Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems

Massimo Ficco [a,*], Beniamino Di Martino [a], Roberto Pietrantuono [b], Stefano Russo [b]

[a] Seconda Università degli Studi di Napoli, Via Roma 29, Aversa (CE), Italy

[b] Università degli Studi di Napoli Federico II, Via Claudio 21, 80125, Naples, Italy

HIGHLIGHTS

- Hybrid simulation of complex critical systems in private cloud is presented.
- Simulation and emulation interoperability has been managed by federation.
- Allocation of simulation tasks is implemented by multi-objective approach.

ARTICLE INFO

Article history:
Received 24 September 2015
Received in revised form
9 January 2016
Accepted 30 January 2016
Available online xxxx

Keywords:
Large-scale critical systems
Hybrid simulation
Resource optimization
Middleware
Cloud computing

ABSTRACT

Simulation represents a powerful technique for the analysis of dependability and performance aspects of distributed systems. For large-scale critical systems, simulation demands complex experimentation environments and the integration of different tools, in turn requiring sophisticated modeling skills. Moreover, the criticality of the involved systems implies the set-up of expensive testbeds on private infrastructures. This paper presents a middleware for performing hybrid simulation of large-scale critical systems. The services offered by the middleware allow the integration and interoperability of simulated and emulated subsystems, compliant with the reference interoperability standards, which can provide greater realism of the scenario under test. The hybrid simulation of complex critical systems is a research challenge due to the interoperability issues of emulated and simulated subsystems and to the cost associated with the scenarios to set up, which involve a large number of entities and expensive long running simulations. Therefore, a multi-objective optimization approach is proposed to optimize the simulation task allocation on a private cloud.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Modern critical infrastructures play a key role in improving and preserving the quality of life. Few examples include power grids, smart cities, transportation (e.g., aerial, railway, maritime) systems, and financial IT infrastructures. Designing and managing complex critical infrastructures is increasingly challenging, as more and more systems are required to interoperate in order to provide new functionalities while assuring dependable and cost-effective implementations. Future generation of such systems in domains like air, naval, and railway traffic management, is expected to have a stronger focus on a System of Systems (SoS) concept as driving design criterion, as new needs for interoperability among (often independent) actors are rapidly rising.[1] As a result, the design, development and analysis of such systems is extremely complicated, due to the integration of heterogeneous and widely distributed systems, of new and legacy entities, of proprietary and off-the-shelf components into a unique design. Besides functionality, performance and dependability requirements can become hard to satisfy in such complex scenarios, because of subtle defects difficult to reproduce [1,2].

A powerful tool to support the lifecycle of these systems is simulation. Being able to simulate their behavior accurately would allow engineers to evaluate alternative design decisions, to

* Correspondence to: Dipartimento di Ingegneria Industriale e dell'Informazione, Seconda Università degli Studi di Napoli, Via Roma 29, Aversa (CE), Italy.
E-mail address: massimo.ficco@unina2.it (M. Ficco).

[1] Examples are initiative like the SESAR program: "As the technological pillar of Europes ambitious Single European Sky (SES) initiative, SESAR (Single European Sky ATM Research) is the mechanism which seeks to coordinate and concentrate all EU research and development activities in ATM, pooling together a wealth experts to develop the new generation of ATM"—http://www.sesarju.eu.

http://dx.doi.org/10.1016/j.future.2016.01.022
0167-739X/© 2016 Elsevier B.V. All rights reserved.

assess the expected performance and dependability under several different scenarios, to pinpoint possible architectural bottlenecks, so as to drastically reduce the cost for testing and maintenance by setting up onsite simulated testbeds instead of expensive (often unfeasible) real test scenarios. All this favors a suitable design and an early detection of possible problems before the actual deployment, after which the occurrence of failures could be extremely harmful for both producers (indeed, cost of operational failures is much higher than cost of pre-deployment failures) and for end-users (operational failures of such infrastructures could cause serious damages to people and/or the environment). On the other hand, simulation of these complex and distributed systems present serious challenges. Different simulation tools, simulation environments, real sub-systems (usually Commercial Off-The-Shelf) and experimental platforms need to interact in a coordinated way. Such an integration requires sophisticated modeling practices and complex experimentation environments. In addition, despite the advantages of simulation, the complexity of the systems to simulate and the large number of involved entities can lead to very high cost and simulation time.

This work proposes a middleware to implement locally controlled testbed for large-scale mission-critical systems by means of cost-effective distributed and hybrid simulation techniques. The middleware allows setting up simulation platforms integrating emulated and simulated subsystems on distributed testbeds in order to closely mimic the real behavior of the scenarios under test. It offers services for the seamless integration and interoperation among simulated and emulated subsystems and for supporting decisions of the simulation manager about the optimal simulation planning.

The middleware is based on the High Level Architecture (HLA) paradigm [3], a standard for distributed computer simulation that allows the integration of multiple independent simulation environments within a more complex federated simulation system. An HLA-based implementation enabled us to address the critical challenge, exacerbated in large-scale and mission-critical scenarios, of highly heterogeneous environments that need to communicate to each other (e.g., ensuring the accuracy of the experiment threatened by the different time domains on which the simulated and the real platform operate and by communication overhead between the simulation and the emulation environment). On top of this platform, a cost-effective management of simulation resources is implemented. Indeed, very high costs are entailed by the complex scenarios of the simulated systems of systems and by the high number of entities involved. On the other hand, the criticality of the involved systems imply the set-up of testbeds on private infrastructure. Therefore, the careful planning of such scenarios over the available resources is crucial. In the proposed solution, resources are managed via virtualization – a solution that allows overcoming the HLA shortcomings in managing and scheduling capabilities of resources – enabling the on-demand elastic deployment of simulation environments on modern cloud platforms [4]. Based on this, an optimization approach is implemented to support the simulation manager decisions about how to allocate simulation tasks, based on their features, to the available cloud resources, while controlling the total cost of the simulation and its running time. A multi-objective model is proposed, solved by three well-known evolutionary algorithms, which provides the best scheduling of simulation tasks on resources according to a user-desired objective to optimize.

The middleware is designed to support development of complex and critical infrastructures. It is being developed within the frame of a public–private research project named *DISPLAY (Distributed hybrId Simulation PLAtform for ATM and VTS sYstems)*, which is realizing a distributed and hybrid simulation platform for engineering systems of Air Traffic Control (ATC) and maritime Vessel Traffic Systems (VTS). The platform will support the system engineers, enabling the relatively fast deployment of complex scenarios on local testbeds, favoring alternative system design evaluations, pre-deployment testing activities and maintenance operations—the latter two being usually very expensive tasks, due to the wide geographically distributed nature of the considered systems. The platform is expected to remarkably reduce ATC and VTS production and maintenance cost.

The rest of this paper is organized as follows: related work is presented in Section 2. Section 3 shows the logical view of the hybrid simulation middleware named 'DISPLAY'. A detailed description of the DISPLAY implementation is presented in Section 4. A typical simulation scenario of a Vessel Traffic System is described in Section 5. The optimization model of simulation tasks on virtual resources and its experimental evaluation are presented in Sections 6 and 7. Section 8 concludes the work.

2. Related work

Several works proposed unified stand-alone frameworks for simulating complex systems, usually based on a single specialized specification language used to represent the whole system. For example, MATLAB/Simulink packages and tools are available for implementing discrete event simulations, as well as providing interfaces among different discrete event/continuous time blocks [5]. A discrete event-based hybrid simulation solution based on the DEVS formalism and dealing with atomic or coupled models can be found in [6]. Similarly, a hybrid approach for system modeling based on visual syntax is presented in [7]. However, such approaches can provide a limited re-usability of the implemented system models.

