Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

A TUTORIAL ON CLOUD COMPUTING FOR AGENT-BASED MODELING & SIMULATION WITH REPAST

Simon J. E. Taylor
Anastasia Anagnostou

Department of Computer Science
Brunel University London
Kingston Lane
Uxbridge, UB8 3PH, UNITED KINGDOM

Tamas Kiss
Gabor Terstyanszky

Centre for Parallel Computing
University of Westminster
115 New Cavendish Street
London, W1W 6UW, UNITED KINGDOM

Peter Kacsuk

MTA SZTAKI
Victor Hugo utca 18-22
Budapest, 1132, HUNGARY

Nicola Fantini

ScaleTools Schweiz AG (Headquarter)
Huobstrasse 10
Pfäffikon, Zurich, 8808, SWITZERLAND

ABSTRACT

Cloud computing facilitates access to elastic high performance computing without the associated high cost. Agent-based Modeling & Simulation (ABMS) is being used across many scientific disciplines to study complex adaptive systems. Repast Simphony (Recursive Porous Agent Simulation Toolkit) is a widely used ABMS system. Cloud computing can speed-up significantly ABMS to facilitate more accurate and faster results, timely experimentation, and optimization. However, the many different Clouds, Cloud middleware and Service approaches make the development of Cloud-based ABMS highly complex. This tutorial introduces the CloudSME Simulation Platform (CSSP) that enables simulation software to be deployed as service (SaaS) supported by a cloud platform (PaaS). It shows how Repast can be deployed as a cloud computing service as part of a workflow of tasks. A case study demonstrates how the CSSP can easily run agent-based simulations written in Repast on multiple Clouds.

1 INTRODUCTION

Agent-based Modeling & Simulation (ABMS) is being used in many scientific disciplines to study complex adaptive systems. As with other types of simulation, the need for many runs may restrict the amount of experimentation and investigation due to limited time constraints. Cloud computing offers an attractive solution to this problem by providing elastic computing resources. However, cloud computing can be very complex to use. This paper introduces how cloud computing could be used to support the computing needs of ABMS. The paper first introduces cloud computing, ABMS and the repast simphony ABMS system. It then presents the CloudSME Simulation Platform, a suite of technologies dedicated to development Modeling & Simulation as a Service approaches to cloud computing. A case study is then presented showing how a repast ABMS can be easily run over three different clouds (Amazon EC2 and two University clouds).

2 CLOUD COMPUTING

The concept of using computing power as a utility can be traced back to the 1960s (Hill et al. 2013). "Cloud computing" as a term appeared in the mid-2000s and is used to describe the access of internet-based computing resources. The National Institute of Standards and Technology (NIST) has made efforts to standardize the terminology of Cloud computing. The following section draws on Mell and Grance (2011) and the recently published NIST Cloud Computing Standards Roadmap (NIST 2013) to define Cloud computing, its deployment and service models. Cloud computing is "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". *Elasticity* is the main concept of Cloud computing. That is, the use of computational resources, storage, applications, etc. that can be instantly increased according to user needs, and ceases when the user does not need these services.

2.1 Cloud Deployment Models

Cloud services can be deployed according to the organizational structure and the security requirements of the users. Currently, there are four main cloud deployment models (see Figure 1) for cloud resource provision.

Private clouds can be accessed only by a single organization. The provided services are accessed via private networks and the cloud consumers are members of the organization. In this model there is a major consideration on security.

Community clouds can be considered as an extension of a private cloud. A community cloud can be accessed by more than one organization with common interests. The infrastructure and datacenters are located on or off site and can belong to one or more of the organizations in the community or a third party. An example of a community cloud is a government cloud where the different governmental organizations can access the provided services.

Public clouds services can be accessed by the general public. These are available by public networks, most commonly via the internet. The infrastructure and datacenters are located on the premises of the cloud provider. An example of a public cloud is the Amazon Elastic Compute Cloud (EC2). Also, public cloud services are commonly provisioned by academic institutions.

Hybrid clouds are a combination of two or all the three of the above. This model allows service portability by using standardized methods. An issue in hybrid cloud deployment could be the lack of standards for communicating data and services among different cloud providers.

2.2 The Cloud Computing Stack

The cloud computing stack reflects the cloud computing service models. There are three defined main service models, as shown in Figure 2. From the bottom-up after the physical hardware and networking layers, these are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Cloud consumers (users) typically can access all three services via a web interface or some language supported API.

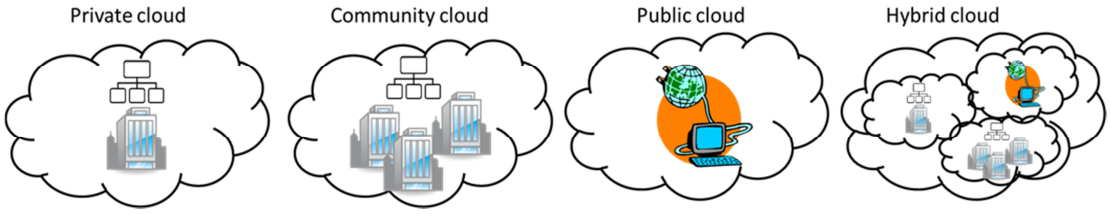

Figure 1: Cloud deployment models.

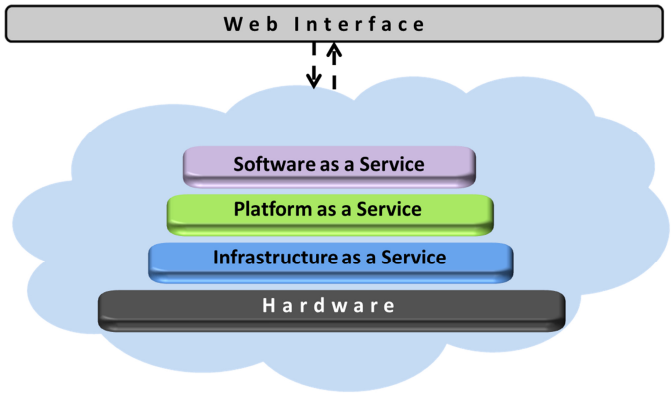

Figure 2: The Cloud computing stack.

