A Workflow Architecture for Cloud-based Distributed Simulation

NAUMAN RIAZ CHAUDHRY, Department of Computer Science, University of Gujrat, Gujrat, Punjab, Pakistan
ANASTASIA ANAGNOSTOU and SIMON J. E. TAYLOR, Department of Computer Science, Brunel University London, London, United Kingdom

Distributed Simulation has still to be adopted significantly by the wider simulation community. Reasons for this might be that distributed simulation applications are difficult to develop and access to multiple computing resources are required. Cloud computing offers low-cost on-demand computing resources. Developing applications that can use cloud computing can be also complex, particularly those that can run on different clouds. Cloud-based Distributed Simulation (CBDS) is potentially attractive, as it may solve the computing resources issue as well as other cloud benefits, such as convenient network access. However, as possibly shown by the lack of sustainable approaches in the literature, the combination of cloud and distributed simulation may be far too complex to develop a general approach. E-Infrastructures have emerged as large-scale distributed systems that support high-performance computing in various scientific fields. Workflow Management Systems (WMS) have been created to simplify the use of these e-Infrastructures. There are many examples of where both technologies have been extended to use cloud computing. This article therefore presents our investigation into the potential of using these technologies for CBDS in the above context and the WORkflow architecture for cLoud-based Distributed Simulation (WORLDS), our contribution to CBDS. We present an implementation of WORLDS using the CloudSME Simulation Platform that combines the WS-PGRADE/gUSE WMS with the CloudBroker Platform as a Service. The approach is demonstrated with a case study using an agent-based distributed simulation of an Emergency Medical Service in REPAST and the Portico HLA RTI on the Amazon EC2 cloud.

CCS Concepts: • **Networks** → *Network services*; *Cloud computing*; • **Computing methodologies** → *Modeling and simulation*; *Simulation types and techniques*; *Distributed simulation*;

Additional Key Words and Phrases: Modeling & simulation, distributed simulation, HLA, cloud computing, scientific workflows, science gateway

ACM Reference format:
Nauman Riaz Chaudhry, Anastasia Anagnostou, and Simon J. E. Taylor. 2022. A Workflow Architecture for Cloud-based Distributed Simulation. *ACM Trans. Model. Comput. Simul.* 32, 2, Article 15 (February 2022), 26 pages.
https://doi.org/10.1145/3503510

Authors' addresses: N. R. Chaudhry, Department of Computer Science, University of Gujrat, Gujrat, Punjab, Pakistan; email: nauman.riaz@uog.edu.pk; A. Anagnostou and S. J. E. Taylor, Department of Computer Science, Brunel University London, London, United Kingdom; emails: {anastasia.anagnostou, simon.taylor}@brunel.ac.uk.

1049-3301/2022/02-ART15 $15.00
https://doi.org/10.1145/3503510

1 INTRODUCTION

Distributed simulation techniques can link and interoperate models over a network to build larger models, share the processing demands of simulation execution across multiple computers, and potentially speed up simulation execution [1, 2]. Apart from defense applications, the field has yet to be adopted by the mainstream simulation community. Reasons for this might be that distributed simulation applications are difficult and complex to develop. Access to multiple computing resources is also needed to run these applications. Cloud computing offers convenient access to rapidly provisioned, on-demand computing resources [3]. Cloud computing support for distributed simulation, or **Cloud-based Distributed Simulation (CBDS)**, is attractive, as it would eliminate the need for local computing resources and facilitate remotely accessible deployment. However, multiple clouds with different APIs and characteristics, different middleware and service models mean that developing cloud-based applications can be difficult and complex. The development of a generalized approach to CBDS may be significantly challenging.

There has been large-scale investment in the development of distributed systems for science (or e-Science). These e-Infrastructures (cyberinfrastructures) are widely used by many scientific communities and make use of a range of **Distributed Computing Infrastructures (DCIs)** including computational grids, high-performance computing and, more recently, clouds. Application development for e-Infrastructures is also complex. **Workflow Management Systems (WMS)** have been created to simplify the use of these systems for end-users. Workflows consisting of a network of tasks are developed in dedicated programming languages or via graphical user interfaces. These tasks are translated into jobs that are run on specified computing infrastructures. This approach to the simplification of complex technologies has been highly successful as shown by the widespread adoption in some scientific communities [4].

Given the success of e-Infrastructures and WMS, we wanted to investigate if CBDS could benefit from this approach. This article reports on the results of this investigation and the development of our **WORkflow architecture for cLoud-based Distributed Simulation (WORLDS)**.

The article is structured as follows: A review of distributed simulation, cloud computing, and related CBDS work is presented in Section 2. The limitations of CBDS systems and applications as well as the issues involved in creating a general approach are discussed. Section 3 introduces e-Infrastructures and WMS and considers how these could be used for CBDS. This section then presents our WORLDS architecture. In Section 4, we give a proof-of-concept implementation of WORLDS. This implementation is demonstrated in Section 5 with a case study using an agent-based distributed simulation of an Emergency Medical Service in REPAST and the Portico RTI on the Amazon EC2 cloud. Section 6 discusses the limitations and potential of WORLDS and concludes the article with future work.

2 BACKGROUND

This section presents the background to the main concepts in this article by first introducing distributed simulation and cloud computing. Related work in CBDS is then presented. The limitations of these approaches and how the state-of-the-art could be extended to developing a generalized architecture for CBDS are then discussed.

2.1 Distributed Simulation

Distributed simulation can be defined as "the distribution of the execution of a simulation program across multiple processors" [2]. Using distributed simulation techniques, it is possible to simulate a single model by dividing it across several processors or to simulate multiple models by joining together the different processors on which they run. A significant reason for the reluctance to adopt

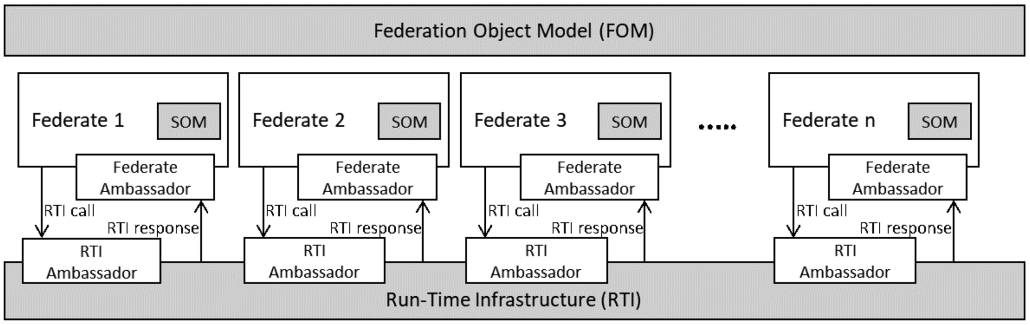

Fig. 1. The High Level Architecture.

distributed simulation is the high degree of technical expertise necessary for implementing applications. A long-term goal in this field has been to simplify the development of these applications [5]. Standardizing distributed simulation practices might help to overcome this obstacle to formalize simulation protocols and improve reusability and interoperability of components [6]. Different standards for distributed simulation have been established, including **Distributed Interactive Simulation (DIS)** [7] and the **High Level Architecture (HLA)** [8]. Both were developed by the US Department of Defense with the initial objective of supporting military training simulations. These simulations could involve thousands of objects (representing soldiers, vehicles, aircraft, etc.) that need to be connected together for the exchange of information. The standard continues to be updated with the last version released in 2010 [9]. As shown in Figure 1, in the HLA an object or collection of objects is termed federate and the collection of federates is termed the federation. Federates are able to interact with each other via middleware called the **Run-Time Infrastructure (RTI)**. This implements the HLA's Federate Interface Specification and interacts with federates via RTI and Federate Ambassador implementations. The **Object Model Template (OMT)** allows information exchange to be defined. The **Simulation Object Model (SOM)** defines this for a federate and the **Federation Object Model (FOM)** defines this for the entire federation. Other forms of distributed simulation exist [10]. However, we shall restrict our discussions to HLA-based distributed simulation in this article.

2.2 Cloud Computing

The term "cloud" refers to the provisioning of services that are highly elastic in terms of networking scope, computing power, storage facilities, or application range. The **US National Institute of Standards and Technology (NIST)** proposed a definition that is widely accepted: "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" [3]. Resource demand is matched rapidly at any specific time by using fast provisioning facilities to operate those resources. This "elasticity" means that cloud computing can scale resources to meet the user demands in terms of both computational resources and storage [11]. Resources are usually fixed at the point of use although new approaches are being developed to auto-scale cloud resources according to some performance metric [12]. Billing is applied using a pay-per-use policy. Cloud therefore reduces costs by "hiring" computational resources rather than having local computing facilities. However, cloud can have additional costs such as data transfer. There are well-known service models. **Software as a Service (SaaS)** involves the software

applications running on the cloud infrastructure accessed via thin clients, web browsers, or program interface. **Platform as a Service (PaaS)** facilitates the deployment and execution of software applications and the management of cloud infrastructure. **Infrastructure as a Service (IaaS)** provides services to provision cloud resources, operating systems, firewalls, and so on. There are also several cloud types including private (single organization use), community (multiple organizations use), public (open use), and hybrid (multiple clouds presented as a single cloud). The service paradigm has roots in **Service-Oriented Architecture (SOA)** and has led to many variants (e.g., **"everything" as a service (XaaS)**) [13]. Most applications developed as services tend to be deployed to single clouds (e.g., Amazon, Azure) and can be prone to "vendor lock-in," as different clouds have different APIs. Given the growth in the number of cloud providers, locked-in services cannot easily take advantage of competitive pricing of an emerging cloud market without potentially costly redevelopment. Also, national data protection laws and government/corporate policies can restrict the choice of cloud.

2.3 Distributed Simulation Architectures

The late 1990s and early 2000s saw three technological advances: the original architecture for grid computing (the pre-cursor to e-Infrastructures) along with the **Globus grid computing toolkit (GT)** [14], SOA and web services, and the HLA. Research into developing service-oriented services for distributed simulation or service-oriented simulation arguably began in the early 2000s with the **Extensible Modeling and Simulation Framework (XMSF)** that proposed the extension of the HLA with web service technologies to support simulation development and interoperability [15]. At the same time, approaches to service-oriented grid computing were being developed that resulted in the **Open Grid Service Architecture (OGSA)** and grid services [16]. The use of these in the context of XMSF was discussed by Pullen et al. [17]. Xie et al. [18] implemented these in the HLAGrid prototype that investigated the feasibility of using GT3 grid services to run HLA-based distributed simulation. Theodoropoulos et al. [19] extended this to support agent-based simulation using REPAST [20]. HLAGrid was further developed using GT4 resulting in the SOAr-DSGrid to support general distributed simulation [21]. This research continued with the Decoupled Federate Architecture that added fault tolerance and simulation cloning functionality [22]. Building on experiences with SOAr-DSGrid led to the more comprehensive **Service-Oriented HLA RTI framework (SOHR)** that added more HLA services [23]. This was used to investigate optimistic protocols running over networks of virtual machines [24]. Feng et al. [25] and Zhang et al. [26] presented early examples of how an RTI with GT4 services could be realized as a PaaS.