Other approaches support the integration of separated simulation models by means of ad-hoc interfaces. For example, some interesting tools for integrating simulation models in hybrid control environments are provided by the BCVTB [8] and by CODIS [9] MATLAB/Simulink interfaces. The former works within the Ptolemy II framework integrated within a MATLAB/Simulink scenario. The latter supports discrete and continuous simulations, providing a co-simulation bus for integrating both a MATLAB/Simulink and a SystemC model. The different models interoperate by using specific Inter-Process Communication (IPC) primitives or interface libraries.

A more recent approach uses agent-based simulations to model and study complex multi-actor systems, complementing the long-established system dynamics and discrete-event simulation approaches [10,11]. Krozel [12] uses the Future ATM Concepts Evaluation simulation Tool (FACET) to model air-traffic in inclement weather. Agogino [13] exploits agent-based simulation for incremental enhancements of the air-traffic management (ATM). Gorodetsky [14] proposes an agent-based simulation for ATM within the air-space of large airports. The *AgentFly* [15] system, built by using the *AglobeX Simulation* platform, also aims to provide a model of the air traffic.

An alternative approach provides HLA/RTI-based interoperability among discrete event-based models and continuous ones, through an explicit spatial separation between them. For example, the framework proposed in [16] implements an HLA/RTI-based interface enabling the complete interoperability between DEVS and the MATLAB/Simulink models. Such a framework leverages the HLA/RTI time management facilities to synchronize the above models, by also relying on the analog/event interface mechanisms for data exchange among them. Furthermore, the mechanism proposed in both [17] and standard MATLAB implements an HLA/RTI interface provided as a specific package or library. There are some

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

European projects that are also taking care of simulation for critical infrastructures: CRIPNET[2] (Critical Infrastructure Preparedness and Resilience Research Network) aims at developing new knowledge about modeling, simulation and analysis for critical infrastructure protection and forming the foundation for the European Infrastructures Simulation and Analysis Centre (EISAC) by 2020; DIESIS (Design of an Interoperable European federated Simulation network for critical InfraStructures)[3] aims at developing a standardized pan-European platform for modeling and simulation tasks as basis for the transnational exploration of safety aspects of the critical European infrastructures. These initiatives witness the increasing interest about this topic.

Along this line, we presented, in a previous work, a solution to make simulated and emulated components interoperable, by implementing mechanisms to manage the time progress and synchronization of the involved components [18,19]. In this work, we describe the HLA-based middleware for hybrid simulation where the focus is not only on the interoperability and synchronization problems, but it is, at a higher level, on the optimal management of resources to run cost-effective simulations and provide actionable results to engineers. Specifically, a multi-objective (MO) solution is proposed for the optimal planning of simulation sessions over the implemented middleware. MO optimization is exploited in several fields of research, ranging from engineering to economics, finance, logistics, project planning. As a powerful means for multiple criteria decision making, it is recently being applied to the management of cloud resources, mainly for quality of service (QoS) optimization, load balancing, and energy efficiency. Several approaches are proposed to schedule tasks, processes and resources in the cloud—useful surveys and taxonomies are provided in [20,21]. The approaches often target objectives related to the optimization of QoS parameters, such as execution time, deadline, cost, bandwidth of communication, make-span, reliability, and others. MO criteria have overcome the conventional usage of single-objective priority assignment to make allocations, wherein tasks with the highest priority are executed first (e.g., [22,23]). For instance, multiple criteria, corresponding to multiple QoS parameters, are used in [24,25], where an algorithm for task scheduling to virtual machines is proposed. In [26], the authors use the NSGA-II algorithm, a well-known evolutionary metaheuristic, for load balancing of CPU, memory and bandwidth. In [27,28], the ant colony optimization is adopted to look for solutions minimizing the execution time and cost of tasks. In [29], authors implement a MO task scheduling algorithm assigning tasks to VMs aimed to improve the throughput of the datacenter and to reduce the cost in SaaS environments. Malawski et al. develops several multi-criteria static and dynamic algorithms for task scheduling and resource provisioning accounting to optimize cost and makespan for scientific workflow [30]. The work in [31] adopts a MO approach looking for tradeoffs between cost and makespan in task scheduling and resource allocation problem, by using an improved differential evolution algorithm (IDEA) compared with other metaheuristics (namely, with NSGA-II, DEA, SPEA2 and IBEA). The algorithm proposed by Vasile et al. addresses the resource-aware hybrid scheduling for batch jobs and workflows by accounting for hierarchical clustering of the available resources into groups in the allocation phases [32]. Several papers also deal with energy consumption optimization, rather than QoS parameters. For instance, the work in [33] formulates the energy efficiency resource allocation for cloud computing as a MO optimization problem, solved by NSGA-II. The paper in [34] proposes and energy efficient scheduling algorithm of VMs considering the deadline constraints. Although, most of works are devoted to minimize the execution time or cost, there are also some papers that address the problem of increasing the scalability [35] and reliability by optimal allocation [28,36–38]. All these works are relatively recent research, which reflect a felt need for solutions that support providers or cloud platform managers to exploit resources optimally. Our proposal is along the same line, in that we aim at exploiting such a research trend in the field of hybrid and distributed simulation management, where the problem of the optimal resource allocation is still unaddressed. Our goal is to fully leverage cloud-based technology for hybrid simulation of complex large-scale systems, for which virtualization is a very attractive (and maybe the only) alternative to implement scalable realistic simulation scenarios.

3. A middleware for supporting hybrid simulations

Fig. 1 shows a logical view of the proposed solution, which is organized according to three levels. The highest level includes Application Programming Interface (API) for interfacing with the external environment. In particular, specific interfaces are provided to configure the simulation process for both its static (i.e., at the architecture level) and dynamic (i.e., at the level of functional process) behavior. At user-level, the modules responsible for configuring and managing the simulation scenarios are implemented, including (i) the *Configuration Manager*, which interacts with the final user by the high level API to define the simulation scenario, as well as with the lower-level modules to configure the simulation environment (Fig. 2); (ii) the *Network Manager* responsible for setting-up and managing the network emulation scenario; and (iii) the *Policy Manager* for user profiles administration.

Finally, the system-level includes the basic functionalities provided by the implemented DISPLAY middleware:

- *Simulation Manager*—It is enabled to configure, manage and monitor the hybrid simulation process. The Simulation Manager allows defining the system behavior in terms of interactions among the involved subsystems that best identify the simulated application scenario. It interfaces with the Configuration Manager to offer its capabilities at user-level in order to operate on the simulation process in real-time.
- *Data Distribution Manager*—It is in charge of managing the constant exchange of information among components involved in the simulation, providing services for data distribution and information synchronization.
- *Time/Event Manager*—In order to simulate distributed environments, it is necessary that each involved component perceives the progress of time in a uniform manner, regardless of their world of origin (real, emulated, simulated). Time/Event Manager is responsible for the correct advancement of time, by managing the alignment of real and emulated components' time with the simulation time, which constitutes the reference variable. The progress of time among simulated and emulated components is based on events. To share the time reference among the various components, specific wrappers are implemented that wait for events received by components and interact with them for managing the time progress. Specifically, to enable synchronization, the Time/Event Manager receives all the time events by simulators, and selects the smallest non-negative value; this is then sent to all components and wrappers as the actual time to run.

