL exporter in Figure 4).

At the time of development, the implementation of the MiCADO Master component supported the deployment, execution and scaling of only one application at a time. To support parallel execution of multiple experiments/projects from the SakerGrid platform, a dedicated MiCADO instance is needed to be deployed for the lifetime of every experiment. A new MiCADO launcher component was designed and implemented that instantiates a new MiCADO master whenever the SakerGrid Manager Service requires. When a user creates a new experiment through the SakerGrid Client Service, the SakerGrid Manager invokes a REST call from MiCADO Launcher component interface and passes the experiment ID to be

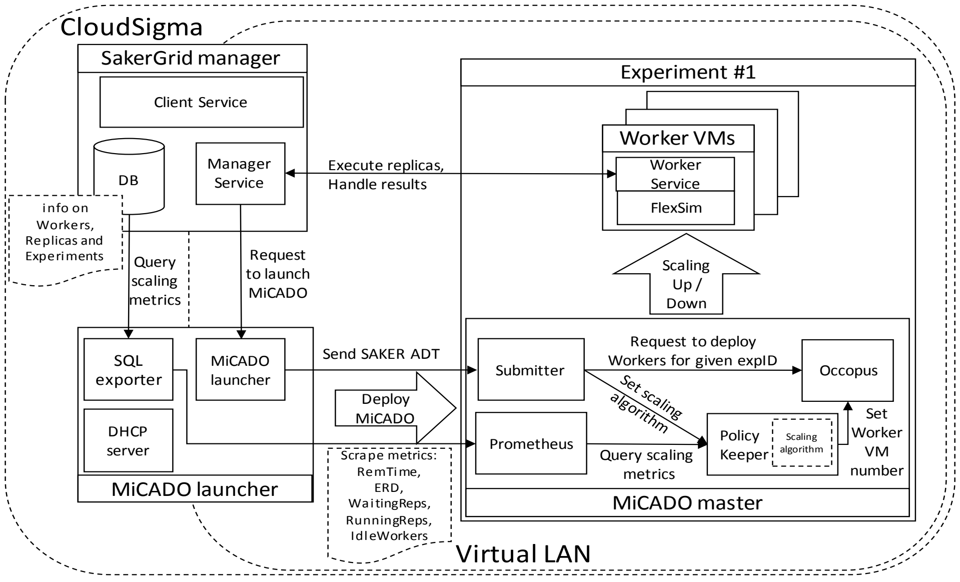

Figure 4: SakerGrid-MiCADO Architecture

delivered to the Worker nodes. The Launcher then instantiates the MiCADO master, generates the TOSCA-based application template by inserting the experiment ID and submits this to the newly created MiCADO Master instance. The TOSCA template contains blocks to describe the components to be deployed and the related scaling policy. The template is finalised by the Launcher component by adding experiment ID for the Worker nodes. The ID is also used in monitoring to ensure the correct experiment-related information is used.

Once the MiCADO Master for an experiment has been created, and the launcher has generated and submitted the TOSCA template, the deployment is performed automatically and the scaling activity is started by the MiCADO Master. The Submitter generates the necessary descriptors for Policy Keeper and Occopus. The Occopus descriptor contains the experiment ID for the Worker VMs in order to specify the associated experiment. The Policy Keeper descriptor includes the scaling policy and the specific Prometheus queries for the current experiment. With these descriptors, the Submitter initiates the Worker VM creation through Occopus with the initial number of VMs. Next, each newly created Worker VM starts the Worker Service which joins to the Manager Service and executes the simulation application (in this case FlexSim™, to process simulation runs belonging to the experiment. Finally, Prometheus starts monitoring by collecting periodically the necessary parameters and delivering them to the Policy Keeper that makes the necessary decisions on the number of required Worker nodes.

In MiCADO the Policy Keeper component executes executing the scaling algorithm periodically which returns the number of Worker VMs to be kept for the experiment. The algorithm takes the following metrics as input: remaining time until deadline, estimated running time of one replication, number of runs (jobs) which has not yet started, number of runs being processed and number of idle workers. These variables are periodically queried from Prometheus and used by the scaling algorithm during execution. The scaling policy checks the deadline formula and up/down scales the Worker VMs. When the execution of the experiment finished (i.e. all replications successfully completed), the SakerGrid manager may instruct the MiCADO launcher to shut down the MiCADO Master belonging to the given experiment to release all resources associated with the experiment. Otherwise it is possible to generate new replications in this experiment, override the deadline and start new Worker nodes to process replications again.