IaaS stands at the lowest level of the main provided service models. IaaS does not give access to the physical layer, however allows access to a virtual infrastructure where the user can deploy operating systems and application software. Also, an IaaS cloud consumer can deploy network features, such as security settings and firewalls. PaaS stands on the top of the IaaS and allows for software deployment and configuration on the existing platform. The PaaS cloud consumer does not control or manage the operating systems and network features. The users of this service can deploy existing software or develop software with the provisioned libraries and compilers by the cloud provider. The most commonly used service model is the SaaS, which sits atop the PaaS. A SaaS cloud consumer has only application level access. Users can use the provided applications deployed as online services with no knowledge of the platform or the infrastructure. The applications can be accessed by web browsers or program interfaces.

As a service, Modeling & Simulation (M&S) software have particular characteristics that support simulation. For example, a simulation experiment usually runs for a number of different input parameter sets and each parameter set executes for a number of replications. Stochastic simulations introduce randomness that has to be controlled for each replication. In order to compare the outputs for each input parameters set, the analyst must be able to repeat the same random number generation for each run. Furthermore, when the replications of a simulation run are distributed, the risk of using the same random stream in every node must be considered.

This, and other issues such as reuse and interoperability, strengthen the view that specific cloud services need to be developed to support M&S. Simulation studies require extensive computer resources and usually are conducted infrequently. Also, building simulation models requires technical expertise and specialized software. Modeling and Simulation as a Service (MSaaS) would allow users standardized access to existing simulations, build their own simulation models by accessing specialized software, and run simulation experiments. Also, users could access services that simplify the simulation model building process and enhance the simulation experiments, e.g., industry-specific templates, data modeling tools, simulation optimizers, etc.

For these reasons, cloud-based M&S software is perceived as a separate cloud service model and it has been suggested that MSaaS needs further specification and refinement; this would sit on top of the Cloud computing stack and make use of SaaS, etc. MSaaS consumers would then be able to access sophisticated cloud-based MSaaS via a web browser or API.

On this basis, this paper now discusses how MSaaS might be realized with respect to the development of Cloud computing for Agent-Based Modeling and Simulation.

3 AGENT-BASED MODELING AND SIMULATION

Agent-Based Modeling and Simulation (ABMS) historically originated from Complex Adaptive Systems (CAS), where the principal area of study is the complex behaviors among individual and autonomous agents. CAS have the ability to self-organize and dynamically restructure their components in order to adapt and respond to their environment. Their first widespread use was in the investigation into adaptation and emergence of biological systems (Macal and North 2006). CAS can be characterized by properties and mechanisms that demonstrate a valuable framework for ABMS design. As identified by Holland (1995), the properties and mechanisms of CAS are:

- *Properties*: **Aggregation** that allows groups to form, i.e., individuals can be classified into general categories; **Nonlinearity** that invalidates simple extrapolation, i.e., simple changes can cause large effects that are not easily predictable; **Flows** that allow the transformation and transfer of information and resources between the nodes of a network. Two main concepts that describe the flows in CAS are the multiplier effect and the recycling effect. The multiplier effect denotes the changes in the system when a node is added, and the recycling effect denotes the changes in the system when information and resources are reused; **Diversity** that allows agents to behave differently from one another.
- *Mechanisms*: **Tagging** that allows agents to be named and identified. Tagging may refer to a simple name or ID of an item of the aggregated group or it may refer to more complex behaviors that characterize an item; **Internal models** that allow agents to reason about their micro-worlds, for example, an agent is able to anticipate the outcome of an input if this input reoccurs; **Building blocks** that allow components and whole systems to be composed of simpler components. For example, a bicycle can be a combination of a frame, wheels, etc. These components can have different characteristics, i.e., color, size, etc. and can be reused and recombined as building blocks to compose different bicycles.

ABMS is used mainly to model decentralized, complex systems that consist of many inter-dependencies. Compared with other modeling techniques, ABMS provides a useful alternative when a major element of a system consists of interacting entities. The main components of ABMS are the agents. Agents are autonomous components that have a sort of intelligence in a way that they can recognize their environment and other agents and interact with them. Also, they can be heterogeneous, adaptive and have goal-directed components. That is, the agent characteristics and behaviors may vary, agents may learn from their environment and change their behavior accordingly, and they may have a goal to reach and therefore compare their status with their goals and adjust accordingly. Agents can contain a basic level set of rules that determine their behavior and a higher lever set of rules that can change these rules (Loefstrand et al. 2003, and Macal and North 2010).

Agents have attributes and methods (Hawe et al. 2012). According to Macal and North (2010), agents' attributes can be static, that is they do not change, i.e., name, ID, etc., or dynamic, that change during the simulation run. Dynamic attributes can be the agent's memory that holds instances of past events, the resources that the agents may have (i.e., energy), the knowledge of their neighbors, etc. The methods of an agent are, among others, its behaviors, its ability to modify these behaviors, and the ability to update its rules and its dynamic attributes. Agents have four essential characteristics:

- Agents are distinguished, independent individuals with rules that administer their behavior and decision-making capability. Their nature is discrete, which means that they have clear boundaries and it can be easily determined whether a characteristic belongs to a specific agent or is shared among agents.
- Agents are active components of an environment and coexist with other agents, and, therefore, can be characterized as social components. Usually, communication protocols enable agents to

interact with one another and their environment. Agents can recognize the behavior of other agents.

- Agents are autonomous and self-directed. They have their own set of behavioral rules that dictate their decisions and actions. The degree of sophistication of these behavioral rules indicates the intelligence of the agent which is decided according to the scope of the model.
- Agents have a state that varies over the simulation time. The state of an agent is dictated by its state variables and can be a set or a subset of its attributes.