2.4 Related Work: Cloud-based Distributed Simulation

With the emergence of cloud computing and cloud service models, researchers began to investigate how distributed simulation could take advantage of this technology. He et al. [27] proposed a secure cloud-based HLA RTI that could run on a cloud using on-demand computing resources. It demonstrated several key features of a cloud-based RTI using a local cluster. Vanmechelen, De Munck, and Broeckhove [28] developed a conservative distributed simulation using Amazon EC2 services using a manager-worker approach to scale the simulation over the resources of a single image. Performance results were reported that compared different variants of their conservative protocol. Guan, De Grande, and Boukerche [29] proposed a cloud architecture to enable HLA-based distributed simulations on cloud that used virtual resource management to use cloud computing resources if local computing resources were not sufficient. A comparison between a local grid computing implementation versus a cloud implementation demonstrated lower energy consumption for cloud and comparable performance (Portico RTI, 15 physical hosts versus 15 virtual ones). It appears that a single cloud instance was used (actual cloud not specified). This work was further

developed into a multi-layered elastic cloud simulation scheme supporting the capability of automatic deployment, reduction of energy consumption, and local to cloud resource scaling using a deployment management scheme [30]. It argued that this approach could support multiple clouds by using an architecturally layered approach. Experiments were performed for several distributed simulation protocols including the HLA and again compared local grid to cloud resources (a single Amazon EC2 instance mapping several virtual CPUs). The paper reported acceptable comparative performance results (slower due to virtualization overheads) on cloud balanced with low costs, energy consumption, and so on. D-Mason is the distributed version of the agent-based modelling library MASON [31]. The simulation is partitioned into cells that are mapped onto **logical processes (LPs)**. Each LP synchronizes with its neighbors using a publish/subscribe mechanism. The cloud-based version of D-Mason was conceptualized as a simulation as a service environment where a D-MASON simulation is mapped to Amazon EC2 cloud resources using StarCluster [32]. GridSpice was a simulation platform for the distributed agent-based simulation of smart power grids consisting of a transmission network, distribution networks, and power generators. These elements are mapped onto the Amazon EC2 cloud using StarCluster. The simulation was implemented in REPAST and used its time synchronization features to synchronize time across the distributed simulation. The **Scalable Electro-Mobility Simulation (SEMSim)** platform was developed to study the impact of large-scale electromobility schemes on a city's infrastructure [33]. The SEMSim Cloud Service consisted of a distributed simulation with two parts: an agent-based traffic simulation and a discrete-event power system simulation. A Dispatch Server managed simulation experiment submission to cloud. The service was deployed on the Google Compute Engine and reported good performance with respect to dedicated HPC. Megaffic was another distributed agent-based simulation that simulated traffic networks [34]. It used Bulk Synchronous Processing to synchronize between the elements of the distributed simulation. Checkpoints were used to rebalance processing load. The simulator was implemented using the Google Compute Engine cloud. Zaheer et al. [35] demonstrated how locality-aware logical process placement could reduce network overheads in busy cloud data centers. General concerns of data security, service reliability, and slower execution due to potential sharing of instances on physical machines with other users have also been discussed in this area [36, 37]. Simulation partitioning and load balancing difficulties involved in using on-demand multi-instance multi-core systems such as cloud can also be encountered [38].

Several authors have also explored **Modelling & Simulation as a Service (MSaaS)**. For example, Liu et al. [39] **(Simulation as a Service (SIM as a service)** as well as **Modelling as a Service (MaaS), Execution as a Service (EaaS), Analysis as a Service (AaaS)**, and **Simulation Resource as a Service (SRaaS)**) and Zehe et al. [40] **(Simulation Software as a Service (SSaaS))**. Siegfried et al. [41], Cayirci [42], Taylor et al. [43], and Kiss et al. [44] have proposed forms of MSaaS. While useful in advancing the discussion of what cloud-based simulation services could achieve, these contributions tend to be conceptual in nature, discussing the potential of some form of simulation-related service, or focus on one issue (e.g., theoretical architecture, security, business models, grand challenges).

2.5 Discussion

There have been several attempts at developing CBDS. A range of cloud resource management schemes have been investigated. Those that use multiple cloud instances tend to demonstrate results using a local cluster and not cloud. Examples that provision actual cloud instances appear to focus on single instance use and not multiple instances. Some have discussed how multiple instances could be used but have not implemented this feature. This may be due to the costs involved as, although cloud computing is cheaper than purchasing equivalent distributed computing

systems, extensive investigation using multiple cloud instances might be prohibitively costly to researchers. Several works have investigated how HLA federations can be run on cloud and several have focused on other variants of distributed simulation. There is evidence of a small number of approaches continue to be developed. Only one of these approaches could be considered generalizable (for HLA-based distributed simulation). The others focus on a specific technique (distributed agent-based simulation) or domain (power systems, electromobility). Work on MSaaS and associated concepts have explored issues in this area but are conceptual in nature.

To summarize our observations, distributed simulation applications are difficult and complex to develop. Federates may need to run on their own "computer" and therefore accessing multiple computing resources is desirable. This need is especially true if multiple experiments are to be carried out that could run multiple federations with different parameters in parallel. Cloud computing is a possible approach to multiple computing resource provision. It also has the benefit of remote network access. Application development with cloud computing is also complex. However, it is possible that once a cloud-based application has been developed, end-users can easily and repeatedly access and run it remotely with computing resources being provisioned automatically. End-users might also be able to choose which cloud they would prefer to use. Currently, although we cannot claim that our review is exhaustive (e.g., advances made by vendors), apart from a small number of domain-specific examples, there does not appear to be a generalizable approach to CBDS.

To summarize the above limitations, current approaches to CBDS have the following issues:

- Demonstrations/performance tests are conducted on a local cluster and not an actual cloud;
- Demonstrations/performance tests are conducted with a single instance and not multiple instances;
- There are many approaches to CBDS with, currently, no single approach emerging;
- Some approaches to CBDS investigate how HLA federations can run on cloud;
- Most approaches focus on a specific type of simulation technique and domain of application;
- MSaaS has been considered in terms of CBDS but is mostly conceptual in nature with no real demonstrations;
- These approaches to CBDS are highly complex and require advanced computing skills to develop and implement—they also develop technologies that are unique to the article/project with little evidence of reusing existing technologies (apart from RTIs, for example);
- No evidence of a supporting community (apart from specific simulation technologies (e.g., MASON, Portico);
- Most approaches are tied to one cloud (i.e., no "multi-cloud" approach);
- There is little or no research into a generic CBDS approach that can support a range of simulation techniques and domains;

In the next section, we present an approach that has had some success with cloud-based application development and discuss how this approach could be used for CBDS.

3 A WORKFLOW ARCHITECTURE FOR CLOUD-BASED DISTRIBUTED SIMULATION

The previous section discussed cloud-based approaches to distributed simulation. Pre-cloud research investigated how grid services could be used as the basis for some form of distributed simulation service. Arguably, cloud took researchers in a different direction. However, work on grid computing continued and matured into e-Infrastructures (sometimes called digital research infrastructures, e-Science infrastructures, or cyberinfrastructures). These have been developed to support the needs of different research communities to share common data, simulations, software, and computing resources across organizations [45]. An e-Infrastructure brings these

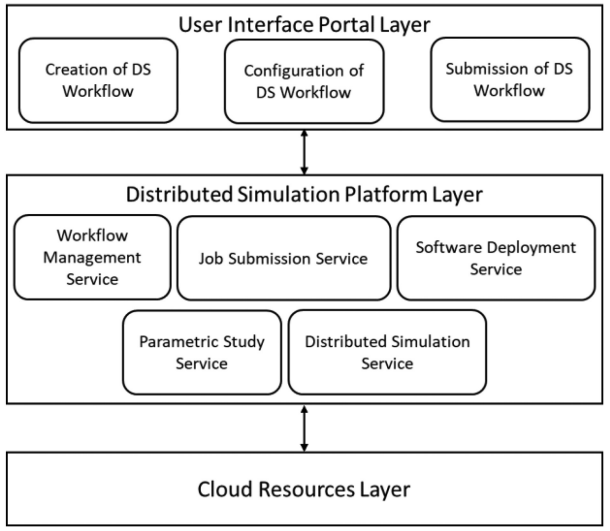

Fig. 2. Workflow Architecture for Cloud-based Distributed Simulation (WORLDS).

together as a virtual organization and provides single sign-on access (e.g., through eduGAIN [46]). Computing resources are managed as DCIs and can include clusters, networks of PCs, cloud, and high-performance computing facilities [47]. Users submit and manage jobs across these DCIs via grid computing job submission systems such as HTCondor [48]. e-Infrastructures have benefited from over 20 years of development, and there is a large worldwide community. Scientific **Workflow Management Systems (WMS)** have emerged as a means to process workflows and organize data sets for many scientific computing scenarios [49, 70]. Examples include Askalon [50], Chiron [51], Galaxy [52], Kepler [53], Pegasus [54], Swift [55], Taverna [56], Triana [57], and WS-PGRADE/gUSE [58] and are used extensively in fields such as astronomy, biology, computational engineering, and **Modeling & Simulation (M&S)** [4]. There are examples of where e-Infrastructures and WMS/Science Gateways have been extended to incorporate cloud (e.g., Pegasus and Amazon EC2 [59], Taverna and multiple clouds (Amazon EC2, Azure, Rackspace, clouds running OpenStack) [60], Galaxy and multiple clouds (Amazon EC2, clouds running OpenStack/OpenNebula) [61], the **Distributed Application Runtime Environment (DARE)** science gateway framework and multiple clouds (Amazon EC2, Eucalyptus, and Nimbus) [62] and WS-PGRADE/gUSE (via CloudBroker integration to multiple clouds (Amazon EC2, Azure, CloudSIGMA clouds running OpenStack/OpenNebula), and HPC Center resources deployed as clouds) [63]. Could these technologies be used to create distributed simulation applications?

WMS and supporting architectures introduced above, generally have elements of workflow management, job management, and cloud deployment. Figure 2 shows our **Workflow Architecture for Cloud-based Distributed Simulation (WORLDS)** based on a generalized workflow architecture. Users create the workflow in the User Interface Portal Layer. A workflow takes the form of a **Directed Acyclic Graph (DAG)** consisting of an array of nodes in which each node represents a specific task. An edge represents input and output (i.e., file transfer or external parameters). Once defined, workflows can be configured with application/platform-specific information for each task (e.g., the specific algorithm (script), input data, method of communicating outputs, cloud infrastructure) Ideally, users could also choose which cloud and resources to use for their federation (e.g., using one cloud instance with multiple CPUs, multiple cloud instances with single

CPUs) Once a workflow is configured, it can be submitted to the **Distributed Simulation (DS)** Platform layer. The DS Platform Layer provides services to manage and execute the workflow. To do this, the layer consists of five services. The **Workflow Management Service (WMS)** converts the workflow task to jobs. It composes jobs by adding specified software from a repository (the **Software Deployment Service (SDS)**) and converting associated shellscripts, parameters, and input/output communication requirements. It then manages the execution of these jobs following the dependencies defined in the workflow. The **Job Submission Service (JSS)** takes these individual jobs and their cloud requirements and submits these to the appropriate cloud resources. It takes any application specific deployment images (e.g., virtual machines, containers) specified in the task from the SDS. The service maintains status information for each job and, when completed, downloads the outputs from the cloud infrastructure. It also has some fault tolerance functionality (e.g., when jobs fail and require resubmission). The **Distributed Simulation Service (DSS)** deploys and runs the DS Manager on cloud. This is called by the DSS task in the workflow and directly runs the **Central RTI Component (CRC)** on cloud (via images and software from the Software Deployment Service). This must be directly deployed and executed as the federates require the IP address of the CRC to connect and join the federation. Once deployed, the DSS provides this IP address as an input parameter to the federate composition tasks of the workflow. Additionally, many simulation applications are used to perform experiments. These experiments typically simulate the same model with different parameters. To reflect this, a Parameter Study Service would require dedicated workflow tasks that generate the experiments and collect the results. The generator task would also be capable of specifying the maximum number of resources to be used to process the experiments in parallel. This allows users to easily translate experimentation specifications into the workflow. This is not shown in the example distributed simulation workflow but could be used to generate multiple federations running in parallel. Finally, the **Cloud Resources Layer (Cld)** contains the resources from different clouds.