[2] https://www.ciprnet.eu.
[3] http://www.iais.fraunhofer.de/.

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

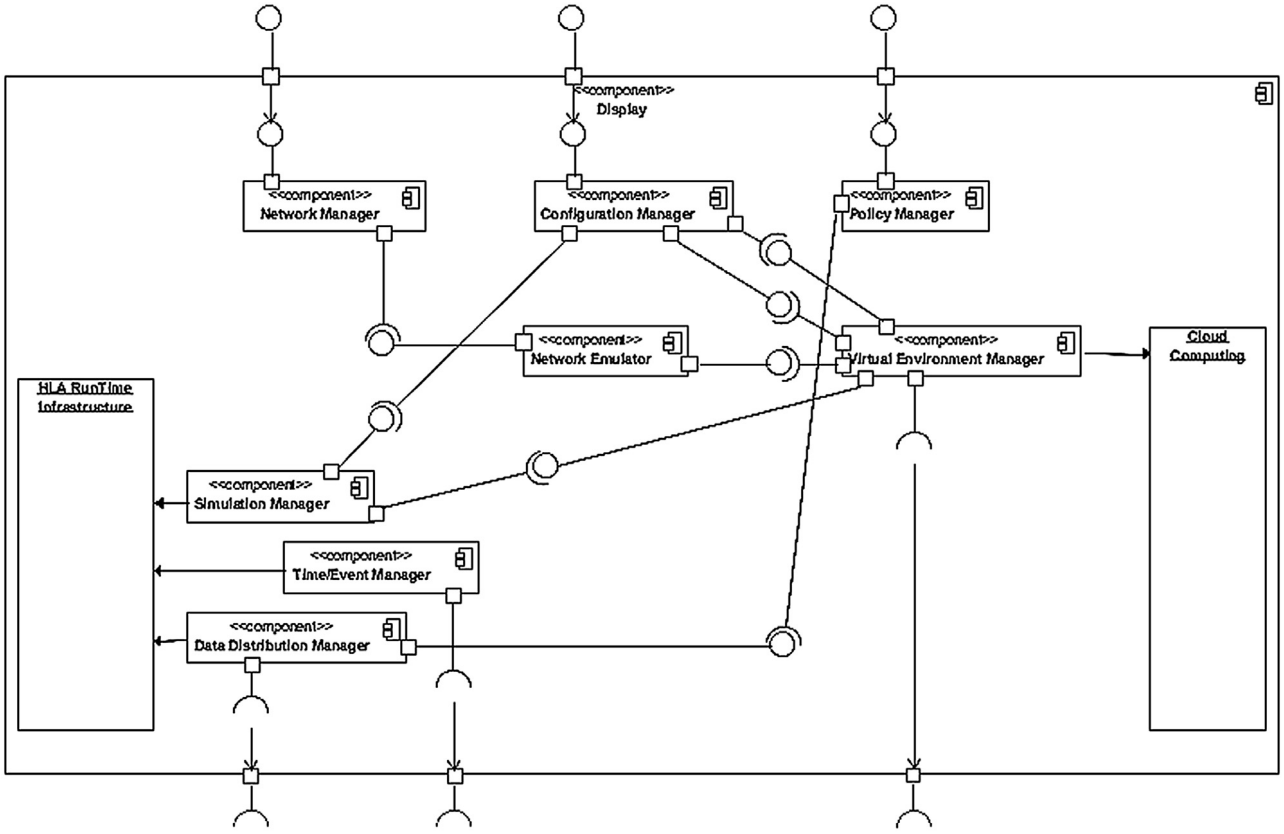


Fig. 1. Logical view of DISPLAY middleware.

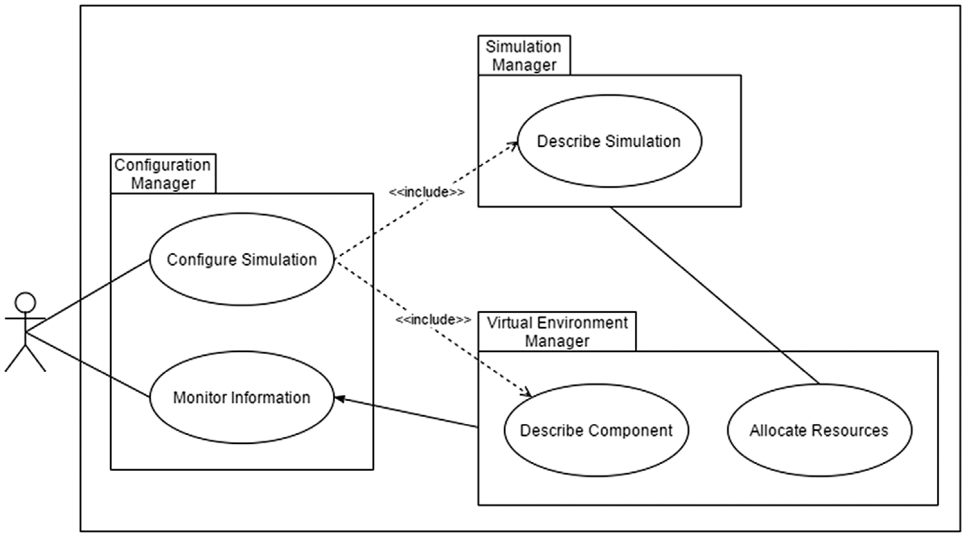


Fig. 2. Configuration manager.

- *Network Emulator*—Given the very complex nature of the considered network-centric systems, we adopt a distributed network emulation solution to reproduce reality with a high degree of verisimilitude. As shown in Fig. 3, the Network Emulator interacts with both the Network Manager, from which it receives a description of the network scenario to deploy, and the *Virtual Environment Manager* to implement one or more instances of the emulated network on virtualized resources.
- *Virtual Environment Manager*—This component enables the dynamic deployment, management and monitoring of virtual resources, which are necessary for the implementation and orchestration of the local testbed. It instantiates virtual resources based on information it receives, in XML format, from the Configuration Manager. Moreover, it is able to: (i) manage the lifecycle of virtual machines; (ii) re-size dynamically the virtual resources; (iii) manage the lifecycle of the network resources and of the virtual storage.

In the next section a detailed description of the proposed implementation is reported.

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

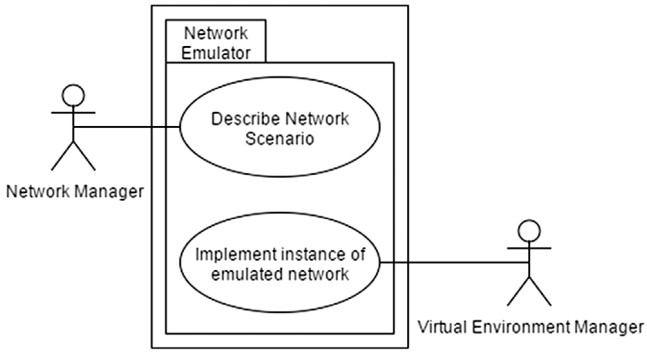


Fig. 3. Network emulation management.

4. DISPLAY middleware implementation

The proposed middleware aims at providing an expandable hybrid simulation environment, which allows the integration and interoperability of different simulation models and emulated subsystems while allowing to configure, manage and monitor the simulation scenarios in an easy and efficient way.