As mentioned earlier, agents can be heterogeneous entities that are characterized by their behavioral rules. The level of the behavioral rules sophistication depends on the agents' cognition, the agents' internal model of the external environment and other agents, the extent of memory of past events that agents use as experience for decision making.

The way that agents are connected to each other constitutes the topology of the ABMS model. Agents usually do not communicate with all the other agents in the space. A common concept of agents is the neighborhood. Each agent can hold information about its local neighborhood and the neighboring agents and communicate with them. The agents can move in a number of different topologies. A very common spatial topology for agents is a form of Cellular Automata (CA). Agents move on a grid and their neighborhood consists of the adjacent grid cells. A common grid neighborhood is the von Neumann neighborhood that consists of five cells, i.e., the central cell and the four adjacent cells that represent right angle directionality. Another commonly used neighborhood is the Moore neighborhood that is formed of nine grid cells and includes the 45° angle directionality. In an Euclidean topology, agents can travel in two- or three-dimensional space. The radius of the agents' neighborhoods is decided by the modeler. Another widely used topology is the network topology. The nodes of the network are the interacting agents and the links indicate the communication between the nodes. When the links are predefined, the network is called static. However, a network topology can be dynamic, too. In a dynamic network, the communication links are changed during the course of the simulation. Sometimes the nodes of the dynamic network are changed, as well. Another popular topology for an ABMS is the Geographic Information System (GIS) where agents move on a realistic geospatial environment. GIS deployment gives a more realistic view of the model. Finally, agents can have no locality. This type of topology is called an aspatial or "soup" model. It is possible for an agent model to have several topologies. Examples of ABMS can be found in Taylor (2014).

Repast simphony (Recursive Porous Agent Simulation Toolkit) is a free and open-source ABMS environment developed by The University of Chicago and the Decision and Information Sciences Division at Argonne National Laboratory (www.dis.anl.gov). It is widely used for modeling complex and dynamically adaptive behaviors of a system (North et al. 2013). We now give a brief overview of a repast agent-based simulation.

The initialization of a repast ABMS is programmed in a class that uses the ContextBuilder<T> interface. In this class, the environment, the initial number of agents that are located in the environment, etc. are set up. Each agent type's logic is programmed in the respective agent class. Each agent can interact with other agents and the environment using their methods, as portrayed in Figure 3. The number

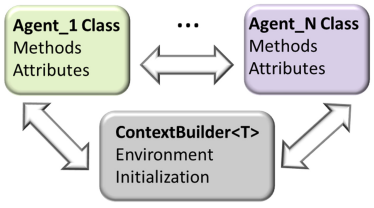

Figure 3: Repast agents and environment.

of objects of an agent class is the population of the specific agent type that exist in the context at a specific time instance. The agents have all the fundamental concepts of object-oriented programming, i.e., inheritance, encapsulation, polymorphism.

Repast models run using the repast simphony runtime GUI. This allows a modeler to configure the model runtime requirements, the visualization requirements, the graphical presentation of the outputs, etc. Furthermore, if specified in the runtime GUI, the user can experiment with different input parameter values. For example, as shown in Figure 4, the listed parameters can be changed from the GUI before initializing a run. The model parameters and their initial values are described in an XML file.

Repast libraries offer a batch run main() method. When running using the batch main(), repast models can execute in parallel runs and perform parameter sweep executions. Batch runs are called either from the GUI or from a command line. This method is used in the presented implementation, however, we do not use the parallel runs but rather the parameter sweep functionality. ABMS models are usually stochastic simulations. For example, an agent can replicate itself at time intervals that are sampled from a probability distribution. To achieve the acceptable level of confidence that the results are accurate and not represent only a random case, many replications of the simulation needs to execute in a run and the output values are the means of these replications. The standard deviation of the values is reported too. The number of replications depends on the degree of the randomness that governs the model, but usually a large number is necessary to reach a steady state.

Most commonly, ABMS is used to model large and complex systems. For example, the number of agents in an infection model in an urban environment reflects the actual population of the city and their interactions. Such a model can take hours to execute on a single machine and consumes a considerable amount of memory. Being able to distribute the computational load across many processors can, among others, decrease the execution time. To understand how Cloud computing can speed-up the execution of an ABMS using different cloud deployment models and resources, the next section introduces the CloudSME Simulation Platform.

4 CLOUDSME SIMULATION PLATFORM

The CloudSME Simulation Platform has been developed by the European Cloud-based Simulation platform for Manufacturing and Engineering (CloudSME) project (www.cloudsme.eu) that aims to develop a cloud-based simulation services and platforms that enables SME end-users, especially from manufacturing and engineering, to access to simulation services and speed-up experimentation via Distributed Computing Infrastructures (DCIs) (e.g., cloud, grid, HPC cluster, etc.). CloudSME is creating a combination of PaaS and SaaS solutions to support MSaaS, i.e., a PaaS solution will allow simulation

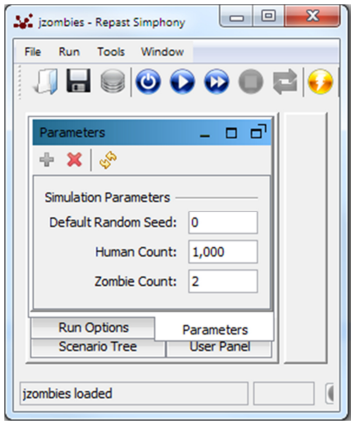

Figure 4: Repast simphony runtime.

software providers or consultant companies to build SaaS solutions for end-users and deliver MSaaS.

An important feature of this is that given the complexity of cloud-based solutions, the access mechanism to the simulation software should completely hide the complexity and potential heterogeneity of the cloud platform and should allow the simulation software to utilize different types of clouds based on their deployment models and the underlying cloud middleware. The CloudSME project does this by combining Cloud Broking services (provided by CloudBroker, CH (www.cloudbroker.com)) and WS-PGRADE/gUSE workflow development and deployment services (provided by MTA SZTAKI, HU) (Kacsuk et al. 2012) into the CloudSME Simulation Platform (CSSP). This is the PaaS that supports the deployment of simulation services and their access to HPC cloud resources in a user-transparent way.