Figure 3 shows a federation and a possible workflow. The federation is composed of several federates and an RTI. The aim of the workflow is to automate federation deployment and execution and could be composed of several steps [10]: Initialize (set up each federate with its model and data), Compose (to bring the federation together with the RTI/DS Manager and map these to DCI resources), Execute (to run the federation on the DCI, possibly running many federations in parallel), and Collect (of the results of multiple federations that have run in parallel). Figures 4 and 5 show how WORLDS would process this workflow. The **workflow (WF)** tasks are identified by labels.

Initialize: The task and data *Init(Names, FedData)* is sent to the WMS. *Names* consists of the names of the federates and *FedData* consists of the data, parameters, models, and so on, that will be used to update the federate software stored in the SDS. The WMS splits this into two messages. *Init(Names)* is passed to the DSS to begin the initialization of the DS software. On receipt, the DSS instructs the cloud management software to commission a cloud resource that will host the CRC (*InitCRCRes*) and to pass back the IP address of this resource (*CRCIP*). The DSS also validates the *Names* of the federates and passes this and the CRCIP back to the WMS (*DSData*). The WMS then instructs SDS via *Init(DSData, FedData)* and instructs it to update the relevant federations (*InitFeds*). When this is complete, the WMS returns the new *DSData* back to the WMS completing this task.

Compose: This task consists of two activities: the composition of the CRC and the composition of the Federates. This is shown as a parallel task, as these can potentially be done in parallel if the WMS allows this. Both follow a similar pattern. The WF sends *Names* and *CRCIP* to the WMS. The WMS forwards this to the DSS and instructs it to begin the composition of the CRC software. Relevant installation scripts, etc. and the CRC are then passed as *CRCSw* to the JSS and onto Cloud

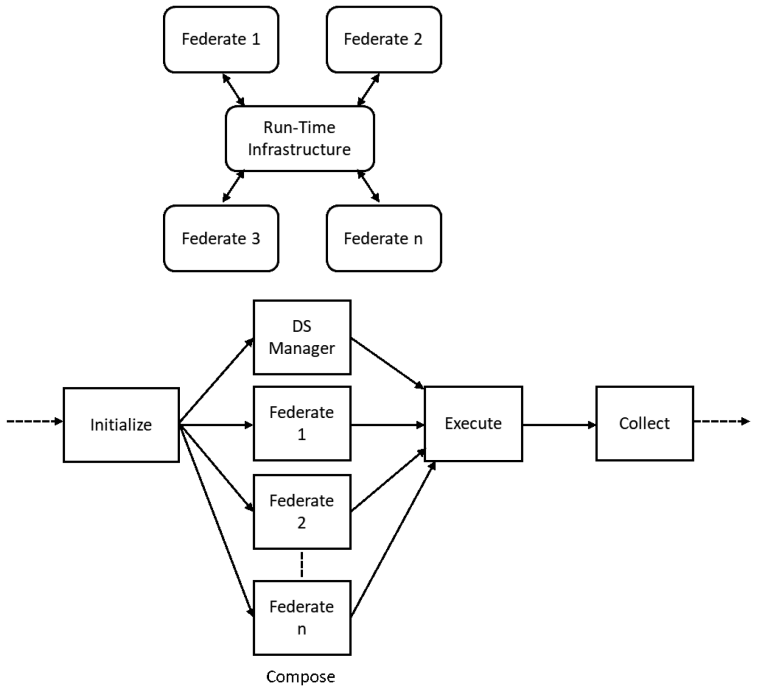

Fig. 3. A federation and possible distributed simulation workflow.

to execute and return the composed *CRCComp* software to the DSS. The DSS informs the WMS with the message *Ready*. The Federates are composed in a similar manner with the SDS instead of the DSS. When the process has finished, the WMS will return *Done* to note the end of the task.

Execute: The WMS first instructs the DSS to start the CRC with *ExecCRC*. The DSS sends the composed CRC software with the instruction to deploy and start the CRC to the JSS via *CRCCompExec*. The JSS then manages the deployment and execution of this on Cloud (*CRCRun*). The WMS then instructs the SDS to send each Federate software for deployment and execution (*ExecFed*). The SDS sends the composed federates to the JSS with their own instruction to deploy and run to the JSS (*FedExec*). The JSS also then manages the deployment and execution of these on Cloud. The Federate software when executed will register with the CRC, synchronize (*Fed*Synch*) and begin execution (*DSRun*). When complete (*DSTermination*), the CRC and Federates will return *DSTermination* to the DSS to note the end of the DS.

Results: Once the Federates have finished their simulations they return their results back to the WMS via the JSS. The JSS can then decommission cloud resources, and so on.

It should be noted that the above could be represented as a general WMS architecture with a management layer containing a dedicated distributed simulation service. However, we wanted to present our conceptualization to emphasize the use of contemporary workflow technologies in the context of distributed simulation. This is similar to previous works that have taken existing architectures and applied them to distributed simulation (e.g., References [15, 19, 23]). We have also presented our architecture with general features common to virtually all WMS. All WMS have some method to create, configure, and submit workflows, either by a GUI (i.e., a science gateway) or by some dedicated WMS language. Workflow management, job submission, and software deployment services are common to all. Only WS-PGRADE [58] has a dedicated parametric study

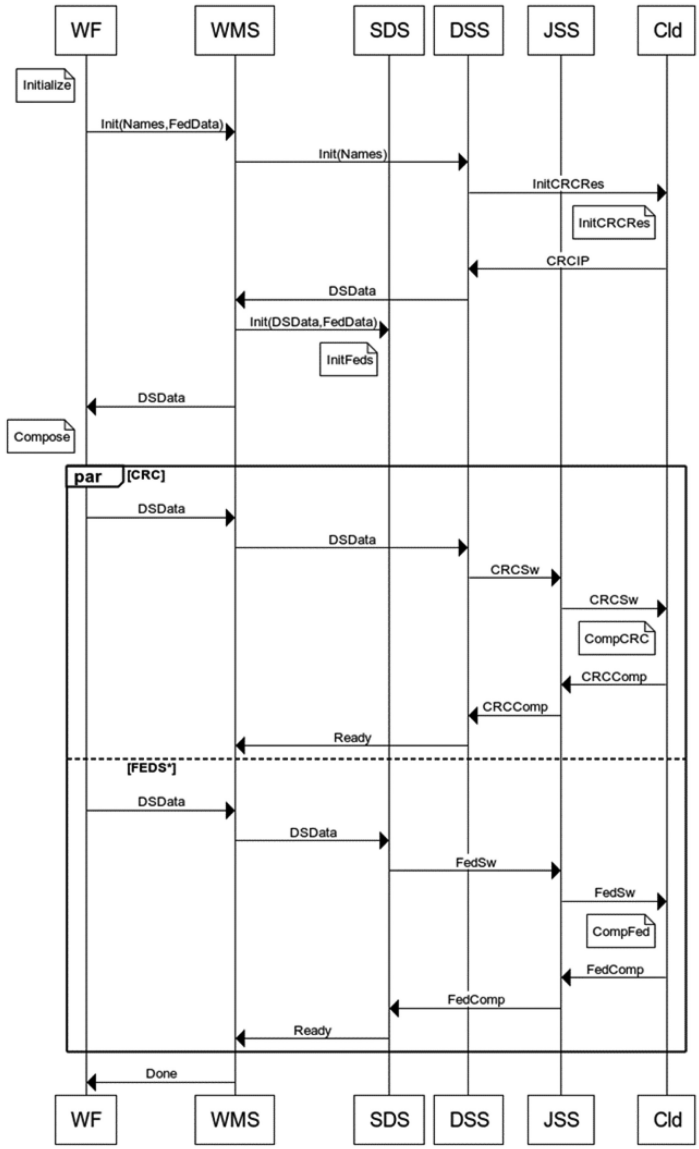

Fig. 4. WORLDS architecture sequence diagram (part 1).

service—other WMS rely on external experimentation management to perform parameter sweep experimentation. It appears that no dedicated DS Service has been created for any WMS to date.

Comparing our architecture to other works in CBDS, References [27] and [29] (extended in Reference [30]) propose comprehensive architectures with services similar to ours in terms of distributed simulation experimentation management, job submission to cloud, job management on cloud, and cloud resource management. However, neither use any form of workflow or indeed the ability to choose which cloud or cloud resources to use. References [31–35] use variants of manager-worker schemes where distributed simulations are submitted to a manager for distribution to cloud workers mapped onto one or more cloud instances. These are essentially the same as

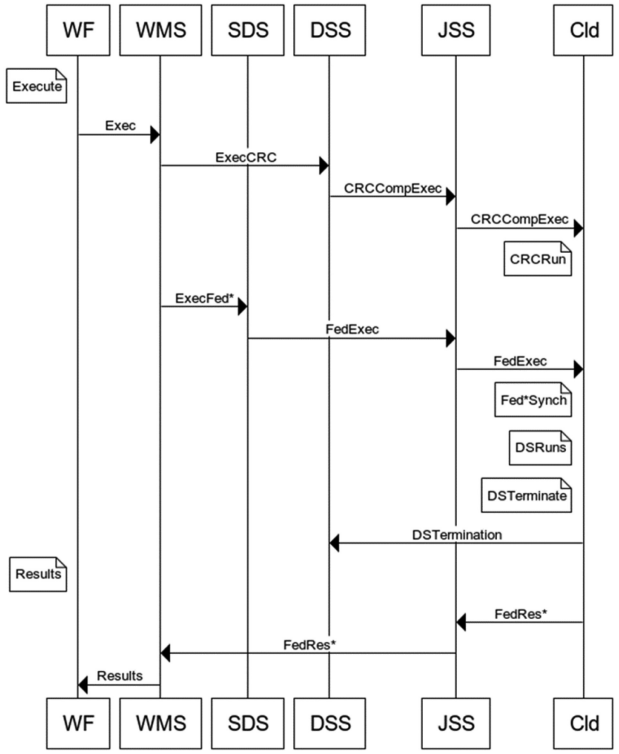

Fig. 5. WORLDS architecture sequence diagram (part 2).

the job submission/management elements of the CBDS architecture of ours and the above works. All examples of CBDS use either single cloud (or cluster-based) instances with multiple CPUs or multiple single instances with a single CPU. We could not identify a similar CBDS approach to ours that facilitates distributed simulation deployment using different cloud resource schemes via workflow.

The next section discusses how WORLDS can be implemented using an existing WMS/e-Infrastructure/Cloud technology stack.