To support the integration of distributed and independent simulation environments, each with its own features, languages and operating systems, within a more complex federated simulation system, the High Level Architecture (HLA) simulation structure is adopted. In particular, according to the HLA paradigm, a *federation* is a distributed simulation system for a particular purpose, which consists of a number of interactive members. Each application program (e.g., simulation tool) that participates in a simulation is called *federate*. The interface specification of the HLA describes how to communicate within the federation, and is implemented by the *Run Time Infrastructure (RTI)*. RTI implements all the services of HLA interface specification, and provides a range of service functions to support interoperability of the federation members. Federates can assume the role of *Publisher* to publish information within the federation, and the role of *Subscriber* as well, to receive information created and updated by other federates. In HLA, the information exchanged is represented by using the classical object oriented paradigm. The main entities belong to the *Object* and *Interaction* classes. The Object class refers to object-oriented data, characterized by specific attributes, shared within the federation that persist during the run time, whereas Interactions are associated with events exchanged among federates. These entities are implemented by the XML format. Each federate can register object instances and modify their attributes, so that the other federates, subscribed to the involved objects, receive value updates for these attributes. Interactions work in a similar way, except that an interaction is only used once with a specified set of parameter values and then discarded.

As a framework for advanced distributed interactive simulation, HLA meets the needs of DISPLAY simulation middleware. In particular, as shown in Fig. 4, three out of five functionalities offered at system-level, namely *Simulation Manager*, *Time/Event Manager*, and *Data Distribution Manager*, are implemented by using RTI services. The *Simulation Manager* is responsible for the simulation process life-cycle (Fig. 5). Specifically, according to the HLA framework architecture, the *Simulation Manager* provides mechanisms for managing the federation members, as well as for specifying the exchange of data among the federates. It uses a common, standardized, formalism for describing the capabilities of potential federation members. In particular, the subsystems (federates) to be simulated are modeled according to the HLA object model. Each federation member is represented by an HLA simulation object model (SOM), whereas the interactions among the federates are described by the federation object model (FOM).

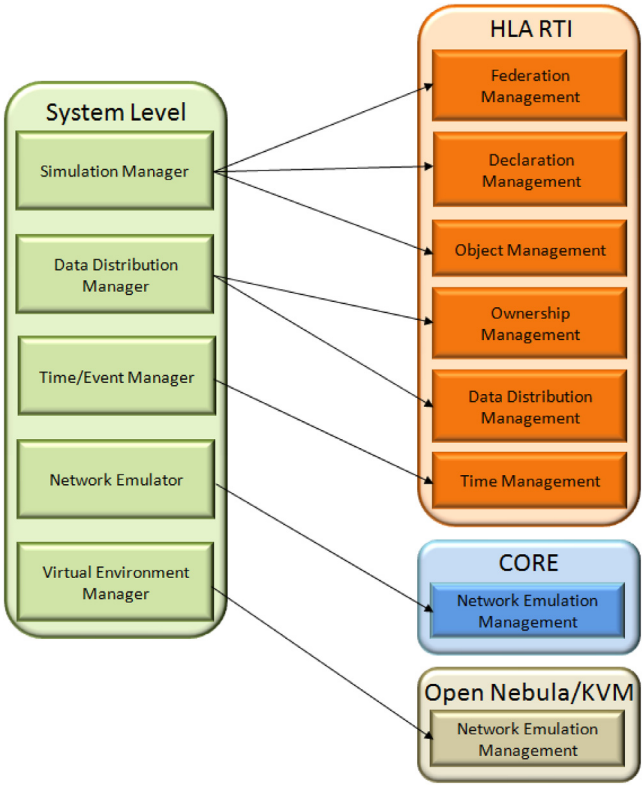


Fig. 4. Mapping of RTI services at system-level.

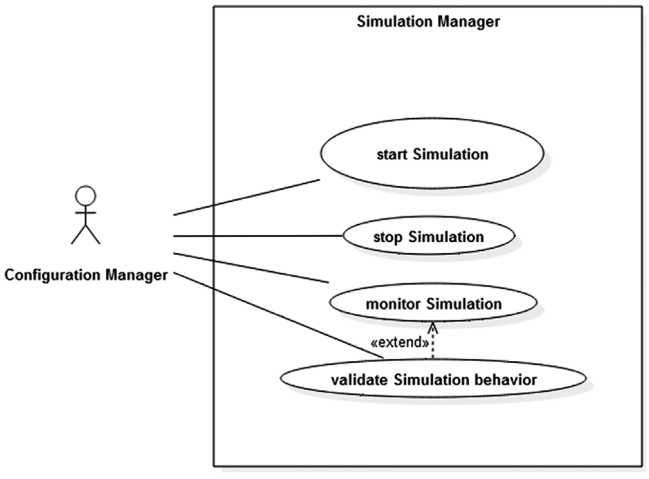


Fig. 5. Simulation use cases.

SOM specifies the types of information that a federate can provide to HLA federations, as well as of information that a federate can receive from the others. FOM is specified by a file that contains a description of data exchanged in the federation, e.g., objects and interactions that will be exchanged. This can be seen as the "language" of the federation. During a federation execution, all exchanges of FOM data among federates shall occur via the RTI. RTI specifies a group of interface services supporting federation members in accordance with the provisions of the FOM.

To correctly exchange simulation data among federates, which evolve along a different temporal model (real time, emulated, and simulated), the *Time/Event Manager* exploits the RTI to coordinate how fast the simulators advance in their logical and emulation scenarios. Moreover, it allows defining synchronization points that enable the members of the federation to coordinate when they

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

have reached a certain state (for example, when they are ready to start the simulation of the next phase of a scenario).

Data exchange within the federation is implemented through the *Data Distribution Manager* by exploiting the publish/subscribe paradigm provided by the RTI. For individual federates, SOM developers should associate whatever information they feel could be needed by subscribers with their publishable interaction classes. In addition, for interaction classes to which a federate can subscribe, SOM developers should determine what type of information needs to be included, so that the federate can calculate the associated effects.

The adopted HLA–RTI implementation including the three described functions is based on the open-source cross-platform *PoRTIco* [39], a modular and flexible platform providing a production-grade RTI environment.

The *Network Emulator* enhances the overall HLA-based architecture with network services needed for a distributed and hybrid simulation. It has been structured as a distributed emulation environment, able to easily manage multiple emulated networks and systems over the same shared infrastructure. It is based on Common Open Research Emulator (CORE) [40].