There are two types of end-users involved in this project that demonstrate how different MSaaS solutions can be developed: simulation software providers and end-users who either use simulation to improve their business or use simulation software as part of their business offering. Simulation software providers currently include SIMUL8 (UK), ASCOMP (CH), INGECON (ES) and 2MORO (FR). End-users currently include SAKER SOLUTIONS (UK), PODOACTIVA (ES), EUROBIOS (FR) and CUTTING TOOLS (UK). The development of repast with the CSSP represents another case study in MSaaS. The Centre for Parallel Computing at the University of Westminster (UK) leads the project. Brunel University (UK) has the responsibility for managing the development of the cloud-based products and end-user adoption. CloudSigma (CH), SZTAKI (HU), University of Westminster (UK) and UNIZAR (ES) support the project with cloud resources.

Figure 5 shows the main parts of the CSSP and the split into three layers: the Simulation Application Layer, the Cloud Platform Layer and the Cloud Resource Layer. The Cloud Platform Layer is split into two: CloudBroker and gUSE. To create a simulation service on the platform the simulation software must be deployed on one or more of the clouds indicated in the figure. For each of these clouds, CloudBroker must have a cloud adaptor that allows an instance of the simulation software to be created, run and managed on a virtual instance on that cloud. The simulation software must be converted for deployment on a cloud. The approaches to do this vary on operating system. Typically the simulation executable is

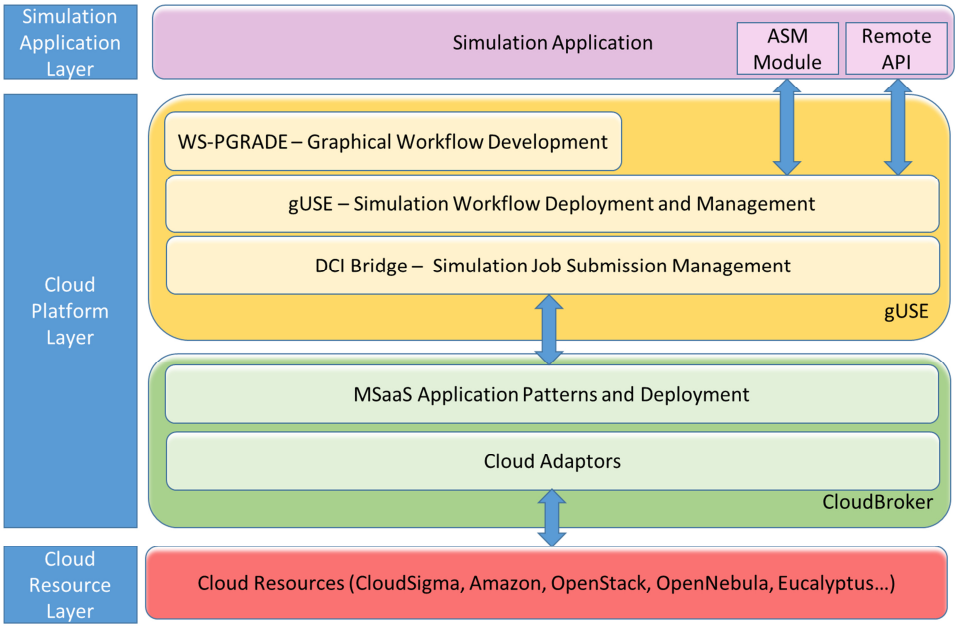

Figure 5: CloudSME Simulation Platform.

uploaded to a repository with runtime scripts that configure the software for execution (when jobs are submitted for processing). These executables and scripts form the MSaaS Application Patterns and Deployment configurations. The CloudBroker services are accessed via a web-based workflow management system. Here a workflow is created using WS-PGRADE that specifies how the software is run and managed. The workflow is stored on a workflow server (gUSE) and launched either via a web-based portal or some API (using gUSE's ASM Module or REMOTE API). For flexibility the workflow is "agnostic" and therefore has the functionality to run on any Distributed Computing Infrastructure (DCI) as long as the interface is supported by the workflow manager via the DCI Bridge. To run a simulation service the user runs the workflow and supplies the appropriate models and data. The workflow manager then manages this across the DCI Bridge on to CloudBroker and across multiple cloud instances. We now describe the current implementations of CSSP on CloudBroker and WS-PGRADE.

4.1 Cloud-Broker

The CloudBroker (CB) Platform is a web-based application store for the deployment and execution of compute-intensive scientific and technical software in the cloud. It is provided as a cloud service that allows for on-demand, scalable and pay-per-use access via the Internet. CB currently offers the platform as public version under https://platform.cloudbroker.com, as hosted or in-house setup, as well as licensed software. This also means that the platform can be run at different physical places and under different legislation if desired.

CB uses IaaS from resource providers and provides PaaS to software vendors and SaaS to end-users. It offers a marketplace where users can register, deploy, charge for and use their own cloud resources and application software or use resources and software provided by others (e.g., CB itself). Surcharges for platform usage are derived as percentage of the resource and software prices. From this follows a freemium model, that is, if resource providers and software vendors set zero prices for their resources and software, the corresponding platform usage is also for free.

CB incorporates adapters both to public and private cloud infrastructures, and both to compute and storage resources. Currently supported technologies include:

- Amazon EC2 compute and S3 storage resources
- IBM SmartCloud Enterprise compute (SoftLayer in preparation) and Nirvanix storage resources
- OpenStack compute resources (storage in preparation)
- Eucalyptus compute and storage resources
- OpenNebula compute resources (in preparation)

On the application side, CB is cross-domain and supports all kinds of non-interactive, batch-oriented applications, both serial and parallel ones, with a focus on software running under Linux. Windows is also supported. CB has already ported various software in different scientific and technical fields to the platform, in particular in chemistry, biology, health and engineering (see www.sci-bus.eu).