4 WORLDS PROOF OF CONCEPT

To give a proof of concept of how the WORLDS architecture could be implemented, we needed an e-Infrastructure that could access cloud as a DCI and a WMS, ideally one with some form of Science Gateway. As discussed in the previous section, there are several candidate systems. WS-PGRADE/gUSE is a well-known WMS [64]. Previous research had investigated how multi-cloud functionality could be added to WS-PGRADE/gUSE via the integration of the CloudBroker Platform [65]. This work was significantly extended for commercial applications to create the **CloudSME Simulation Platform (CSSP)** [63]. This has been used by over 30 European SMEs to develop cloud-based simulation applications and is a core technology in several large European projects. It has also been used to develop web-based cloud portals for agent-based simulation (REPAST) and discrete-event simulation (JaamSIM) [66]. We chose the CSSP as the vehicle to demonstrate WORLDS partly due to the platform having all the components we needed but also our familiarity with it. We now briefly describe the relevant aspects of the CSSP.

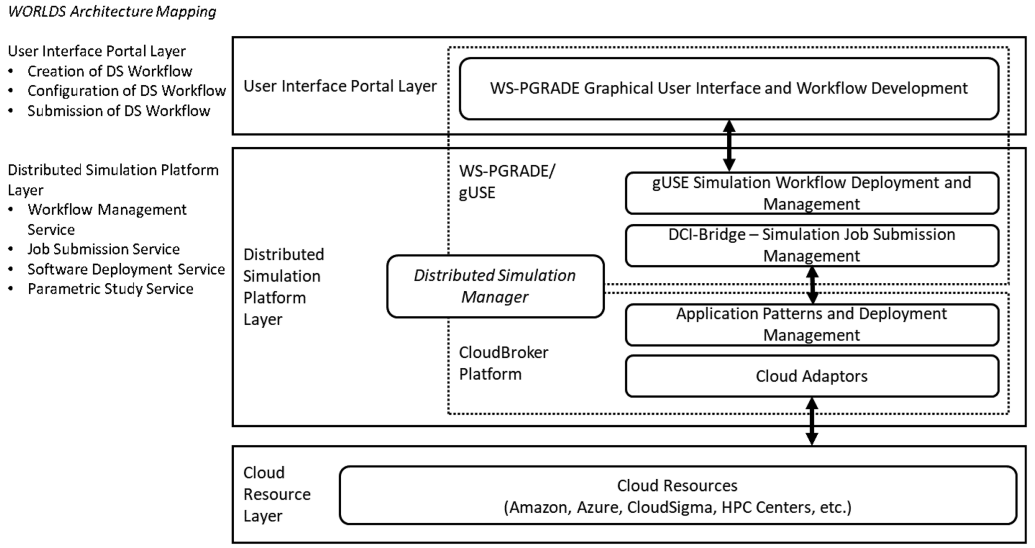

Fig. 6. WORLDS demonstration with the CloudSME simulation platform.

The CSSP is shown in Figure 6 with its mapping to WORLDS. WS-PGRADE/gUSE consists of three tiers: a presentation tier consisting of WS-PGRADE that enables users to create and populate workflows using a graphical editor for applications to run on DCIs, a middle tier of gUSE services to save, deploy, and manage workflows, and an architectural tier that uses the DCI Bridge job submission service and allows gUSE to submit and monitor jobs on a DCI. WS-PGRADE has a graphical editor that can be used to build workflows and specify job configurations using typical directed acyclic graph semantics. gUSE has services to support workflow management and execution (e.g., file storage, workflow repository, and workflow interpreter to manage the execution of workflows). The DCI Bridge has various plug-ins to manage job submission, as specified in the workflow using the OGSA Basic Execution Service interface. Workflows are created in WS-PGRADE and managed in gUSE. gUSE creates jobs that are submitted and executed on a DCI through the DCI Bridge. The CloudBroker Platform is a PaaS that allows developers to implement, manage, and bill applications on different clouds. It is a web-based store for applications that deploy and execute compute-intensive applications on a cloud. It is suitable for any kind of batch-oriented command line software, both Linux- and Windows-based, and either serial or parallel processing. It provides several development options, including web browser deployment and/or various APIs (REST, Java, etc.). Access to different clouds is provided by various cloud adaptors. CloudBroker stores SaaS application patterns (typically application code and any associated installation/launch scripts), instantiates these on demand and launches/manages these through the cloud adaptors on different cloud resources. CloudBroker also deals with fault-tolerance, errors and exception handling through its Scalability and Fault Tolerance Handler module. The CSSP also has an App Center—this was not used in this work, as there was no requirement for billing and user management over and above the services available in WS-PGRADE.

The CSSP has been used to create SaaS M&S applications. To run on cloud, the executable part of the application needs to be able to run headless and take command line parameters, as it needs to be remotely executed. This is stored as an application pattern with deployment and runtime scripts on CloudBroker. Specific implementations need to be created for single/multi-core cloud instances. A workflow is then created/reused in WS-PGRADE. Versions of the workflow are then parameterized

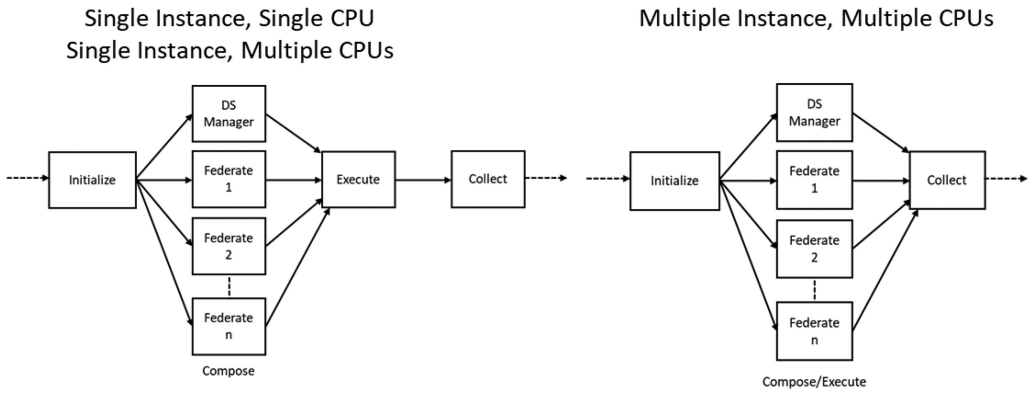

Fig. 7. Demonstration workflows.

for the different simulation application patterns and different clouds/instances. The workflow can be executed directly or called by the simulation application deployed as a desktop or web-based application via the WS-PGRADE REST API. When executed, WS-PGRADE calls gUSE to run the different tasks of the workflow. gUSE instructs CloudBroker via the DCI Bridge to instantiate the relevant application pattern for each task. CloudBroker then deploys the application to cloud via the appropriate Cloud Adaptor to whatever cloud and instance type have been selected and then runs it. The CloudBroker Platform and WS-PGRADE/gUSE then monitor the execution of the application, coordinate the return of the results and the transfer of data/files between the tasks until the workflow is complete.

The Workflow Management Service maps to WS-PGRADE, the Job Submission Service to gUSE and the Software Deployment Service to CloudBroker. The DS Manager and Federate application patterns are first stored on CloudBroker for each cloud and instance type needed by the end-user. These are the deployments of the RTI CRC and supporting application software, respectively (see Section 5 for further details). When called, CloudBroker would request a specific cloud/resource type via a cloud-specific adaptor. It would then deploy these images via this cloud adaptor to the resource and initialize them. WS-PGRADE supports the Parametric Study Service by providing dedicated workflow tasks (generator/collector). The Distributed Simulation Service is split between WS-PGRADE and CloudBroker. In WS-PGRADE this would be ideally implemented in a similar manner to the Parametric Study Service. To demonstrate how WORLDS could be realized, the DS Manager was split into two parts. The DS Manager patterns were stored on CloudBroker as noted above. Prior to running the workflow, CloudBroker would be used manually to boot the specific cloud resource so the external IP address could be obtained. This is then manually entered in the workflow. The specific workflow DS Manager task would still instruct CloudBroker to deploy the CRC software on the dedicated cloud resource.

As part of this demonstration, as shown in Figure 7, three workflows were developed to show how different cloud resource types could be used. These are:

- **Single Instance, Single CPU (SISC)** – cloud resource with one CPU (the federation runs on a single core instance);
- **Single Instance, Multiple CPU (SIMC)** – cloud resource with many CPUs (all federates run on their own CPUs within a single instance);
- **Multiple Instances, Single CPU (MISC)** – multiple cloud resources with one CPU each (the federates run on separate single core instances and communicate via a TCP/IP network on cloud).

The SI workflows require four stages: initialize, compose, execute, and collect. The initialize task takes as input the list of federates in the distributed simulation, validates it, and passes it to the next task. This task could do more, depending on the application (such as updating probability distributions, lookup tables, and so on, used in the simulation). There is one compose task for each federate and one for the DS Manager. These bring together the model files, federate names, supporting software/scripts, data, and so on, for each federate. The DS Manager task brings together the RTI software, federate names, and any other supporting software/scripts. Each task outputs a zipped file to the execute task. The execute task unzips each file (DS Manager first and then in the order of the federate names) and runs each associated process (the RTI CRC and then each federate in order). The SISC execute task would do this on a single CPU. The SIMC execute task would require some method of allocating the DS Manager and the federates to specific CPUs. When the DS Manager runs, a synchronization point labelled "ReadyToRun" is registered with the Register Federation Synchronization Point service. As a federate registers with the DS Manager, it invokes the RTI Announce Synchronization Point service to inform the federate of a new synchronization point "ReadyToRun." Federates then set a corresponding synchronization point and inform the DS Manager through the RTI Synchronization Point Achieved service. Once the DS Manager records that all federates have invoked this service, it then invokes the Synchronization Point "ReadyToRun" Achieved service and the federates will begin to execute their models. Once execution has finished, the federates zip their results to be passed on to the final task. Collect receives all result zips, brings them together, and outputs the final results. This task could also do some form of analysis of the results.

The MISC workflow is similar but does not have the execute task, as its function is merged with the distributed compose tasks. The IP address of the CRC is added to the list of inputs of the initialize task to be passed on each compose task. These prepare the zip file for execution with a different scriptfile to the SI workflows. This script actually runs the DS Manager and federates once they have been booted on their cloud instance. The DS Manager would be done first on the specified IP address followed by each federate (which has the IP address of the DS Manager). A **multiple instances, multiple CPUs (MIMC)** workflow is possible, if a "hierarchical" mapping of federates to instances and CPUs is needed, by combining the MISC and SIMC workflows.

5 EXPERIMENTS AND RESULTS

5.1 Motivation for the Experiments and Case Study

To demonstrate the CSSP implementation of the WORLDS architecture, we used the agent-based **Emergency Medical Service (EMS)** distributed simulation developed in previous work [67, 68]. The motivation behind the case study selection was the scaling flexibility of the EMS in terms of federation size. We ran experiments for all workflow configurations to investigate the performance and scalability. This section briefly describes the EMS and the associated workflows for SISC, SIMC, and MISC. Results are then presented from running these on the Amazon EC2 cloud.

5.2 Case Study

Figure 8 shows the concept of the EMS distributed simulation. It was originally developed to show how distributed simulation could be used to develop large-scale models of city-wide emergency services. The federation consists of one ambulance model and several **Accident & Emergency (A&E)** (Emergency Room) models. The EMS was implemented using REPAST [20] and the Portico RTI [69]. The distributed simulation advances time at regular intervals. Essentially, at each "tick" each A&E federate updates its patient capacity and sends an interaction to the Ambulance federate to specify the number of patients it can take. The Ambulance federate updates the current state

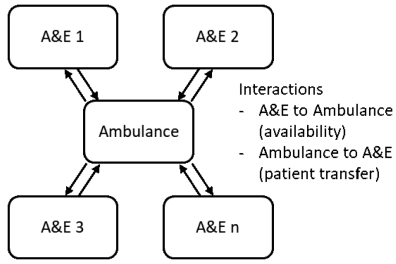

Fig. 8. Agent-based Emergency Medical Service distributed simulation [67, 68].