6 DISCUSSION

The previous sections have presented three innovations in cloud-based simulation experimentation. Experiences with the CloudSME Simulation Platform have shown that it is feasible to create innovate cloud-based simulation applications. The AppCenter, or at least some kind of monitoring, billing and payment functionality, is needed for commercial applications as there needs to be some way for a user to paying. The workflow functionality can be complex for some developers and given that for most simulation experimentation applications only the parameter sweep function was used this is often replaced with the new cloud-based simulation application having a dedicated job launch. Experiences with MiCADO have led to some innovative non-simulation cloud-based applications that, for example, automatically provision new web services for web hosting/management applications. The porting of MiCADO to Saker Solution's SakerGrid continues and forms part of the on-going development of cloud-based applications and infrastructure at the company. This approach is also being currently integrated with the original CloudSME Simulation Platform.

These innovative approaches has enabled the development of several cloud-based simulation experimentation applications. However, one of our goals is to make Cloud Computing as accessible as possible to simulation users. Anecdotally, from our experience, we can identify four kinds of people in the simulation community that we have discussed this problem with over the years. These were (very informally!) the "super geeks", the "geeks", the "outsiders" and the "potentials". The super geeks are highly technically (computing) aware expert simulation users who could both develop their own cloud-based simulation applications and use them for fast, methodologically correct simulation experimentation (e.g. members of the Parallel and Distributed Simulation community, technical developers working at simulation vendors, etc.). The geeks are technically aware expert simulation users who were attempting to build cloud-based simulation applications but did not have the technical capability to take advantage of contemporary technical approaches. The outsiders are those who did have the technical capability but did not really understand critical issues of simulation methodology. The potentials are those that regularly build models and correctly perform simulation experimentation and either always or sometimes see experimentation time as a major issue in their work. It has been rare to meet simulation users that never see experimentation time as a challenge.

Our "potentials", typically have considerable simulation expertise and (quite rightly) little technical (computing) knowledge and use the "computer" as a tool. They have no desire or motivation to become geeks and cloud-based solutions need to account for this. Some simulation vendors have developed cloud running functionality in their software but this tends to be hidden and cloud costs appears to be covered (and limited) by annual license agreements or specifically negotiated contracts. Cloud requires payment and, despite individual cloud resources being extremely cheap, continued cloud use can become expense (especially data transfer in and out of the cloud). Arguably users also need to be exposed to choice – the cloud market is no longer just supported by the likes of Amazon, Microsoft, etc. A new and competitive cloud market is developing and cost/quality of service varies across these clouds (what might be the best deal today from one cloud provider might be bettered tomorrow by another). This gives us two perspectives: we want the solution to be as simple as possible and we want to give users choice.

To address these we created a web-based portal that enables users to run simulation experiments on cloud with two open source simulation software: JaamSim (JaamSim Development Team 2016) (discrete-event simulation) and REPAST (North et al. 2013) (agent-based simulation). These were selected so that we had one exemplar from each simulation community, they can be run on linux cloud instances (Microsoft Windows is possible but at extra cost as it is not an open source operating system), and (bring open source) there is no license cost for running multiple simulation in parallel. The portal uses the CloudBroker element of the CloudSME Simulation Platform for billing and payment. In Cloud Computing there is a need for some online fileserver that holds the models, software, experiment parameters that a cloud-based simulation experimentation application would access to send each experiment run/job (consisting of the model, the simulation software and the parameters) to a cloud instance. The results from each run would also be sent back to the fileserver. Many of our "potentials" do not know how to setup a web-based fileserver or work at a company or University that can set one up without financial justification (essentially another barrier to cloud use). We decided that we would use the Dropbox API as a fileserver – users store their files on a Dropbox account and our system accesses it as it if was a fileserver (other web-based file stores are available). To use our new CloudSME Portal, users add credit to their CloudBroker account, logon onto the Portal (the CloudBroker account is linked), add their Dropbox credentials (to allow the Portal to access their Dropbox account), upload their JaamSim or REPAST models/parameters to a specified Dropbox directory, choose a cloud/cloud instance type and number of instances (costs are shown – currently we help users to understand the cost/performance implications), and press "run". Users monitor their Dropbox account to see results returning and to see when their experiment is complete. Our CloudSME Portal is in beta testing and we welcome volunteers to test our system (hosted by CloudSME UG). Overall, performance is similar to performances reported in the testing of the CloudSME Simulation Platform (Taylor et al. 2018). We will integrated autoscaling into our Portal at a later date.