CB can be accessed in several ways. Its main two operation modes to manage and use software in the cloud are either as direct front-end or as back-end middleware service. For the former, the platform can be accessed with any regular web browser as a service via the internet. This is fine for first-time and individual usage. However, for frequent, advanced and automatic usage, API access is provided. These include REST web service interface, Java client library and Linux shell command line interface (CLI). Via its different APIs, CB can be utilized by front-end software as middleware to allow access to applications in the cloud.

CB has been integrated with the WS-PGRADE framework within the FP7 SCI-BUS project ("SCIentific gateway Based User Support" – www.sci-bus.eu). SCI-BUS aims at making scientific gateways available that can transparently utilize different distributed computing infrastructures. The

corresponding web portals provide workflows using the WS-PGRADE framework and can access applications on commercial and private cloud infrastructures using the CB Platform.

The current CB Platform incorporates two main types of security features:

- communication layer security using SSL transport layer encryption both between client and platform and between platform and cloud infrastructures
- authentication and authorization security using login and password for each user and different levels of organization and user roles

The platform employs industry standard application and server technology and is typically operated in industry standard secure data centers. The different cloud infrastructures it utilizes usually also provide authentication mechanisms, isolated virtual machines and security-certified cloud technologies and data centers.

4.2 Workflow Management

Overall, to develop an application to run on a cloud or a grid, a developer will use the graphical environment of WS-PGRADE to describe the sequence of tasks to run the application by describing the workflow of the application. A workflow is a directed acyclic graph of nodes and arcs where a node represents a specific task and an arc (typically) a file transfer between tasks. Arcs are connected to specific ports on each node to distinguish actions specific to those edges. Nodes can be configured to perform virtually any computing task via a visual interface. Some specialist nodes have been created to support High Performance Computing (HPC) tasks. Once an applications developer has created a workflow, it is saved in the gUSE Services environment. This a complete environment for workflow support and execution. It provides secure access and authorization to DCI resources by using appropriate security certification. In the CSSP architecture this transparently uses the security processes of the CloudBroker platform. It enables the management, storage and execution of workflows and has flexible deployment options (single hosting/distributed hosting to optimize resource usage or increase performance). It also offers data services (user database, proxy credential database, application code database, workflow database, application history and change log database, etc.) and control services (e.g., job submission, workflow enactment, etc.). The gUSE Information System supports workflow discovery and naming services. To execute a workflow, gUSE uses the DCI-BRIDGE to access the resources of various DCIs (in the CSSP these are the virtual machines of the cloud provider selected in the workflow). The DCI-BRIDGE is a web service-based application that provides standard access to DCIs via DCI plug-ins. Workflows are submitted and managed using the Basic Execution Service (BES) interface. This hides the access protocol and the technical details of the DCIs and uses a the standardised job description language JSDL. BES uses a job registry, an input queue, an upload manager to manage the execution of jobs generated by a workflow.

WS-PGRADE/gUSE has been developed by MTA SZTAKI as an open source software that is available in Sourceforge at: http://sourceforge.net/projects/guse. The framework is very popular among the different user communities. That is proven by the fact that so far more than 5,000 downloads from more than 40 countries have been done from Sourceforge. It is actively used by many different user communities and NGIs both in and outside Europe. In the SCI-BUS project 13 different user communities and four companies are developing their own gateway based on WS-PGRADE.

5 CASE STUDY

To illustrate how the CSSP can be used to support MSaaS for ABMS, we present a short case study using the JZombies model, a well-documented ABMS model that is used as a tutorial for the repast for java edition http://repast.sourceforge.net/docs/RepastJavaGettingStarted.pdf.

5.1 Cloud Deployment

The JZombies model is a non-terminating simulation, where zombie agents move towards the grid point with the most human agents and infect the latter by turning them into zombies. First, we modified the model to a terminating simulation, where the end point is when all the humans have turned into zombies. For illustration, each run consists of N replications. We have multiple runs (experiments) that show how we divide work across a cloud.

A model with its required repast plugins are deployed on different cloud resources supported by the CSSP, i.e., Amazon EC2, BIFI cloud (OpenStack Grizzly – provided by the University of Zaragoza, ES), and UoW cloud (OpenStack Folsom – provided by the University of Westminster, UK). All cloud resources support both Linux and Windows applications apart from UoW cloud, that currently supports only Linux applications. In order to be able to run the application in all available cloud resources, we developed both Windows Batch and Shell Script executables. The only requirement for the cloud resources was to have java runtime installation. In the following example repast will run in Windows on Amazon EC2 and BIFI and in Linux on UoW.

5.2 Workflow Creation

The workflow creation is supported by the WS-PGRADE. The first step is to use the graph editor for creating the workflow. The graph editor is a Java Network Launch Protocol (JNLP) application and provides a user-friendly drag-and-drop environment, where the structure of the workflow can be created. Figure 6 illustrates the graph editor, where the bigger squares denote jobs and the attached smaller squares denotes input and output ports (green and grey squares, respectively). Each job can have more than one input and output. In the example of Figure 6, the initialize node generates two different outputs, that are then fed to the respective job nodes. Once all the job runs are finished the outputs are collected to the output collector node.

Simulation experiments are usually parametric runs and each set of parameters executes for a number of replications. CSSP can distribute the parametric runs and the replications to different nodes on the available cloud resources. The jobs are distributed and managed from WS-PGRADE over the DCI Bridge to CloudBroker. CloudBroker then creates and manages the instances on the different clouds as instructed.

5.3 Workflow Configuration

The graphical workflow is configured at the WS-PGRADE web portal. The simplest structure is a one job workflow. An example of a simple one job workflow configuration for a parameter sweep execution of

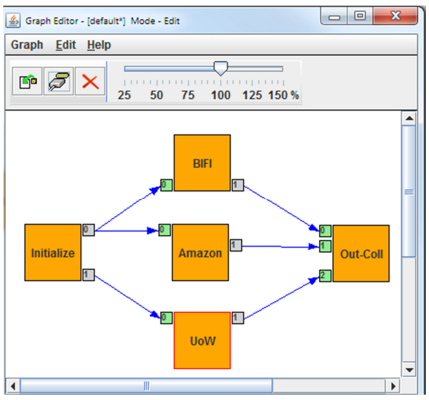

Figure 6: WS-PGRADE graph editor.