Table 1. Configuration of Amazon EC2 Cloud Resources for Each Workflow Task

Task	Workflow	CPU Specification	CPUs	RAM
Initialize	SISC, SIMC, MISC	Intel(R) Xeon(R) E5-2650 2.00 GHz	1	1 GB
DS Manager	SISC, SIMC (compose)	Intel(R) Xeon(R) E5-2650 2.00 GHz	1	1 GB
	MISC (compose and execute)	Intel(R) Xeon(R) E5-2650 2.00 GHz	1	2 GB
EMS Federates	SISC, SIMC (compose)	Intel(R) Xeon(R) E5-2650 2.00 GHz	1	1 GB
	MISC (compose and execute)	Intel(R) Xeon(R) E5-2650 2.00 GHz	1	2 GB
Execute	SISC	Intel(R) Xeon(R) E5-2650 2.00 GHz	1	2 GB
	SIMC	Intel(R) Xeon(R) E5-2666 V3 2.9GHz	2	3.75 GB
		Intel(R) Xeon(R) E5-2666 V3 2.9GHz	4	7.5 GB
		Intel(R) Xeon(R) E5-2666 V3 2.9GHz	8	15 GB
		Intel(R) Xeon(R) E5-2666 V3 2.9GHz	16	30 GB
Collect	SISC, SIMC, MISC	Intel(R) Xeon(R) E5-2650 2.00 GHz	1	1GB

of the emergency, the demand for ambulances, and the current ambulance loading. On receipt of the patient number interactions, it decides where to place patients. Interactions are then sent to the A&E federates to update their patient numbers (i.e., the patients have arrived at A&E) and the details of those patients (currently one patient per interaction). These are derived from the hourly emergency call distribution and the percentage of these requiring ambulance services. In this case study, time in the EMS was fixed to advance hourly, one interaction is sent by each A&E federate to the Ambulance federate, and approximately 14 interactions are sent from the ambulance to the A&E federates (random distribution).

5.3 Experimental Setup

Two types of experiments were carried out: the first to show how each federation performed over time and the second to show the scalability when increasing the number of federates. The aim of both is to show indicative results and should be considered as the first steps in a larger study. All experiments were repeated five times, and we use the average of these runs in our results.

We used the Amazon EC2 cloud. The configuration of the resources that we used in our experiments are shown in Table 1. All cloud instances installed the 64-bit Ubuntu Server 16.04 LTS Operating System. Initialize and Collect tasks run on a single CPU with 1 GB RAM in all three workflows. In all SI workflows, the Compose tasks ran on a single CPU with 1 GB RAM. The Execute task requires a larger instance in the SI workflows, as it has to run the federation and requires adequate memory and compute resources. We therefore ran the SIMC on different instance types with 2, 4, 8, and 16 CPUs with 3.75, 7.5, 15, and 30 GB RAM, respectively (i.e., the DS Manager and

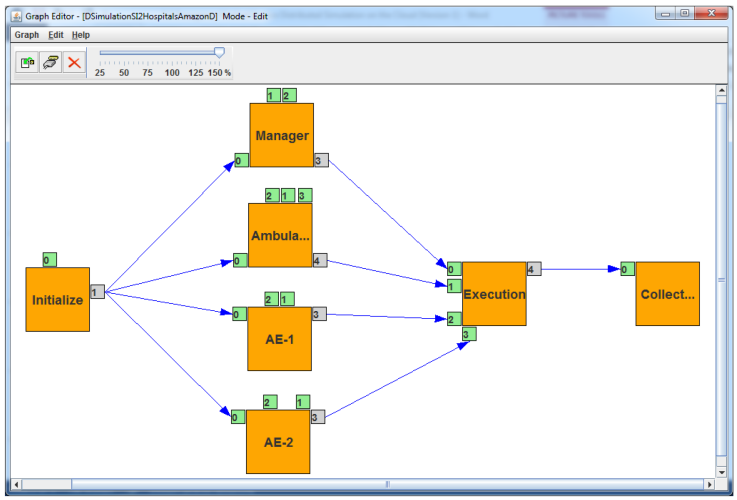

Fig. 9. Workflow for SISC and SIMC.

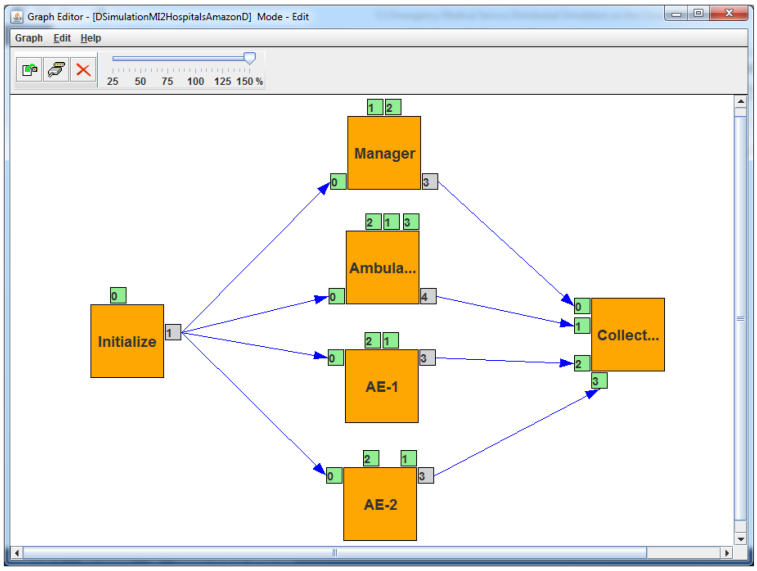

Fig. 10. Workflow for MISC.

the EMS federates are allocated to specific CPUs). In MISC, Compose is merged with Execute. The DS Manager and each of the EMS federates in the MISC workflow run on instances with 1 CPU with 2 GB RAM. The SISC and SIMC workflows are shown in Figure 9 and the MISC in Figure 10 as they appear in the WS-PGRADE graphical editor. The amber squares indicate the workflow nodes. Attached to the nodes are the input ports (small green squares) and the output ports (small gray squares). The arrows indicate the dataflow between nodes. Table 2 shows the inputs, outputs, and scripts used in our workflows. For example, the Initialize node in the SI workflows takes as input a text file that lists all federates (0: federateslist.txt in Table 2). The file is transferred via the Initialize output port (1: federateslist.txt in Table 2) to the Manager node's input port (0: federateslist.txt in

Table 2. Workflow Configuration of Distributed Hybrid ABS-DES EMS

Job Node	Configuration Parameters	SI Workflows	MI Workflow
Initialize	Software	initialize.sh	initialize.sh
	Instance Type	See Table 1	See Table 1
	Argument String		IP address of the CRC
	Input Ports	0: federateslist.txt (end-user uploads)	
	Output Ports	1: federateslist.txt	Output-port 0: CRCIP.txt
Manager, Ambulance, AE-1, AE-2, etc.	Software	manager.sh for DS Manager model.sh for Ambulance, AE-1, AE-2, etc.	manager.sh for DS Manager model.sh for Ambulance, AE-1, AE-2, etc.
	Instance Type	See Table 1	See Table 1
	Argument String	Name of the federate, i.e., Ambulance for Ambulance job node, AE1 for AE-1 job node, and AE2 for AE-2 job node	
	Input Ports	0: federateslist.txt 1: model.tar 2: batch_params.xml 3: data.tar (Only in Ambulance)	0: federateslist.txt 1: model.tar 2: batch_params.xml 3: data.tar (Only in Ambulance)
	Output Ports	3: fManager.tar for the Manager 3: AE1.tar for the AE-1, AE2.tar for the AE-2, etc. 4: Ambulance.tar for Ambulance	3: fManager.tar for the Manager 3: AE1.tar for the AE-1, and AE2.tar for the AE-2, etc. 4: Ambulance.tar for Ambulance
Execute	Software	repastmulti.sh (to map federates on to different CPUs)	Does not run on MI
	Instance Type	See Table 1	
	Input Ports	0: fManager.tar 1: Ambulance.tar 2: AE1.tar 3: AE2.tar	
	Output Ports	4: output.tar	
Collect Results	Software	results.sh	results.sh
	Instance Type	See Table 1	See Table 1
	Input Ports	0: output.tar	0: fManager.tar 1: Ambulance.tar 2: AE1.tar 3: AE2.tar

Table 2). The Initialize node when executes runs the initialize.sh script. The SI and MI workflows and their dependencies are openly available.[1]

5.4 Performance Metrics and Results

5.4.1 Performance. The first set of experiments investigated the performance of the federation and the cloud overhead in relation to the total cloud runtime. These two metrics are used to calculate the speedup and the speedup/cost ratio of the examined workflow configurations. In

[1] https://doi.org/10.17633/rd.brunel.14853666.

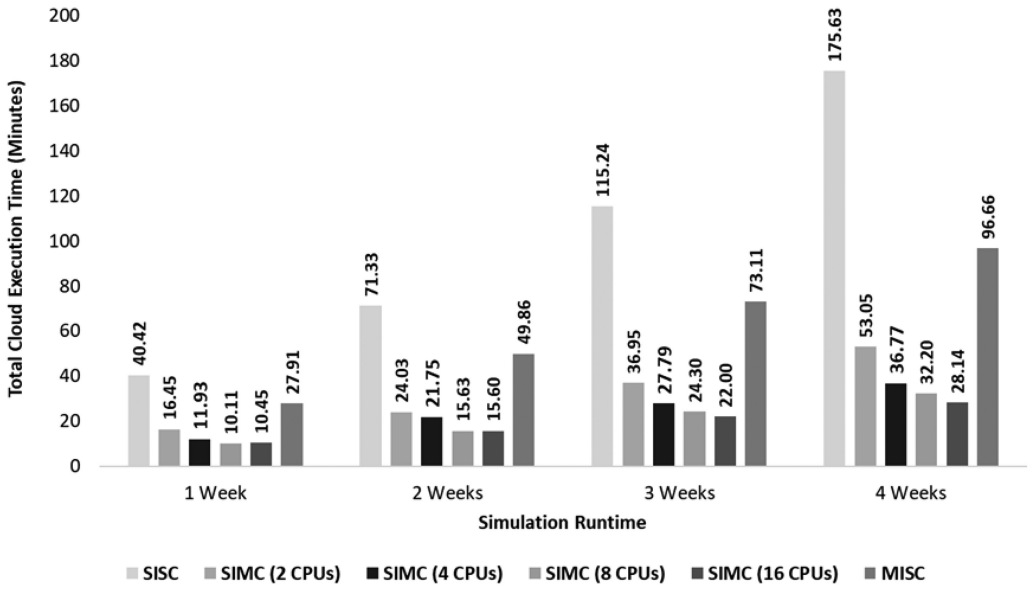

Fig. 11. Total average cloud execution time by workflow/experiment.

these experiments the federation was fixed at 15 federates—1 ambulance federate and 14 A&E federates.