7 CONCLUSIONS

The main problem addressed in this paper is how to overcome limitations of experimentation time in simulation by using one or more cloud-based innovations presented in this paper. The literature cited in the paper gives examples of use and performance of the different systems. Overall, the potential impact of these and other similar cloud-based innovations is widespread as simulation users discover that high speed experimentation is possible and so making time savings in projects or enabling wider exploration of problems. Indeed an interesting area of research is how these extra, but not unlimited, parallel computational resources can introduce new forms of optimization into simulation. Overall, one might observe that these innovations might make a significant impact in the use of simulation for innovation.

ACKNOWLEDGMENTS

This work was part funded by the EU FP7 CloudSME project (Project no. 608886), EU H2020 COLA (Cloud Orchestration at the Level of Application) project (Project no. 731574) and the ASCLEPIOS Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare (Project No. 826093). We would also like to thank Nicola Fantini and his team (Scaletools AG) and Andeas Ocklenburg (CloudSME UG) for their continued support in this work. We would also like to note that there are many other authors that have been involved in this work and they have been noted in the cited papers.

REFERENCES

Anagnostou, A, S. J. E. Taylor, N. T. Abubakar, T. Kiss, J. DesLauriers, G. Gesmier, G. Terstyanszky, P. Kacsuk, and J. Kovacs. 2019. "Towards a Deadline-based Simulation Experimentation Framework using Micro-services Auto-scaling Approach." In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 2749-2758. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Anderson, K., J. Du, A. Narayan, and A. E. Gamal. 2014. "GridSpice: A Distributed Simulation Platform for the Smart Grid". *IEEE Transactions on Industrial Informatics* 10(4):2354-2363.

Balasko, A., Z., Farkas, and P. Kacsuk. 2013. "Building Science Gateways by Utilizing the Generic WS-PGRADE/gUSE Workflow System". *Computer Science, Open Journal Systems* 14(2):307–325.

Cai, Z., Q. Li, and X. Li. 2017. "ElasticSim: A Toolkit for Simulating Workflows with Cloud Resource Runtime Auto-Scaling and Stochastic Task Execution Times". *Journal of Grid Computing* 15(2):257-272.

Carillo, M., G. Cordasco, F. Serrapica, C. Spagnuolo, P. Szufel, and L. Vicidomini. 2016. "D-MASON on the Cloud: An Experience with Amazon Web Services". In *Proceedings of the Euro-Par 2016: Parallel Processing Workshops*, August 22nd-26th, Grenoble, France, 322-333.

Farkas Z., A. Hajnal, and P. Kacsuk. 2014. "WS-PGRADE/gUSE and Clouds". In *Science Gateways for Distributed Computing Infrastructures*, edited by P. Kacsuk, 97-110. Cham: Springer.

Hanai, M., T. Suzumura, A. Ventresque, and K. Shudo. 2015. "An Adaptive VM Provisioning Method for Large-scale Agent-based Traffic Simulations on the Cloud". In *Proceedings of the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, December 15th-18th, Singapore, 130-137.

JaamSim Development Team. 2016. JaamSim: Discrete-Event Simulation Software. Version 2016-14. http://jaamsim.com, accessed 15th July.

Kacsuk, P., Z. Farkas, M. Kozlovszky, G. Hermann, A. Balasko, K. Karoczkai, and I. Marton. 2012. "WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities". *Journal of Grid Computing* 10:601–630.

Kiss, T., P. Kacsuk, J. Kovacs, B. Rakoczi, A. Hajnal, A. Farkas, G. Gesmier, and G. Terstyanszky. 2019a. "MiCADO - Microservices-based Cloud Application-level Dynamic Orchestrator". *Future Generation Computer Systems* 94:937-946.

Kiss, T., J. DesLauriers, G. Gesmier, G. Terstyanszky, G. Pierantoni, O. Abu Oun, S. J. E. Taylor, A. Anagnostou, and J. Kovacs. 2019b. "A Cloud-agnostic Queuing System to Support the Implementation of Deadline-based Application Execution Policies". *Future Generation Computer Systems* 101:99-111.

Kite, S., C. Wood, S. J. E. Taylor, and N. Mustafee. 2011. "Sakergrid: Simulation Experimentation Using Grid Enabled Simulation Software", In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 2283-2293. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mao, M., J. Li, and M. Humphrey. 2010. "Cloud Auto-Scaling with Deadline and Budget Constraints". In *Proceedings of the 11th IEEE/ACM International Conference on Grid Computing*, October 25th-28th, Brussels, Belgium, 41-48.