The last component of the architecture is the *Virtual Environment Manager*, in charge of mastering complexity by means of virtualization and optimization functions. To implement complex simulation scenario, the virtualization infrastructure must provide an efficient configuration platform, so as to manage the execution of a large number of simulated subsystems, emulated network devices/links, applications under test, and storage servers (storing the huge amount of data produced during the simulation process) [41], all running over the same shared cloud infrastructure. The use of virtual machines (VM) decouples the execution of specific software components from the involved system hardware, allowing a complete control over their run-time operations, as well as over their perception of time. This also enables the transparent suspension of the execution of any networked entity in order to ensure synchronization between simulated time and real-world time in emulated components, e.g., (i) by letting the simulation run while the VMs are suspended, waiting for time synchronization or, alternatively, (ii) suspending the simulation time while VMs are live-migrated over the cloud to search for new available resources. Therefore, in the context of the DISPLAY project, the implemented *Virtual Environment Manager* supports: (i) the configuration of the virtual infrastructure; (ii) the optimization of the simulation task allocation on a private cloud; (iii) the deployment of simulation and emulation instances on the cloud; and (iv) the monitoring of virtual resources. In particular, we used KVM as virtualization engine, whereas OpenNebula (ONE) open-source technology [42] is used to build up the private cloud, providing the *IaaS* facility. The ONE platform assumes that the underlying physical infrastructure is built up in line with a classical cluster-like architecture, where the hypervisor-enabled nodes are managed by a so-called front-end node, responsible for the centralized management and monitoring of ONE services. Such services include a user interface for submitting VMs and continuously checking their status. The front-end node manages the whole life-cycle of clusters, hosts, VMs, storage areas and other virtualized resources. Furthermore, ONE supports the elastic deployment and reallocation of VMs on a group of run-time/storage resources fully distributed throughout the physical network, by decoupling the individual entity not only from the underlying infrastructure but also from its location constraints. The monitoring of the virtual resources is implemented by using *Host sFlow* agents, which infer and export virtual nodes' performance metrics by using the *sFlow* protocol with minimal impact on the systems being monitored [43]. Based upon this virtual resources management, there is the optimal simulation planning feature implemented to enable an efficient scheduling of tasks over available resources on the private cloud. This is detailed in Sections 6 and 7.

4.1. Simulation interoperability

The typical simulation scenarios of a large-scale critical system could require information available only on site by interacting with the real system. Therefore, besides the management capabilities of the simulation, there is the need to incorporate, in the DISPLAY middleware, a functionality to interconnect simulations with emulated networks and the real system. As shown in Fig. 6, we introduce a specific federate (named *EmuGateway*), which operates as a gateway between the simulation federates and the real system [44], which supports the integration of simulation and emulation environments. A *Bridge* allows the real system interacting with the simulation federates through the emulated network.

We assume that the real system runs on physical hosts with a specific IP address assigned. Moreover, one or more UDP and TCP ports are associated with the real system, providing specific communication endpoint facilities and external interfaces to its application services. The main goal of the *EmuGateway* is to manage the publication and subscription of federates to messages to be sent to (or coming from) the real system, as well as to communicate with the real system by using IP sockets on the aforementioned service ports. In particular, the *EmuGateway* federate is a multi-thread server, which accepts connections from the real system and processes each interaction exchanged between the latter and the simulation federates. It can both publish real system messages over RTI, and wrap each incoming message from the RTI forwarding it directly to the target application service. Message delivery is managed according to a *store and forward* policy based on dedicated application queues. To enable application service to process the interaction, the *EmuGateway* adds to each message sent by federates:

1. a timestamp representing the time at which the real system has to process the interaction;
2. the name of the target application service (the process' port at the end host) that has to handle the interaction;
3. the IP address of the node hosting the real system process.

The *Bridge* receives the message, and converts the timestamp from the simulation time to a real time, then forwarding the message to the associated application.

However, real/virtual hosts on which the applications will be deployed might not be known at modeling time. Therefore, when the simulation starts, each bridge registers itself to the *EmuGateway* federate, by sending its IP address, the port of the associated real system, and its current system time. The *EmuGateway* assigns a symbolic name to each registered real system's application. Such symbolic name is used to identify the application as a federate. Based on this information, the *EmuGateway* manages the mapping information from the symbolic name to the host IP addresses.

Finally, as the HLA was developed with a specific focus on simulation in Local Area Network (LAN) or in Virtual Private Network scenarios, we implemented a service provisioning model of RTI based on web services, so as to make the local federates interoperable, through WAN links or through the Internet, with federates that run remotely. Such a solution, can also enable the interoperability of simulation subsystem with other SOA-based commercial systems, like Google Earth, as well as establish communication links through firewalls that block RTIs' underlying communications [45,46,44]. Thus, in order to support RTI interactions on the Web, a stateless Web service (named *Web Federate*) is implemented (Fig. 6). Web Federate is a special federate in charge of creating an RTI service instance associated with the remote federate, and a local federate inside the LAN. The Web Federate provides standard HLA service APIs, which

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

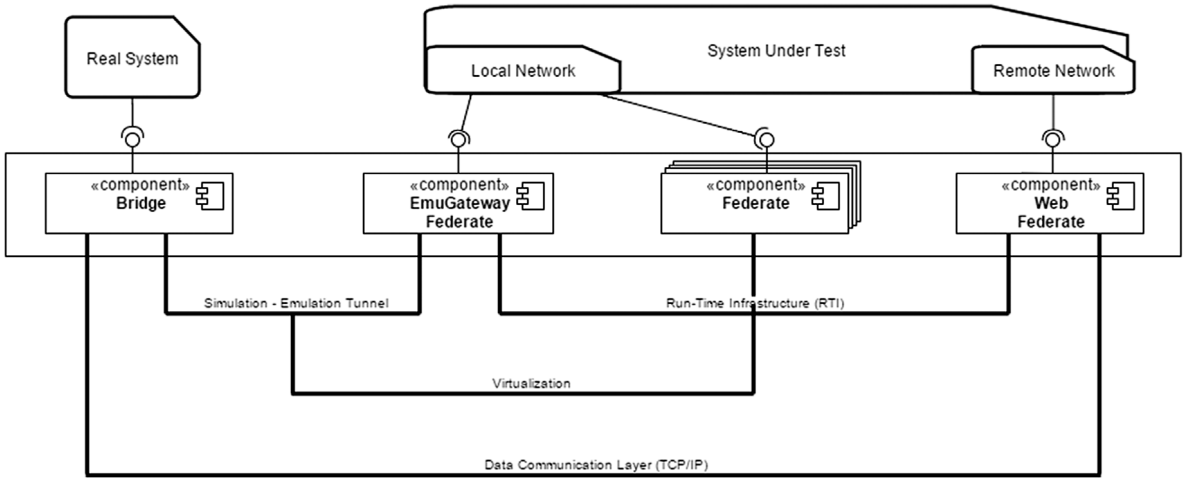


Fig. 6. Hybrid simulation architecture.

enable the encapsulation of service requests from federate to RTI into REST messages and transmit them to the remote federate. In particular, a Web Federate consists of two components: the *FederateToWAN* module providing RTI messaging proxy service and implementing the HLA services on the WAN as Web services; and the *FederateToLAN*, located inside the LAN, that provides standard HLA services to inter-operate with local federates. The remote federate receives remote HLA REST-based requests, and invokes HLA services of the *FederateToWAN* module by using the HTTP protocol.

5. Simulation scenarios

As described above, a paramount responsibility of the *Virtual Environment Manager* to manage complex simulation scenarios is the efficient management of (virtualized) resources. To this aim, a core functionality is the optimal allocation of the simulation components to resources over the cloud infrastructure. Therefore, in this section, we report a typical simulation scenario, which the described middleware solution has to manage, highlighting the main requirements and constraints of the cloud resources management task. The proposed solution is then detailed in the next section.

As mentioned in Section 1, the proposed middleware moves from a research project named DISPLAY (Distributed hybrId Simulation PLAtform for ATM and VTS sYstems), which aims at realizing a distributed and hybrid simulation middleware to be concretely applied in industrial contexts like Air Traffic Control (ATC) and Vessel Traffic System (VTS), both civil and military. To illustrate a typical scenario that engineers might want to simulate, we refer, in the following, to the VTS case.