5.4.1.1 Total Cloud Execution Time and Speedup. To examine how the federation ran over time, four experiments were performed—the federation was run separately for one to four weeks of simulation time for each of the experimental configurations. Figure 11 shows the average total workflow execution time on the cloud for all 24 experiments. Taking as point of reference the SISC workflow, the speedup is calculated for all workflows. Figure 12 depicts the speedup gained when running the federation on more than one CPU and for the four examined simulation times. The SISC workflow is used as reference and therefore the speedup is one. As expected, the execution time increases as the simulation runtime increases. However, the increase in SISC and MISC is significantly larger, while we observe a very small increase in execution time for all SIMC workflows. This behavior is expected, since both SISC and MISC run on instances with 2 GB RAM while SIMC experiments use larger single instances. This demand on resources comes from the number of patients in the simulation—in the first few weeks of the emergency the number of patients grows and the details of each patient need to be stored in the system and transferred via interactions. The performance gets better as larger instances are used. In SISC, all 15 federates run on a single instance, and the single instance has to handle the processing of the entire federation. This performs worse than the other experiments. In MISC, each federate runs on the same instance type as SISC. Performance arguably reflects the balance of dividing the processing/memory load of the simulation across different instances and the communication overhead of the interactions passed to and from the RTI. Similar behavior is observed in the speedup, as shown in Figure 12. Generally, SIMC workflows perform better. The highest speedup is observed in the SIMC (16 CPUs) workflow when the simulation runs for four weeks. However, for one-week simulation time, SIMC (4 CPUs) and SIMC (8 CPUs) outperform the largest instance. Interestingly, we observe a drop in speedup in the SIMC (2 CPUs) workflow when the simulation runs for two weeks. This behavior is due to the high variation in the cloud resources setup time for the specific experiment. It is worth mentioning that

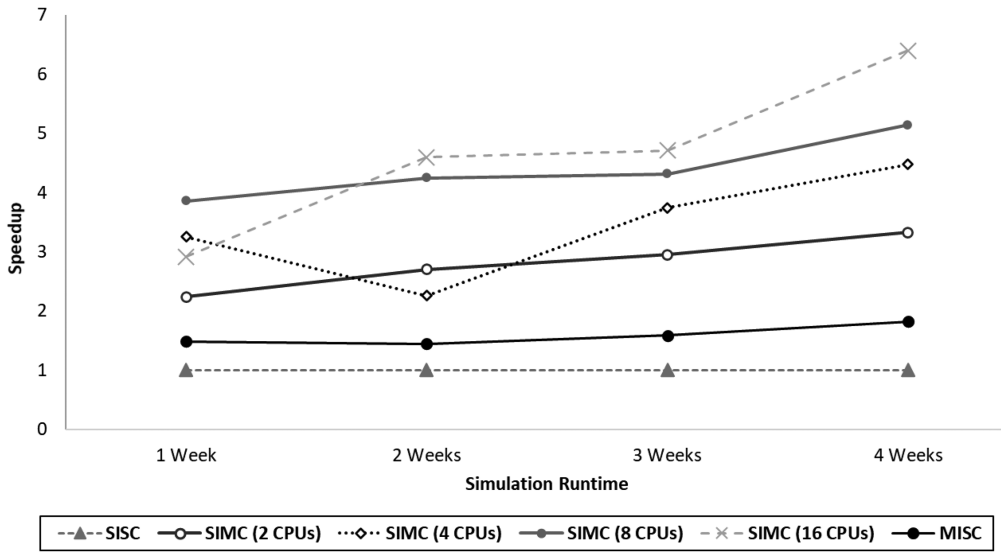

Fig. 12. Speedup by workflow experiment.

the cloud resources used were not dedicated and therefore it was expected to see wide variability in cloud setup times. Nonetheless, the standard deviation of all simulation runtimes is small and shows that once the cloud resources are created the federation runs consistently on this cloud. The data that the performance graphs are based on is openly available.[2]

5.4.1.2 Cloud Overhead and Speedup-cost Relationship. To examine the extra computing cost of the different workflow configurations, we measured the cloud overhead that represents the setup time required to request and commission cloud resources. Figure 13 shows the proportion of runtime split into cloud overhead and workflow execution time. The figures were obtained through the CSSP's monitoring system. These are shown again for the ongoing execution of the federation. As can be seen, as a proportion of total execution time, the cloud overhead for SISC is lowest followed by MISC and then the various SIMC configurations. This is expected, since the cloud overhead as an absolute value is similar for all workflows. However, the actual simulation execution time in the single CPU instances, both in SI and MI, is longer than that of the multi-CPU instances and therefore the cloud overhead is more significant. All multi-CPU instances have more RAM than the single CPU instances, which explains why we observe better simulation performance. MISC takes longer than SISC, which is because this workflow requires different cloud instances to be set up. This behavior reflects the general behavior of shared tenancy cloud resources provisioning in which the spin up time can vary, depending on general demand. As the simulation progresses, the relative proportion of cloud overhead reduces. This would be expected from the "one-off" cost of setting up the instances against computation time. Overall, it does appear in this limited investigation that the largest multi-CPU instance used in the SIMC workflow produces the best performance in terms of cloud overhead proportion. To understand the effectiveness of the studied workflow configurations, we examined the speedup cost relationship, too. Table 3 shows the speedup/cloud overhead ratio. This analysis shows that the cost-effectiveness differs for the examined simulation times. For example, the SIMC (eight CPUs) is the most cost-effective configuration for one week simulation time, while the SIMC (four CPUs) is the most cost-effective

[2]https://doi.org/10.17633/rd.brunel.14852466.

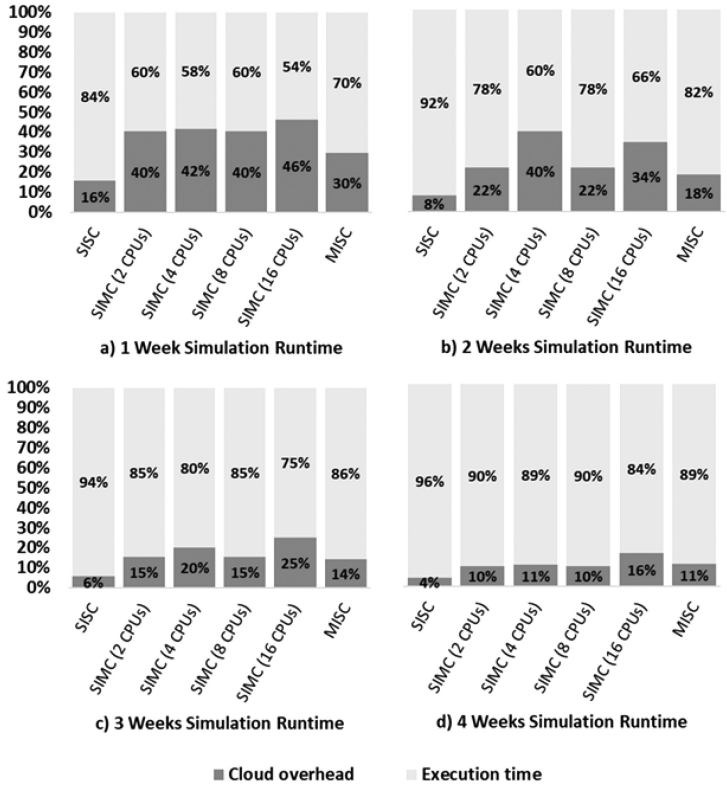

Fig. 13. Average relative cloud overhead by workflow/experiment.

Table 3. Speedup/Cloud Overhead Ratio by Workflow

Workflow	1 Week	2 Weeks	3 Weeks	4 Weeks
SISC	6.37	12.71	17.89	22.81
SIMC (2 CPUs)	5.56	12.53	19.19	33.42
SIMC (4 CPUs)	7.83	5.67	18.74	42.22
SIMC (8 CPUs)	9.63	14.72	17.99	28.51
SIMC (16 CPUs)	6.29	13.44	18.69	38.88
MISC	5.04	7.88	11.26	16.53

configuration for four weeks' simulation time. This leads to the conclusion that instance type selection should be carefully considered for different federation sizes and experimentation setup. It would be interesting to see if better performance could be obtained with a larger instance type used in the MISC workflow as well as partitioning the simulation to use larger instances a MIMC workflow (e.g., four instances running four federates).

5.4.2 Scalability. The second set of experiments investigated the scalability of the federation. For these experiments, we used federations of different "size" and ran them on all six workflow configurations. Federation size ranged from 3 to 15 federates (i.e., 1 ambulance with 2, 4, 6, 8, 10, 12, 14 A&E federates). The interactions remained the same but were equally split between A&Es (e.g., 7/7 per hour for the 1+2 federation). All 42 experiments ran for four weeks' EMS simulation time.

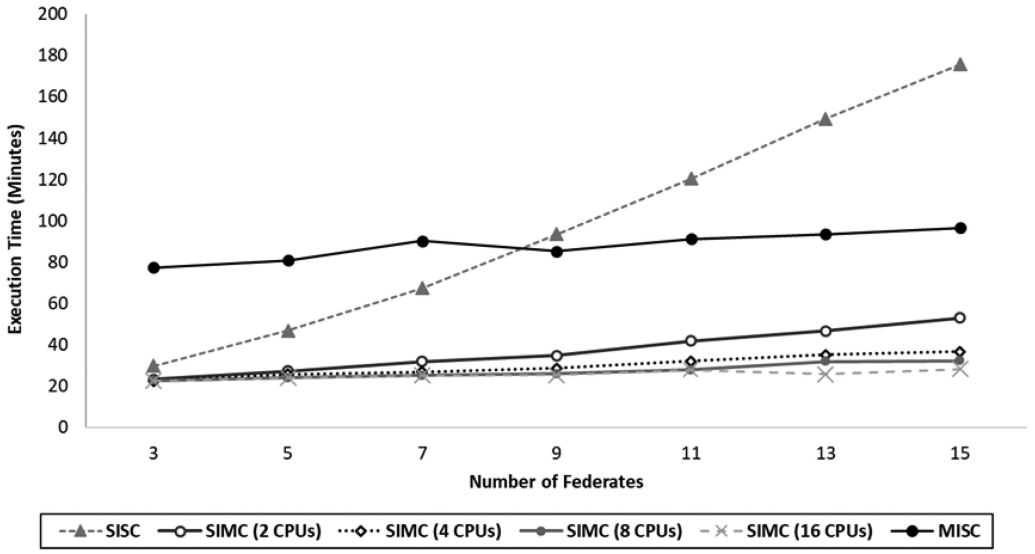

Fig. 14. Execution time by federation size (1 ambulance + 2n A&E federates)/workflow/configuration.