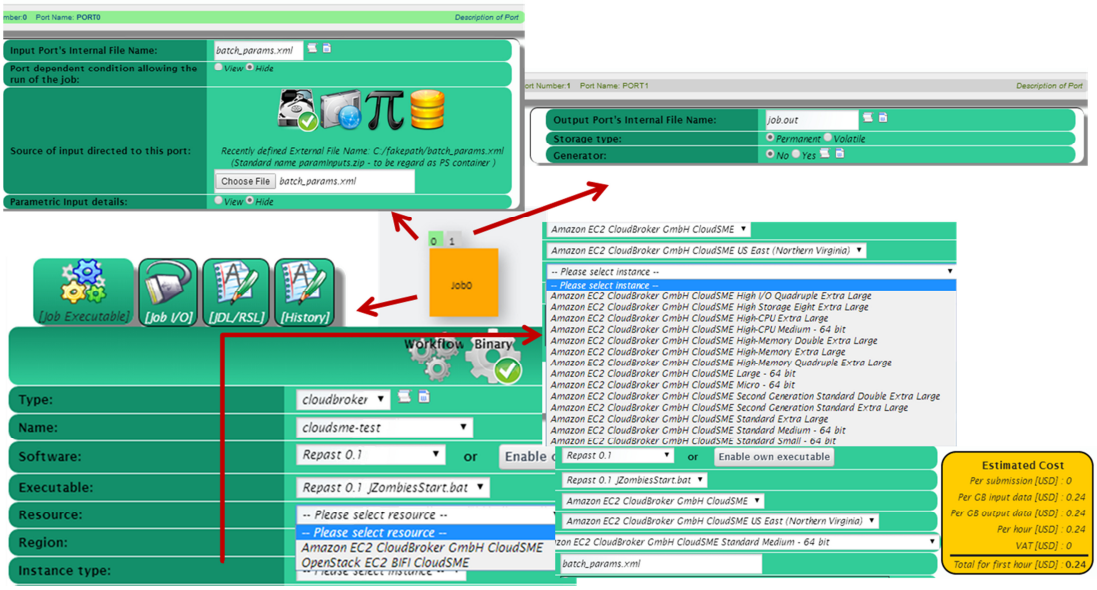

Figure 7: CloudSME workflow configuration.

the JZombies repast model is portrayed in Figure 7.

In the job node, the first step is to select the type of DCI. As mentioned earlier, CSSP's cloud resources are managed by CB. CB account authentication is needed in order to be able to access the portal. Then the software that will execute in the workflow (in this case Repast 0.1 is the repast model) and the executable (Windows batch file) are selected. From the available cloud resources, once a selection of the instance type is made, an estimation of the job cost is displayed. The next step is to configure the input and output ports. In repast batch runs, the set of parameters is indicated in an XML file, therefore this is provided in the input port. The output port will export the resulting files from the job run. The output files can be downloaded after the completion of the job.

An example workflow that can utilize all the available cloud resources in CSSP (where repast is deployed) can be seen in Figure 8. This takes a set of repast simulation runs and distributes them over three clouds (these could be the same one – we do this to show the flexibility of this workflow approach). The initialize node will output the XML parameter file that is required as input for every repast simulation run. Then, each node can initiate several instances each in the three CSSP clouds on which repast is deployed. (i.e., Amazon EC2 and BIFI will run the Windows application and UoW will run the Linux application). It is notable to mention that for the commercial cloud there is cost estimation information, while for the academic clouds, the cost is zero. Also, in the job configuration view, it can be seen that for the BIFI and Amazon EC2 clouds, the Windows batch executables is selected, while for the UoW cloud, it is the shell script executable.

Furthermore, the initialize node can be configured as a generator node. A generator replicates the required input file(s) as many times as indicated (or the size of the cloud resources dictate). In this way, many instances can be instantiated automatically. For example, if the user wishes to run X number of replications, the generator replicate the input file(s) X times and the workflow manages the distribution of jobs across the available resources.

The way to create and configure workflows is by no means restricted by the above examples. These are some indicative examples of the CSSP capabilities. The extended demonstration of the infrastructure will be the subject of a future article.