The objective of a VTS is to support safety and efficiency of navigation, as well as protection of the marine environment, adjacent shore areas, work sites and offshore installations from possible adverse effects of maritime traffic. To provide the expected traffic management services, the VTS architecture has to include at least two kinds of site: remote Sensor Sites and VTSLs. The former include all the available sensors to track ship position and environment evolution. The latter provide the first level of control, where operators can interact with the system. VTSs can integrate a wide variety of sensors (e.g., radars, Electro Optical Sensor (EOS), Automatic Identification System (AIS) [47], direction finders, etc.) and of external systems (e.g., LRIT, weather), allowing for importing/exporting data in many formats (e.g., ITU1371 [48] supported both in version 1 and 3, NMEA). Starting from the tracks recognized from the Sensor Sites, VTSLs are responsible for process and store such tracks. The VTS system is a database-centric system, therefore all the system elements (servers and operators) are able to establish remote connections with database.

VTS can be seen as a system of systems, deployed in a complex environment like a harbor or along a coastal area. Ships targeting could be performed by using a large variety of different sensors. Each sensor has its capabilities, functionalities and operating modes. Furthermore, data generated from sensors vary in content, quantity, and use different formats. VTSs provide integration of such different data sources to track and identify ships navigating within the areas of interest. In addition, a VTS integrates systems produced by different companies and using different communication protocols.

A big effort is made to join all sensor data in a single coherent set of information identifying in a clear unequivocal way the target. Adding new sensors could be a very tough task, but the most critical aspect is the operational integration of the whole system. This last task requires the presence of all the sensors at the same time, a representative amount of traffic, as well as the recreation of true on site condition. Operational integration of VTSs is an expensive task and doing it on site lead to additional costs that could bias in a severe way costs of the entire project. To cope with such an eventuality, companies can leverage in-house simulation and emulation of the involved systems, trying to reproduce the scenario of interest as faithfully as possible.

An example of simulation scenario is the generation of an increasing load to be processed by the VTS, which represents the system under test. In particular, the load consists of an increasing number of simulated marine objects (i.e., ships to be monitored), ranging from 100 to 5000. Each ship follows a different path. Moreover, the test scenario includes several sensors, such as an Identification Base Station, 10 radar, 4 cameras, 1 weather stations, and 4 Direction Finders. The data collected by sensors are updated with a rate of 3 s and sent to the VTS. The network scenario involves geographical WAN links, both the up-link and down-link, at a throughput of 35 140 Mbps, while the LAN connection are set to 100 Mbps. Each kind of sensor and marine object is simulated by using a different tool. Therefore, the implementation of a simulation scenario requires setting-up several software components, called 'simulation tasks', such as the simulation tools, the network emulator (Core), the system under test (VTS), and the DISPLAY middleware used to synchronize and make the simulated and emulated subsystems interoperable. To have an idea of the complexity of these test scenarios, in terms of computational resources needed to process the simulation, suffice it to mention that the setting-up of the described simulation testbed requires 6

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

nodes M820 Blade DELL (2 Quad-Core), with 16 GB of RAM, and 72 GB of HDD. The simulation takes about 7 h to complete, and involves on average 34% of nodes' computational resources, with an average throughput of 1768 Mbps.

The assessment of performance and dependability attributes of a complex system like the VTS requires the simulation of several of such test scenarios. Moreover, each scenario can be repeated a large number of times with different settings, and each simulation can require up to several hours to complete. Therefore, to reduce the total time of experimental campaign, more simulations should be performed simultaneously. On the other hand, since such scenarios involve critical data and systems, the simulations must necessarily be performed on private clouds for security concerns—thus, with a limited number of nodes.

With such needs and constraints, the total time of a simulation and the cost for resources usage are a serious concern. In fact, in the foreseen industrial context, it is adopted a model where cloud resources are shared among the various development teams in different business divisions within the company. The DISPLAY middleware (specifically, the *Virtual Environment Manager*) tracks the shared platform computing usage by each team and division, through chargeback tools, and divisions are charged only for the resources they use. Therefore, in order to reduce the amount of required computational resources (and the associated cost) of a simulation while also reducing the simulation time, the *Virtual Environment Manager* implements a multi-objective model to distribute the simulation tasks among the available resources. The model and its experimental evaluation are presented in the following.

6. Optimized allocation of simulation tasks

Considering the large number of simulation tests that could be performed and the characteristics of the involved testbed, the cost and time entailed by simulation can be very high. There is a strong need for policies able of supporting a careful planning of simulation tasks and of wisely managing the assignment of available cloud resources to tasks. We have developed a decision support mechanism based on a multi-objective optimization evolutionary algorithm. The algorithm takes, as input, the declared characteristics of tasks to be simulated, the "importance", in the form of [0, 1]-weights, attributed to three contrasting objectives of a simulation, and a description of resources. The output is a set of solutions (i.e., allocations of tasks to resources) that propose several alternatives trade-offs, which are optimal with respect to the specified objectives in the sense explained below—i.e., the so-called Pareto front. In the following, we first formulate the problem, then describe the solution, and its evaluation in two illustrative scenarios.

6.1. Problem formulation

In a simulation scenario like described in Section 5, suppose there is a set of n simulation tasks $T = (T^1, T^2, \ldots, T^n)$, which can come from various business divisions, to be assigned to a set of m resources, $R = (R^1, R^2, \ldots, R^m)$ available in part in a local testbed, and in part in a remote site. The allocation should be done in order to minimize one or more objectives of interest. A simulation task is characterized, as in [31], by the amount of data and operations it deals with (i.e., the task "size" including both program and data size [49]) and by data it *exchanges* with other tasks. Hence, we distinguish these attributes: T^i_{PR} (*Processing requirement*) in terms of task size (in MB), and T^i_{DER} (*Data Exchange Requirement*), which determines the network utilization (in MB). This, in line with cloud computing resource allocation model, allows establishing which resources a task is requiring in terms of CPU and network.

A resource, in turn, is characterized by: a processing capacity in terms of MB per hour (R^j_{PC}) and a maximum network bandwidth (R^j_{BC}) in Mbps (Megabits per second), Moreover, different resources over the cloud may have different cost of resource usage, depending on many technical and commercial factors. For the considered resources, we account for: the rent cost of processing resource in USD per hour, $R^j_{Cost_{proc}}$, and the cost of the network utilization in USD per MB, $R^j_{Cost_{Net}}$. The specific cost model for the resources is specified by the private cloud platform owner; generally, in a private cloud, the cost of usage is derived by considering the costs of hardware, software, energy, cloud management tools, shared services, staffing, management and maintenance associated with a resource. This leads to a synthetic index expressed in USD per usage unit. Additionally, in the described HLA-based platform architecture, a resource may be local or remote. Of course, the former will have a network latency and response time much lower than the latter, and this needs to be characterized preliminarily, in order to optimally allocate resources accounting for it. Therefore, a series of preliminary ping tests need to be run in order to estimate the average latency and assign it to each resource in the system.

The objective of the algorithm is to find an optimal allocation of tasks to resources. A solution can assign more tasks to one resource, whereas one task must necessarily be assigned to only one resource. Preemption is not allowed; namely, tasks assigned to a resource cannot be interrupted and assigned to another resource. To ease the treatment, we assume, in this implementation, that if there are tasks related by a precedence relation (i.e., a task must execute before another one), these are treated as an atomic task (namely, assigned to one resource); thus, task executions are independent, and, when multiple tasks are assigned to a resource, they will be executed sequentially, without requirements on a specific sequence. Of course, resources cannot perform more than one task at a time.