Figure 14 shows the average execution time of all experiments. The execution time for the SISC workflow increases as the number of federates increases. All SIMC experiments perform better than the others with the two-CPU configuration showing a small decrease in performance with larger workloads. There is virtually no decrease with the other SIMC configurations. This suggests that the four-CPU configuration is the best for this simulation. MISC is similarly unaffected by larger workloads and, as noted above, is probably due to sharing the processing/memory load for the simulation. The communication overhead appears to have little effect. The standard deviation again is small for all experiments. This may give different results if the number of patients increases significantly (again to be investigated as part of a larger study). Overall, it appears that using single instances with larger resources gives the best performance for these experiments up to a point. Given that cloud instances vary in cost, this highlights the need to profile federation performance to give the best performance/cost ratio for a particular cloud. This is especially true if the distributed simulation will be run many times as part of a simulation study. The data that the scalability graphs are based on is openly available.[3]

6 CONCLUSIONS

This article has investigated a novel approach to CBDS that has attempted to simplify the complexity of using distributed simulation on cloud. It has reviewed contemporary approaches to CBDS and identified the limitations of these approaches. E-Infrastructures/WMS were then introduced and the general approach used to run applications on cloud using workflows. The WORLDS architecture was then presented that essentially added a distributed simulation service to e-Infrastructure/WMS. Different workflows reflecting different deployment strategies were then presented to illustrate how WORLDS might be used. A possible implementation of WORLDS was then discussed using the CloudSME Simulation Platform. A case study demonstrated how this implementation could be used with the different workflows using the Amazon EC2 cloud and the distributed Emergency Medical Service simulation.

[3]https://doi.org/10.17633/rd.brunel.14852580.

In this article, we have asked two questions: Could these technologies be used to create DS applications and to what extent does this approach simplified CBDS? We have shown how DS can be successfully implemented by adding a DS Service to a generalized WMS/e-Infrastructure architecture and have then demonstrated this using one example of this technology stack, the CloudSME Simulation Platform. As noted below, further work needs to be carried out to determine if this approach is generalizable to other WMS technologies. However, as we have attempted to start from a general WMS architecture, we anticipate that our approach will be translatable.

In terms of simplifying CBDS, the main issue with previous work is that while most approaches have advanced understanding of what is needed for CBDS, there is little evidence of their continued use. GridSpice and SEMSim are exceptions to this, as they have been used to study problems in their application domains. These do simplify the use of CBDS in their areas. It may be that both are generalizable to other applications. Our work has added to these examples by demonstrating that e-Infrastructures/WMS can be used for CBDS. This is attractive, as these technology stacks are supported by large worldwide communities rather than a single research project. Our demonstration is limited to one example of this technology. However, it does seem reasonable to assume that our WORLDS architecture and workflow could be implemented using other e-Infrastructures/WMS. It must be noted that in our approach distributed simulations still need to be created along with deployment packages for cloud installation. However, once this has been done, workflows can be used to automate deployment and e-Infrastructures, such as the CloudSME Platform, can simply cloud implementation, especially with different clouds.

Our case study demonstrated how CBDS could be deployed using different workflows and different cloud resources. One result of this showed that using larger, more costly, single instances does not necessarily mean faster simulations. Work will continue to investigate how federates or LPs can be profiled to ensure the optimal selection of resource. We are also interested in multiple, parallel DS experimentation, as would be done using a Parameter Study Service. Studies have been done using an auto-scaling approach on cloud with sequential simulations [12]. Extending this to investigate how CBDS could auto-scale distributed processing during parameter studies experimentation would be desirable, as it could optimize cloud use and reduce costs. Another issue is how CBDS could be run over different clouds. If simulations lie under different national or commercial data protection authorities, then it may be that federates have to run on specific clouds (e.g., within national boundaries). It would be interesting to see the performance impact of federate mapping onto sets of resources from different clouds. This would certainly be possible using CloudBroker. Finally, we are also interested in how workflows could be integrated with proposed extensions to the HLA **Distributed Simulation Engineering and Execution Process (DSEEP)** [68].

ACKNOWLEDGMENTS

The sequence diagrams are created in www.websequencediagrams.com.

REFERENCES

[1] Richard M. Fujimoto. 1999. Parallel and distributed simulation. In *Proceedings of the Winter Simulation Conference*. IEEE Press, 122–131.

[2] Richard M. Fujimoto. 2000. *Parallel and Distributed Simulation Systems*. John Wiley, New York, NY.

[3] Peter M. Mell and Timothy Grance. 2011. The NIST definition of cloud computing. *NIST Special Publication 800-145*. Gaithersburg, MD. DOI: https://doi.org/10.6028/NIST.SP.800-145

[4] Ji Liu, Esther Pacitti, Patrick Valduriez, and Marta Mattoso. 2015. A survey of data-intensive scientific workflow management. *J. Grid Comput.* 13, 4 (2015), 457–493. DOI: https://doi.org/10.1007/s10723-015-9329-8

[5] Richard M. Fujimoto. 2016. Research challenges in parallel and distributed simulation. *ACM Trans. Model. Comput. Simul.* 26, 4 (2016), 1–29. DOI: https://doi.org/10.1145/2866577

[6] Simon J. E. Taylor, Richard Fujimoto, Ernest H. Page, Paul A. Fishwick, Adelinde M. Uhrmacher, and Gabriel Wainer. 2012. Panel on grand challenges for modeling and simulation. In *Proceedings of the Winter Simulation Conference*. IEEE Press, 1–15.

[7] IEEE-1278. 1993. IEEE standard for information technology–Protocols for distributed interactive simulations applications. *Entity Information and Interaction*. IEEE Std 1278-1993 (1993). DOI: https://doi.org/10.1109/IEEESTD.1993.115125

[8] IEEE-1516-2000. 2000. IEEE Standard for Modeling and Simulation (M S) High Level Architecture (HLA)–Framework and Rules. IEEE Std 1516-2000 (2000), 1–28. DOI: https://doi.org/10.1109/IEEESTD.2000.92296

[9] IEEE-1516. 2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules. IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000) (2010), 1–38. DOI: https://doi.org/10.1109/IEEESTD.2010.5553440

[10] Simon J. E. Taylor. 2019. Distributed simulation: State-of-the-art and potential for operational research. *Eur. J. Oper. Res.* 273, 1 (2019), 1–19. DOI: https://doi.org/10.1016/j.ejor.2018.04.032

[11] Richard Hill, Laurie Hirsch, Peter Lake, and Siavash Moshiri. 2013. *Guide to Cloud Computing: Principles and Practice (1st.* ed.) Springer-Verlag London. DOI: https://doi.org/10.1007/978-1-4471-4603-2

[12] Tamas Kiss, James DesLauriers, Gregoire Gesmier, Gabor Terstyanszky, Gabriele Pierantoni, Jozsef Abu Oun, Simon J. E. Taylor, Anastasia Anagnostou, and Joszef Kovacs. 2019. A cloud-agnostic queuing system to support the implementation of deadline-based application execution policies. *Fut. Gen. Comput. Syst.* 101 (2019), 99–111. DOI: https://doi.org/10.1016/j.future.2019.05.062

[13] Yucong Duan, Guohua Fu, Nianjun Zhou, Xiaobing Sun, Nanjangud C. Narendra, and Bo Hu. 2015. Everything as a Service (XaaS) on the cloud: Origins, current and future trends. In *Proceedings of the 8th IEEE International Conference on Cloud Computing*. IEEE, 621–628.

[14] Ian Foster, Carl Kesselman, and Steven Tuecke. 2001. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Applic.* 15, 3 (2001), 200–222. DOI: https://doi.org/10.1177/109434200101500302

[15] Katherine L. Morse, David Drake, and Ryan Brunton. 2003. Web enabling an RTI—An XMSF profile. In *Proceedings of the European Simulation Interoperability Workshop*. SISO. E03-SIW-034.

[16] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. 2002. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.* Open Grid Services Infrastructure Working Group. Globus Project.

[17] J. Mark Pullen, Ryan Brunton, Don Brutzman, David Drake, Michael Hieb, Katherine L. Morse, and Andreas Tolk. 2004. Using web services to integrate heterogeneous simulations in a grid environment. *Lecture Notes in Computer Science*, Vol. 3038, 835–847. DOI: https: //doi.org/10.1007/978-3-540-24688-6_108

[18] Yong Xie, Yong Meng Teo, Wentong Cai, and Stephen J. Turner. 2005. Service provisioning for HLA-based distributed simulation on the grid. In *Proceedings of the IEEE Workshop on Principles of Advanced and Distributed Simulation*. IEEE Press, 282–291.

[19] Georgios Theodoropoulos, Yi Zhang, Dan Chen, Rob Minson, Stephen J. Turner, Wentong Cai, Yong Xie, and Brian Logan. 2006. Large scale distributed simulation on the grid. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid*. IEEE Press, 63–63.

[20] Michael J. North, Nicholson T. Collier, Jonathan Ozik, Eric. R. Tatara, Charles M. Macal, Mark Bragen, and Pam Sydelko. 2013. Complex adaptive systems modeling with Repast Simphony. *Complex Adapt. Syst. Model.* 1, 3 (2013). Retrieved from http://dx.doi.org/10.1186/2194-3206-1-3

[21] Xinjun Chen, Wentong Cai, Stephen J. Turner, and Yong Wang. 2006. SOAr-DSGrid: Service-oriented architecture for distributed simulation on the grid. In *Proceedings of the IEEE Workshop on Principles of Advanced and Distributed Simulation*. IEEE Press, 65–73.

[22] Dan Chen, Stephen J. Turner, Wentong Cai, and Muzhou Xiong. 2008. A decoupled federate architecture for high level architecture-based distributed simulation. *J. Parallel Distrib. Comput.* 68, 11 (2008), 1487–1503. DOI: https://doi.org/10.1016/j.jpdc.2008.07.010

[23] Ke Pan, Stephen J. Turner, Wentong Cai, and Zengxiang Li. 2007. A service oriented HLA RTI on the Grid. In *Proceedings of the International Conference on Web Services*. IEEE Press, 984–992.

[24] Zengxiang Li, Xiaorong Li, Ta Nguyen Binh Duong, Wentong Cai, and Stephen J. Turner. 2013. Accelerating optimistic HLA-based simulations in virtual execution environments. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 211–220.

[25] Shaochong Feng, Yanqiang Di, and Zhu Xianguo Meng. 2010. Remodeling traditional RTI software to be with PaaS architecture. In *Proceedings of the 3rd International Conference on Computer Science and Information Technology*. IEEE Press, 511–515.

[26] Hui Yong Zhang, Cai Bin Liu, and Xiao Wen Bi. 2012. PaaS RTI-Supporting distributed simulation interaction in cloud computing age. *Adv. Eng. Forum* 6-7 (2012), 887–894. DOI: https://doi.org/10.4028/www.scientific.net/AEF.6-7.887

[27] Heng He, Ruixuan Li, Xinhua Dong, Zhi Zhang, and Hongmu Han. 2012. An efficient and secure cloud-based distributed simulation system. *Appl. Math. Inf. Sci.* 6, 3 (2012), 729–736.

[28] Kurt Vanmechelen, Silas De Munck, and Jan Broeckhove. 2012. Conservative distributed discrete event simulation on Amazon EC2. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 853–860.

[29] Shichao Guan, Robson Eduardo De Grande and Azzedine Boukerche. 2015. Enabling HLA-based simulations on the cloud. In *Proceedings of the 19th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. IEEE Press, 112–119.

[30] Shichao Guan, Robson Eduardo De Grande, and Azzedine Boukerche. 2019. A multi-layered scheme for distributed simulations on the cloud environment. *IEEE Trans. Cloud Comput.* 7, 1 (2019), 5–18. DOI: https://doi.org/10.1109/TCC.2015.2453945

[31] Gennaro Cordasco, Rosario De Chiara, Ada Mancuso, Dario Mazzeo, Vittorio Scarano, and Carmine Spagnuolo. 2013. Bringing together efficiency and effectiveness in distributed simulations: The experience with D-MASON. *Simulation* 89, 10 (2013), 1236–1253. DOI: https://doi.org/10.1177%2F0037549713489594

[32] Michele Carillo, Gennaro Cordasco, Flavio Serrapica, Carmine Spagnuolo, Przemysaw Szufel, and Luca Vicidomini. 2017. D-MASON on the cloud: An experience with Amazon Web Services. In *Proceedings of Euro-Par: Parallel Processing Workshops. LNCS*, Vol. 10104. Springer, Cham.