North, M.J., N.T. Collier, J. Ozik, E. Tatara, M. Altaweel, C.M. Macal, M. Bragen, and P. Sydelko. 2013. "Complex Adaptive Systems Modeling with Repast Simphony". *Complex Adaptive Systems Modeling* 1(3).

OLeary, P., M. Christon, S. Jourdain, C. Harris, M. Berndt, and A. Bauer. 2015. "HPCCloud: A Cloud/Web-Based Simulation Environment". In *Proceedings of the 2015 IEEE 7th International Conference on Cloud Computing Technology and Science*, November 30th-December 3rd, Vancouver, Canada, 25-33.

Taylor, S. J. E. 2019. "Distributed Simulation: State-of-the-Art and Potential for Operational Research". *European Journal of Operational Research* 273(1):1-19.

Taylor, S. J. E., A. Anagnostou, T. Kiss, G. Terstyanszky, H. Visti, Z. Farkas, P. Kacsuk, A. Sereda, and N. Fantini. 2018. "The CloudSME Simulation Platform and its Applications: A Generic Multi-cloud Platform for Developing and Executing Commercial Cloud-based Simulations". *Future Generation Computer Systems* 88:524-539.

Thai, L., B. Varghese, and A. Barker. 2018. "A Survey and Taxonomy of Resource Optimisation for Executing Bag-of-Task Applications on Public Clouds". *Future Generation Computer Systems* 82:1-11.

TOSCA. 2016. TOSCA Simple Profile in YAML Version 1.0. http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csd03/TOSCA-Simple-Profile-YAML-v1.0-csd03.html, accessed 29th April.

Vecchiola, C., R. N. Calheiros, D. Karunamoorthy, and R. Buyya. 2012. "Deadline-driven Provisioning of Resources for Scientific Applications in Hybrid Clouds with Aneka". *Future Generation Computer Systems* 28(1):58-65.

Zehe, D., A. Knoll, W. Cai, and H. Aydt. 2015. "SEMSim Cloud Service: Large-scale Urban Systems Simulation in the Cloud". *Simulation Modelling Practice and Theory* 58(2):157-171.

AUTHOR BIOGRAPHIES

SIMON J. E. TAYLOR is a Professor and the Director of the Modelling and Simulation Research Group in the Department of Computer Science, Brunel University London. His email address is simon.taylor@brunel.ac.uk and his ORCID is orcid.org/0000-0001-8252-0189.

ANASTASIA ANAGNOSTOU is a Lecturer and the Deputy Director of the Modelling and Simulation Research Group in the Department of Computer Science, Brunel University London. Her email address is anastasia.anagnostou@brunel.ac.uk and her ORCID is orcid.org/0000-0003-3397-8307.

NURA TIJJANI ABUBAKAR is a PhD researcher in the Modelling and Simulation Research Group in the Department of Computer Science, Brunel University London. His email address is nura.abubakar@brunel.ac.uk.

TAMAS KISS is a Professor in Distributed Computing at the Department of Computer Science and the Director of the University Research Centre for Parallel Computing at the University of Westminster. His email address is t.kiss@westminster.ac.uk.

JAMES DESLAURIERS is a Research Associate at the University of Westminster. His email address is J.Deslauriers@westminster.ac.uk.

GABOR TERSTYANSZKY is a Professor in Distributed Computing at the University of Westminster. His email address is G.Z.Terstyanszky@westminster.ac.uk.

PETER KACSUK is the Director of the Laboratory of the Parallel and Distributed Systems in the Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA SZTAKI). His email address is Kacsuk.Peter@sztaki.mta.hu.

JOZSEF KOVACS is a Senior Research Fellow at the Laboratory of Parallel and Distributed Systems (LPDS) at the Institute for Computer Science and Control (SZTAKI) of the Hungarian Academy of Sciences (MTA). His email address is jozsef.kovacs@sztaki.mta.hu.

SHANE KITE is the Managing Director of Saker Solutions, a company he founded in 2003. His email address is shane.kite@sakersolutions.com.

GARY PATTISON is a lead consultant at Saker Solutions with more than 30 years' experience in the application of simulation. His email address is gary.pattison@sakersolutions.com.

JAMES PETRY a lead applications developer at Saker Solutions. His email address is james.petry@sakersolutions.com.