Objectives of interest are: (i) to find allocations that minimize the total *cost* of task execution, including the cost for processing and network utilization; (ii) minimize the *total simulation time*, i.e., when all the tasks have finished processing; (iii) minimize the expected *latency*, namely the time the user has to wait for synchronizing clocks (it corresponds to the time the user waits for the beginning of the simulation).

6.2. Solution description

6.2.1. Encoding

The problem is addressed by means of multi-objective (MO) evolutionary optimization algorithm. We opted for a binary representation of the solution. Specifically, solutions are assignments of tasks to resources. Each of them is viewed as a binary matrix S, of size $n \times m$, with n tasks and m resources, where $S(i, j) = s_{i,j} = 1$ denotes that task i is assigned to resource j. For what said above, it stands that $\sum_{j=1}^{m} s_{i,j} = 1 \; \forall i = 1, \ldots, n$, namely a task is assigned exactly to one resource. Constraints could be set up on specific tasks (e.g., some tasks might be wanted to run on a specific resource, or on a resource with some features), which are dependent on the specific scenario to simulate.

6.2.2. Multi-objective approach

The objective functions are formulated as follows. The total cost associated with a solution (namely the first objective function) is the sum of cost of usage of the processing resources and the network resources. This is:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} s_{i,j} \cdot \left(R^j_{Cost_{proc}} \cdot \frac{T^i_{PR}}{R^j_{PC}} + R^j_{Cost_{Net}} \cdot T^i_{DER} \right). \quad (1)$$

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

The second objective function is about *simulation time*, computed as the total time to execute all the simulation tasks. It is given by the following equation:

$$\max_{1\le i\le n}\left(\sum_{j=1}^{m} s_{i,j}\cdot\left(\frac{T^i_{PR}}{R^j_{PC}}+\frac{T^i_{DER}}{R^j_{BC}}\cdot\frac{8}{3600}\right)\right) \quad (2)$$

made up of processing time and communication time.

Finally, latency is given as the maximum among latencies of used resources, which determines time to wait for synchronization:

$$\max_{1\le j\le m} s_{i,j}\cdot R^j_{latency}. \quad (3)$$

To evaluate the fitness of each solution, we employed a multi-objective approach to simultaneously minimize the objectives described above. Multi-objective optimization is an extremely useful tool to support the search of solutions that simultaneously optimize contrasting goals. For instance, objectives like time and cost minimization, or cost minimization and quality maximization are often contrasting goals implying that the improvement of one causes a degradation in the other. The goal of MO optimization is to devise a set of solutions, called *Pareto optimal solutions* or *Pareto front*, each of which assures a trade-off among conflicting objectives [50]. Moving among Pareto solutions means choosing to achieve a gain in some objectives at the expense of the others.

A very popular way for solving MO optimization problems, also used in this work, is by means of evolutionary algorithms (EAs), because of their ability of finding a set of trade-off solutions in one single run. We exploit three EAs for solving the formulated problem, as detailed in the Section 7.

6.2.3. Handling of multiple solutions

MOEAs use Pareto optimality; hence, a solution X is said to dominate another solution Y, if X is no worse than Y in all objectives and it is strictly better than Y in at least one objective. Given the output Pareto front, there may be several criteria to select one among the proposed solutions. If the user is interested in one specific objective, then he selects the solution with the highest fitness for that objective; however, the most typical case when MO is adopted is that trade-off solutions are of interest. On a Pareto front, a solution can be selected visually looking at the *knee point*. This can be defined in several ways; intuitively, a knee point is the Pareto-optimal point having the maximum reflex angle computed from its neighbors. We leave the decision on how to deal with the solution set to the engineer requiring the optimization, allowing it prioritizing one objective over another.

We adopt the simple approach in which the user is allowed to specify a set of weights for the pursued R objectives, $w = 1,\ldots,w_r,\ldots,w_R$ (so that $\sum_{r=1}^{R} w_r = 1$ and $0 \le w_r \le 1$), denoting the importance of each objective [51]. Let us denote the set of the R fitness values (one per objective) of a solution X as: $Y(X) = \{y_{1,x},\ldots,y_{r,x},\ldots,y_{R,x}\}$. We normalize these values in $[0, 1]$ over the entire Pareto front: $y'_{r,x} = \frac{y_{r,x}-\min_x(y_{r,x})}{\max_x(y_{r,x})-\min_x(y_{r,x})}$. The chosen solution X^* is the one with the maximum utility function value: $U(Y'(X)) = -\sum_{r=1}^{R} w_r\cdot y'_{r,x}$.

7. Experimental evaluation

This section explains the design of the experiment conducted to evaluate our approach to task allocation on the presented platform, namely: the investigated research questions, the scenarios we set up, the metrics and statistical tests used to assess confidently the achieved performance, and the adopted multi-objective algorithms.

Table 1
Requirements of the simulation tasks (in MB).

Task	T^i_{PR}	T^i_{DER}	Task	T^i_{PR}	T^i_{DER}
T^1	500	1	T^9	500	0.9
T^2	600	1.5	T^{10}	120	0.9
T^3	300	3	T^{11}	255	22
T^4	200	0.1	T^{12}	310	13
T^5	100	0.4	T^{13}	730	2.5
T^6	600	12	T^{14}	50	12
T^7	300	4	T^{15}	540	32
T^8	40	0.05	–	–	–

7.1. Experiment objectives

We run the MOEAs for the formulated problem to investigate validity, performance, and usefulness gained in simulation planning:

- *RQ1* (*Validation*): How does the optimal simulation perform compared to a random allocation strategy?
  This is a typical question performed as a preliminary "sanity check"; in fact, any intelligent computational search technique is expected to outperform random search unless there is something wrong in the formulation [52].
- *RQ2* (*MOEA solution quality comparison*): Which of the considered MOEAs provides better solutions for the optimal task allocation problem?
  This question focuses on the comparison among MOEAs according to common performance metrics regarding the goodness of the proposed Pareto solutions. We have adopted all the compared algorithms' implementation in our platform, so as to choose, for a given simulation task planning, the best performing one each time.
- *RQ3* (*Usefulness*): Does our approach yield useful insights into the trade offs between the contrasting objectives of the simulation setup?
  This question is to provide evidence, in realistic scenarios, that the approach can yield actionable insights on the choice of solutions able to respond to user-specified needs (e.g., decrease cost, simulation time, latency, or a combination of them). It is, therefore, an aspect of practical interest in that the final user deals directly with such trade-offs.

7.2. Scenarios

We experiment the simulation planning strategy considering two scenarios.

Scenario 1
The first scenario consists of 15 simulation tasks to be allocated to 5 resource, one of which is remote. Table 1 reports the task processing requirements (i.e., the task size), and the data exchange requirements of each task. In Table 2, there are the characteristics of the five resources available for running the simulation. Resource R^1 is the remote resource, having a much lower bandwidth available. The table also reports the costs per resource usage.

Scenario 2
The second scenario consists of 50 simulation tasks to be allocated to 10 resource, five of which are remote. Thus, this scenario represents a more stressed situation wherein the ratio of tasks over simulation is 5:1, and only five resources over 10 are local. Table 3 reports the task processing requirements, and the data exchange requirements of each task. In Table 4, there are the characteristics of the five resources available for running the simulation. Resource R^6 to R^{10} are the remote resources.

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

Table 2
Characteristics of the resources.