[33] Daniel Zehe, Alois Knoll, Wentong Cai, and Heiko Aydt. 2015. SEMSim Cloud Service: Large-scale urban systems simulation in the cloud. *Simul. Model. Pract. Theor.* 58, 2 (2015), 157–171. Retrieved from http://dx.doi.org/10.1016/j.simpat.2015.05.005

[34] Masatoshi Hanai, Toyotaro Suzumura, Anthony Ventresque, and Kazuyuki Shudo. 2014. An adaptive VM provisioning method for large-scale agent-based traffic simulations on the cloud. In *Proceedings of the International Conference on Cloud Computing Technology and Science*. IEEE Press, 130–137.

[35] Saad Zaheer, Asad W. Malik, Anis U. Rahman, and Safdar A. Khan. 2019. Locality-aware process placement for parallel and distributed simulation in cloud data centers. *J. Supercomput.* 75, 11 (2019), 7723–7745. DOI: https://doi.org/10.1007/s11227-019-02973-9

[36] Richard M. Fujimoto, Asad Waqar Malik, and Alfred J. Park. 2010. Parallel and distributed simulation in the cloud. *SCS M&S Mag.* IV, 3 (2010), 1–10. Retrieved from http://scs.org/wp-content/uploads/2016/12/2010-07-Issue03-4.pdf.

[37] Asad Waqar Malik, Alfred J. Park, and Richard M. Fujimoto. 2010. An optimistic parallel simulation protocol for cloud computing environments. *SCS M&S Mag.* IV, 4 (2010), 1–9. Retrieved from http://scs.org/wp-content/uploads/2016/12/2010-10-Issue04-2.pdf.

[38] Gabriele D'Angelo and Moreno Marzolla. 2014. New trends in parallel and distributed simulation: From many-cores to cloud computing. *Simul. Model. Pract. Theor.* 49 (2014), 320–335. DOI: https://doi.org/10.1016/j.simpat.2014.06.007

[39] Xiaocheng Liu, Xiaogang Qiu, Bin Chen, and Kedi Huang. 2012. Cloud-based simulation: The state-of-the-art computer simulation paradigm. In *Proceedings of the ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation*. IEEE Press, 71–74.

[40] Daniel Zehe, Wentong Cai, Alois Knoll, and Heiko Aydt. 2015. Tutorial on a modeling and simulation cloud service. In *Proceedings of the Winter Simulation Conference*. IEEE Press, 103–114.

[41] Robert Siegfried, Tom van den Berg, Anthony Cramp, and Wim Huiskamp. 2014. M&S as a service: Expectations and challenges. In *Proceedings of the Fall Simulation Interoperability Workshops. SISO*, 248–257.

[42] Erdal Cayirci. 2013. Modeling and simulation as a cloud service: A survey. In *Proceedings of the Winter Simulation Conference*. IEEE Press, 389–400.

[43] Simon J. E. Taylor, Azam Khan, Katherine L. Morse, Andreas Tolk, Levent Yilmaz, Justyna Zander, and Pieter J. Mosterman. 2015. Grand challenges for modeling and simulation: Simulation everywhere—From cyberinfrastructure to clouds to citizens. *Simulation* 91, 7 (2015), 648–665. DOI: https://doi.org/10.1177%2F0037549715590594

[44] Tamas Kiss, Huseyin Dagdeviren, Simon J. E. Taylor, Anastasia Anagnostou, and Nicola Fantini. 2015. Business models for cloud computing: Experiences from developing modeling & simulation as a service applications in industry. In *Proceedings of the Winter Simulation Conference*. IEEE Press, 2656–2667.

[45] Ian Bird, Bob Jones, and Kerk F. Kee. 2009. The organization and management of grid infrastructures. *Computer* 42, 1 (2009), 36–46. DOI: https://doi.org/10.1109/MC.2009.28

[46] eduGain. 2021. GEANT. Retrieved from https://wiki.geant.org/display/eduGAIN/eduGAIN+Home.

[47] Tony Hey and Anne E. Trefethen. 2005. Cyberinfrastructure for e-Science. *Science* 308, 5723 (2005), 817–821. DOI: https://doi.org/10.1126/science.1110410

[48] Brian Bockelman, Miron Livny, Brian Lin, and Francesco Prelz. 2021. Principles, technologies, and time: The translational journey of the HTCondor-CE. *J. Computat. Sci.* 25, 101213 (2021). DOI: https://doi.org/10.1016/j.jocs.2020.101213

[49] Chee Sun Liew, Malcolm P. Atkinson, Michelle Galea, Tan Fong Ang, Paul N. Martin, and Jano I. Hemert. 2016. Scientific workflows: Moving across paradigms. *ACM Comput. Surv.* 49, 4 (2017), 1–39. DOI: https://doi.org/10.1145/3012429

[50] Thomas Fahringer, Radu Prodan, Rubing Duan, Jüurgen Hofer, Farrukh Nadeem, Francesco Nerieri, Stefan Podlipnig, Jun Qin, Mumtaz Siddiqui, Hong-Linh Truong, Alex Villazon, and Marek Wieczorek. 2007. ASKALON: A development and grid computing environment for scientific workflows. In *Workflows for e-Science*, Springer London, 450–471. DOI: https://doi.org/10.1007/978-1-84628-757-2_27

[51] Eduardo Ogasawara, Jonas Dias, Vitor Silva, Fernardo Chirigati, Daniel de Oliveira, Fabio Porto, Patrick Valduriez, and Marta Mattoso. 2013. Chiron: A parallel engine for algebraic scientific workflows. *Concurr. Computat.: Pract. Exper.* 25, 16 (2013), 2327–2341. DOI: https://doi.org/10.1002/cpe.3032

[52] Jeremy Goecks, Anton Nekrutenko, and James Taylor. 2010. Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.* 11, R86 (2010), 1–13. DOI: https://doi.org/10.1186/gb-2010-11-8-r86

[53] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. 2004. Kepler: An extensible system for design and execution of scientific workflows. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management.* IEEE, 423–424.

[54] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. 2015. Pegasus, a workflow management system for science automation. *Fut. Gen. Comput. Syst.* 46 (2015), 17–35. DOI: https://doi.org/10.1016/j.future.2014.10.008

[55] Yong Zhao, Michael Hategan, Ben Clifford, Ian Foster, Gregor von Laszewski, Veronika Nefedova, Ioan Raicu, Tiberiu Stef-Praun, and Michael Wilde. 2007. Swift: Fast, reliable, loosely coupled parallel computation. In *Proceedings of the IEEE Congress on Services.* IEEE Press, 199–206.

[56] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Aanil Wipat, and Petr Li. 2004. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20, 17 (2004), 3045–3054. DOI: https://doi.org/10.1093/bioinformatics/bth361

[57] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. 2007. The Triana workflow environment: Architecture and applications. In *Workflows for e-Science,* I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields (Eds.). Springer, London, 320–339. DOI: https://doi.org/10.1007/978-1-84628-757-2_20

[58] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai, and Istvan Marton. 2012. WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities. *J. Grid Comput.* 10, 4 (2012), 601–630. DOI: https://doi.org/10.1007/s10723-012-9240-5

[59] Ewa Deelman, Karan Vahi, Mats Rynge, Gideon Juve, Rajiv Mayani, and Rafael Ferreira da Silva. 2016. Pegasus in the cloud: Science automation through workflow technologies. *IEEE Internet Comput.* 20 (2016), 70–76. Retrieved from http://dx.doi.org/10.1109/MIC.2016.15

[60] Piotr Nowakowski, Marian Bubak, Tomasz Bartyński, Tomasz Gubała, Daniel Harężlak, Marek Kasztelnik, Maciej Malawski, and Jan Meizner. 2017. Cloud computing infrastructure for the VPH community. *J, Computat. Sci.* 24 (2018), 169–179. Retrieved from http://dx.doi.org/10.1016/j.jocs.2017.06.012

[61] Enis Afgan, Dannon Baker, Nate Coraor, Hiroki Goto, Ian M. Paul, Kateryna D. Makova, Anton Nekrutenko, and James Taylor. 2011. Harnessing cloud computing with Galaxy Cloud. *Nat. Biotechnol.* 29 (2011), 972–974. Retrieved from http://dx.doi.org/10.1038/nbt.2028

[62] Sharath Maddineni, Joohyun Kim, Yaakoub El-Khamra, and Shantenu Jha. 2012. Distributed application runtime environment (DARE): A standards-based middleware framework for science gateways. *J. Grid Comput.* 10 (2012), 647–664. Retrieved from http://dx.doi.org/10.1007/s10723-012-9244-1

[63] Simon J. E. Taylor, Tamas Kiss, Anastasia Anagnostou, Gabor Terstyanszky, Peter Kacsuk, Joris Costes, and Nicola Fantini. 2018. The CloudSME simulation platform and its applications: A generic multi-cloud platform for developing and executing commercial cloud-based simulations. *Fut. Gen. Comput. Syst.* 88 (2018) 524–539. DOI: https://doi.org/10.1016/j.future.2018.06.006

[64] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai, and Istvan Marton. 2012. WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities. *J. Grid Comput.* 10 (2012), 601–630. Retrieved from http://dx.doi.org/10.1007/s10723-012-9240-5

[65] Tamas Kiss, Peter Kacsuk, Eva Takacs, Aron Szabo, Peter Tihanyi, and Simon J. E. Taylor. 2014. Commercial use of WS-PGRADE/gUSE. In *Science Gateways Distributed Computing Infrastructures Development Framework Exploited by Scientific User Communities,* P. Kacsuk (Ed.). Springer Cham, 271–286. DOI: https://doi.org/10.1007/978-3-319-11268-8-19

[66] Simon J. E. Taylor, Anastasia Anagnostou, Nura Abubakar, Tamas Kiss, James DesLauriers, Gabor Terstyanszky, Peter Kacsuk. Jozsef Kovac, Shane Kite, Gary Pattison, and James Petry. 2020. Innovations in simulation: Experiences with cloud-based simulation experimentation. In *Proceedings of the Winter Simulation Conference.* IEEE Press, 3164–3175.

[67] Anastasia Anagnostou, Athar Nouman, and Simon J. E. Taylor. 2013. Distributed hybrid agent-based discrete event emergency medical services simulation. In *Proceedings of the Winter Simulation Conference*. IEEE Press, 1625–1636.

[68] Anastasia Anagnostou and Simon J. E. Taylor. 2017. A distributed simulation methodological framework for OR/MS applications. *Simul. Model. Pract. Theor.* 70 (2017), 101–19. DOI: https://doi.org/10.1016/j.simpat.2016.10.007

[69] Tim Pokorny and Michael Fraser. 2021. The poRTIco Project. Calytrix Technologies, Perth, Western Australia. Retrieved from http://porticoproject.org/.

[70] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. 2009. Workflows and e-Science: An overview of workflow system features and capabilities. *Fut. Gen. Comput. Syst.* 25, 5 (2009), 528–540. DOI: https://doi.org/10.1016/j.future.2008.06.012

Received July 2019; revised October 2021; accepted December 2021