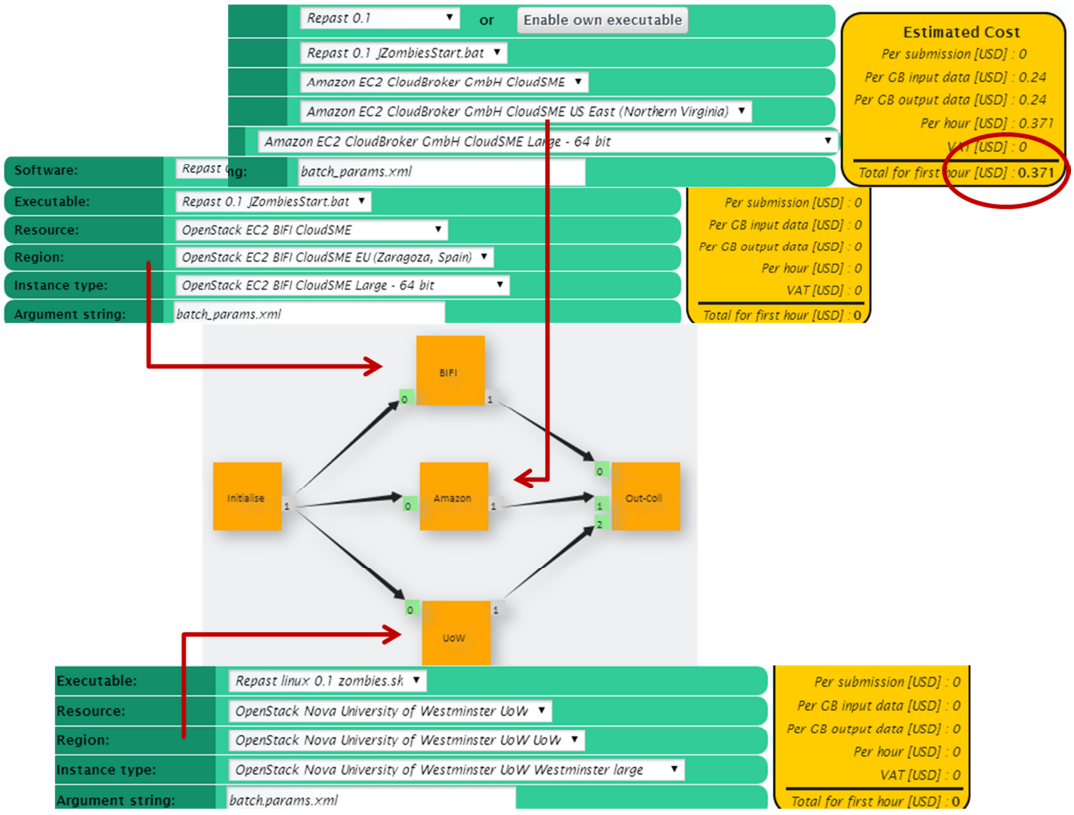

Figure 8: Jobs distribution in different CSSP cloud resources.

6 EXPERIMENTS

One of the cloud performance aspects of those listed in NIST (2013) is the Benchmark Performance (BP). BP involves conducting several tests to measure the relative performance of a system.

We ran the repast JZombies model on two public clouds, one commercial and one academic. The simulation ran for 1,000 runs. The commercial cloud provider, Amazon EC2, offers "unlimited" resources, however the academic cloud provider, BIFI, University of Zaragoza, ES, offers limited cloud resources. Also, cloud providers offer different cloud sizes, i.e., CPUs with different number of cores. The cloud size reflects the pricing model for the cloud usage. For consistency purposes, a medium size cloud was selected in both cases, i.e., 4-cores CPUs.

The first experiment involves running the simulation over one instance for both clouds. For the second experiment, the 1,000 replications were distributed over four instances on a single cloud resource each time, again for both clouds. For the third experiment, the same number of runs were distributed over ten instances on a single cloud resource each time and for both clouds. The starting parameters were: number of zombies = 2 and number of humans = 1000.

6.1 Results and Discussion

The graphs in Figure 9 illustrate (a) the total time to finish the cloud-based simulation when running one, four and ten instances on each of the two clouds of the experiment, and (b) the speed-up for the respective runs. From the graphs, it is clear that the limited resources of the BIFI cloud resulted in increased time for

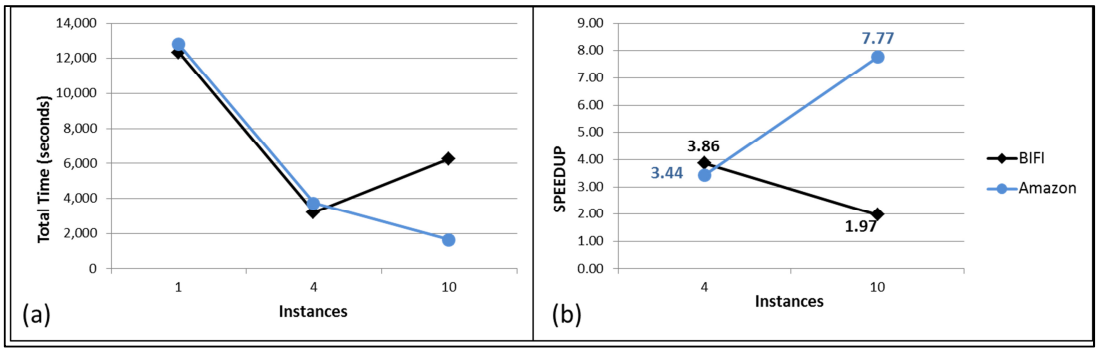

Figure 9: (a) Total time, and (b) Speed-up of 1,000 replications for different number of instances.

more than four instances, while the "unlimited" resources of the Amazon cloud continued to speed-up the runs. This is caused by the instances queuing for instantiation. The achieved speed-up depends on factors such as the size of the ABMS model. In our example, the JZombies model is a small simulation. One replication executes in 20 seconds and therefore the instantiation adds a considerable overhead in the cloud-based simulation.

Figures 10 shows the running instances elapsed time. The behavior of both BIFI and Amazon clouds when four instances are instantiated is similar (solid lines). However, when the simulation runs in ten instances, there is a considerable discrepancy (dashed lines). This is an expected behavior, since the limited resources of the academic cloud will put some of the jobs in a queue before starting the execution.

7 CONCLUSIONS

This paper provided a tutorial on running repast ABMS on the CloudSME Simulation Platform. To demonstrate the different layers and interfaces of the infrastructure, the well-known repast JZombies ABMS model was deployed and configured for the different ways that a parametric job can run and be distributed on different cloud resources. It has demonstrated how ABMS can use cloud computing. The authors welcome collaboration and look forward to developing further cloud-based ABMS applications.

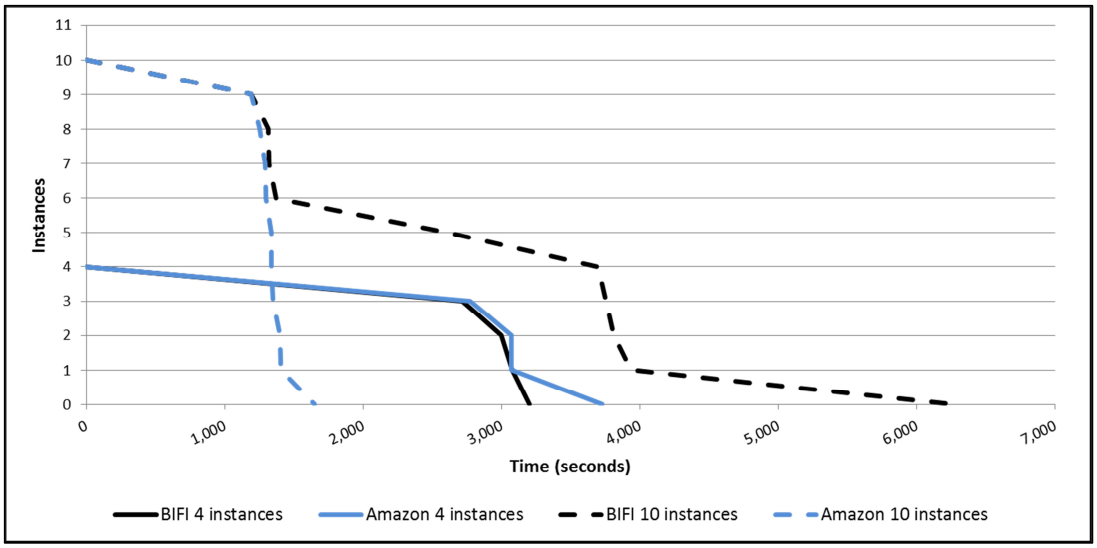

Figure 10: Running elapsed time.

ACKNOWLEDGMENTS

This work is funded by the CloudSME – Cloud-based Simulation platform for Manufacturing and Engineering project No. 608886 (FP7-2013-NMP-ICT-FOF).

REFERENCES

Hawe, G. I., G. Coates, D. T. Wilson, and R. S. Crouch. 2012. "Agent-Based Simulation for Large-Scale Emergency Response: A Survey of Usage and Implementation." *ACM Computing Surveys* 45(1):Article 8.

Hill, R., L. Hirsch, P. Lake, and S. Moshiri. 2013. *Guide to Cloud Computing: Principles and Practice*. London, UK: Springer-Verlag.

Holland, J. H. 1995. *Hidden Order: How Adaptation Builds Complexity.* Basic Books.

Kacsuk, P., Z. Farkas, M. Kozlovszky, G. Hermann, A. Balasko, K. Karoczkai, and I. Marton. 2012. "WSPGRADE/gUSE generic DCI gateway framework for a large variety of user communities." *Journal of Grid Computing* 10(4):601-630.

Loefstrand, B., J. Bystroem, and M. Johansson. 2003. "HLA Design Patterns for Agent Based Federates." In *Proceedings of the Euro Simulation Interoperability Workshop*, Article 03E–SIW–076.

Macal, C. M. and N. J. North. 2010. "Tutorial on Agent-Based Modelling and Simulation." *Journal of Simulation* 4(3):151-162.

Macal, C. M. and M. J. North. 2006. "Tutorial on Agent-based Modelling and Simulation Part 2: How to Model with Agents." In *Proceedings of the 2006 Winter Simulation Conference*, Edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol and R. M. Fujimoto, 73-83. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mell, P. and T. Grance. 2011. The NIST Definition of Cloud Computing. Accessed May 08, 2014. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

NIST. 2013. NIST Cloud Computing Standards Roadmap. Accessed May 08, 2014. http://www.nist.gov/itl/cloud/upload/NIST_SP-500-291_Version-2_2013_June18_FINAL.pdf.

North, M. J., N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, and P.. Sydelko. 2013. "Complex Adaptive Systems Modeling with Repast Simphony." *Complex Adaptive Systems Modeling* 1(3). http://www.casmodeling.com/content/1/1/3.

Taylor, S. J. E. 2014. *OR Essentials: Agent-based Modelling & Simulation*. Basingstoke, Hampshire, UK: Palgrave Macmillan Ltd.

AUTHOR BIOGRAPHIES

SIMON J. E. TAYLOR is the convener of the new phase of Grant Challenge activities. He is the Founder and Chair of the COTS Simulation Package Interoperability Standards Group under SISO. He is the Editor-in-Chief of the UK Operational Research Society's (ORS) Journal of Simulation and the Simulation Workshop Series. He is Reader in the Department of Computer Science at Brunel University and leads the M&S Group. His email address is simon.taylor@brunel.ac.uk.

ANASTASIA ANAGNOSTOU is a Research Fellow at the Department of Computer Science, Brunel University London. She holds a PhD in Hybrid Distributed Simulation, a MSc in Telemedicine and e-Health Systems and a BSc in Electronics Engineering. Her research interests are related to the application of modeling and simulation techniques in the Healthcare and Industry. Her email address is anastasia.anagnostou@brunel.ac.uk.

TAMAS KISS is a Reader at the Department of Business Information Systems, Faculty of Science and Technology, University of Westminster. He holds a PhD in Distributed Computing and his research interests include distributed and parallel computing, cloud, cluster and grid computing. He has been involved in several European research projects. Currently he is Project Director of the FP7 CloudSME project that develops a cloud-based simulation platform for manufacturing and engineering SMEs. His email address is T.Kiss@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. He served as full professor at the University of Miskolc and at the Eötvös Lóránd University of Science Budapest. He has been a part-time full professor at the Cavendish School of Computer Science of the University of Westminster. He is co-editor-in-chief of the Journal of Grid Computing published by Springer. His email address is kacsuk.peter@sztaki.mta.hu.

GABOR TERSTYANSZKY is a Professor of Distributed Computing at the Faculty of Science and Technology at the University of Westminster. He is the Director of the Centre for Parallel Computing. He attracted in excess of £1.4 million in European and UK research funding to the University of Westminster. He has led or contributed to numerous collaborative research projects funded by European Commission Research Framework Programmes and by UK Research Councils. His email address is G.Z.Terstyanszky@westminster.ac.uk.

NICOLA FANTINI is the CEO of the CloudBroker GmbH, CH, and has been software engineer, project manager and entrepreneur in the field of software development for more than 20 years. He holds a MSc degree in Computer Science and Business Administration from the University of Zurich, CH. He has been involved in and leading projects across industries, providing operational, commercial, financial as well as technical solutions. His email address is Nicola.Fantini@scaletools.com.