Resource	R^i_{PC} (MB/h)	R^i_{BC} (Mbps)	$R^i_{Cost_{proc}}$ ($/h)	$R^i_{Cost_{Net}}$ ($/MB)	$R^i_{latency}$ (ms)
R^1	200	10	1.00	0.60	200
R^2	250	100	2.10	0.40	20
R^3	500	100	3.50	0.70	18
R^4	1600	100	2.80	0.06	16.5
R^5	800	100	0.90	0.86	34

Table 3
Requirements of the simulation tasks (in MB).

Task	T^i_{PR}	T^i_{DER}	Task	T^i_{PR}	T^i_{DER}
T^1	500	1	T^{26}	120	1.9
T^2	600	1.5	T^{27}	125	0.8
T^3	300	3	T^{28}	255	26
T^4	200	0.1	T^{29}	290	15
T^5	100	0.4	T^{30}	310	12
T^6	600	12	T^{31}	730	2.5
T^7	300	4	T^{32}	50	12
T^8	40	0.05	T^{33}	540	38
T^9	500	0.9	T^{34}	240	4
T^{10}	120	1.9	T^{35}	740	3.5
T^{11}	255	22	T^{36}	50	12
T^{12}	310	13	T^{37}	540	6
T^{13}	730	2.5	T^{38}	470	8
T^{14}	50	12	T^{39}	420	1.5
T^{15}	540	32	T^{40}	40	0.9
T^{16}	500	1	T^{41}	180	2.5
T^{17}	600	1.5	T^{42}	310	0.4
T^{18}	300	3	T^{43}	200	15
T^{19}	200	0.1	T^{44}	30	0.2
T^{20}	100	0.4	T^{45}	105	4.5
T^{21}	600	12	T^{46}	530	6.5
T^{22}	300	4	T^{47}	420	2.0
T^{23}	40	0.05	T^{48}	380	1.5
T^{24}	500	0.9	T^{49}	450	21.0
T^{25}	500	0.9	T^{50}	35	1.0

7.3. Metrics

This section describes the evaluation metrics and statistical tests by which the algorithms are compared to each other. In order to provide a quantitative assessment, we employ two well-known indicators: the *Hypervolume* (HV) [53], and the *generational distance* (GD) [54]. The HV is one of the most accurate indicators; it calculates the volume, in the objective space, covered by members of a non-dominated set of solutions from an algorithm of interest. The larger the volume, the more it covers of the non-dominated solution space, hence the better the algorithm. The GD is the average distance between the set of solutions S, and the reference front RF. The latter is computed by considering the union of the reference fronts of the approaches compared. As evolutionary algorithms are of stochastic nature, we perform 30 independent runs for each combination of algorithm and scenario. For each of them, the HV and GD indicators are computed and compared by means of the Friedman hypothesis test. The Friedman test is a non-parametric test for repeated-measures to detect if at least one difference among compared algorithms. Then, to detect which algorithms are different, we run a *post hoc* analysis. When we compare all the algorithms with each other (like in RQ2, MOEAs comparison), we adopt the *Nemenyi* test, which is a powerful test for pairwise comparisons after a non-parametric ANOVA [55]. When we compare all the algorithms against one control algorithm (like in RQ 1), the Bonferroni–Dunn test is used as post-hoc, which is more powerful than the Nemenyi test in such a case [55].

Table 5
Configurations of MOEA algorithms.

Configuration	Population size	Generations
Very small (VS)	50	20,000
Small (S)	100	10,000
Medium (M)	200	5,000
Large (L)	500	2,000
Very large (VL)	1000	1,000

7.4. Evolutionary algorithms and parameters tuning

We include three different metaheuristics. We consider: NSGA-II, SPEA2, and MOCELL. They all are well-known evolutionary algorithms (EAs), which are among the most popular metaheuristics for solving MO problems. Both NSGA-II and SPEA2 use the same general scheme to look for solutions, but they differ one each other in the mechanism used to keep a diverse approximated Pareto front. MOCell is a structured EA that includes an external archive to store the nondominated solutions. It uses the same evolutionary operators as NSGA-II and SPEA2. Evolutionary operators are selected accordingly with the binary codification of the problem, namely, the binary tournament selection, a bit flip mutation operator, and single point crossover operator. To tune the population size and number of generations for each algorithm, we have followed a procedure similar to [51,52]; a maximum number of fitness evaluations for all the algorithms is set up (1,000,000): keeping this value fixed (so as to ensure that all require approximately the same computational effort), we varied the population size and number of generations as in Table 5. Each algorithm is run 30 times with each configuration; results are compared in terms of HV and GD by the Friedman test, and then Nemenyi test, with a confidence level of 95%, and the best configuration is selected. The configuration with the highest HV is preferred; if there is no configuration

Table 4
Characteristics of the resources.

Resource	R^i_{PC} (MB/h)	R^i_{BC} (Mbps)	$R^i_{Cost_{proc}}$ ($/h)	$R^i_{Cost_{Net}}$ ($/MB)	$R^i_{latency}$ (ms)
R^1	300	54	1.00	0.06	28
R^2	150	100	1.10	0.40	20
R^3	1000	100	1.50	0.70	18
R^4	1200	100	2.80	0.06	16
R^5	200	100	0.90	0.16	38
R^6	200	10	0.50	0.60	200
R^7	250	10	0.90	0.45	210
R^8	500	8	2.30	0.65	180
R^9	1600	5	2.00	0.05	165
R^{10}	700	8	0.90	0.90	140

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

Table 6
Parameters setting.

	NSGA	SPEA2	MOCELL
Scenario 1			
Generations	10,000	10,000	10,000
Population size	100	100	100
Scenario 2			
Generations	20,000	10,000	10,000
Population size	50	100	100
Both Scenarios: operators setting			
Selection	Binary Tournament	Binary Tournament	Binary Tournament
Crossover (prob.)	Single point (0.9)	Single point (0.9)	Single point (0.9)
Mutation (prob.)	Bit Flip (1/L)	Bit Flip (1/L)	Bit Flip (1/L)
Archive size	–	–	100

Table 7
p-value of the Bonferroni–Dunn test comparing the random search with MOEAs. The former is statistically worse with confidence of (1-*p*-value)∗100%.

	NSGAII	SPEA2	MOCELL
Scenario 1	1.22e−9	2.31e−09	6.32e−09
Scenario 2	1.85e−08	3.02e−08	2.82e−08

statistically better than the others, the configuration with the lowest GD (again at 95% of confidence) is selected. If no configuration is significantly better than the others in both HV and GD measure, the *medium* configuration is selected. This evaluation step is performed for both scenarios, resulting in a total number of runs of 2 scenarios ∗ 3 algorithms ∗ 5 configurations ∗ 30 repetitions = 900. Results of this tuning and the other configuration parameters for each MOEA in both scenarios are in Table 6. Algorithms implementation and experimental settings have been carried out by using jMetal, an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for multi-objective optimization problems.[4]

7.5. Analysis of results

This section presents the results with respect to research questions 1–3.

Results for RQ1—Validation: the first research question is aimed at establishing if the proposed strategy is worth with respect to an unplanned random selection. The latter is implemented in a way to satisfy the same constraints formulated for our strategy. Considering the two scenarios and the 3 MOEA algorithms (hence 6 cases), the random search has been, in every case, statistically worse, to a large extent, than any MOEA algorithm for both quality indicators. Results of the Bonferroni–Dunn test referring to the comparison of Random search with the three MOEAs are in Table 7 over 30 runs. Because of very low solution quality (HV is always less than 0.1), the random search is not considered a possible alternative to a MOEA, but just as a means to validate the choice of adopting a MOEA. We do no longer consider it in the next research questions.

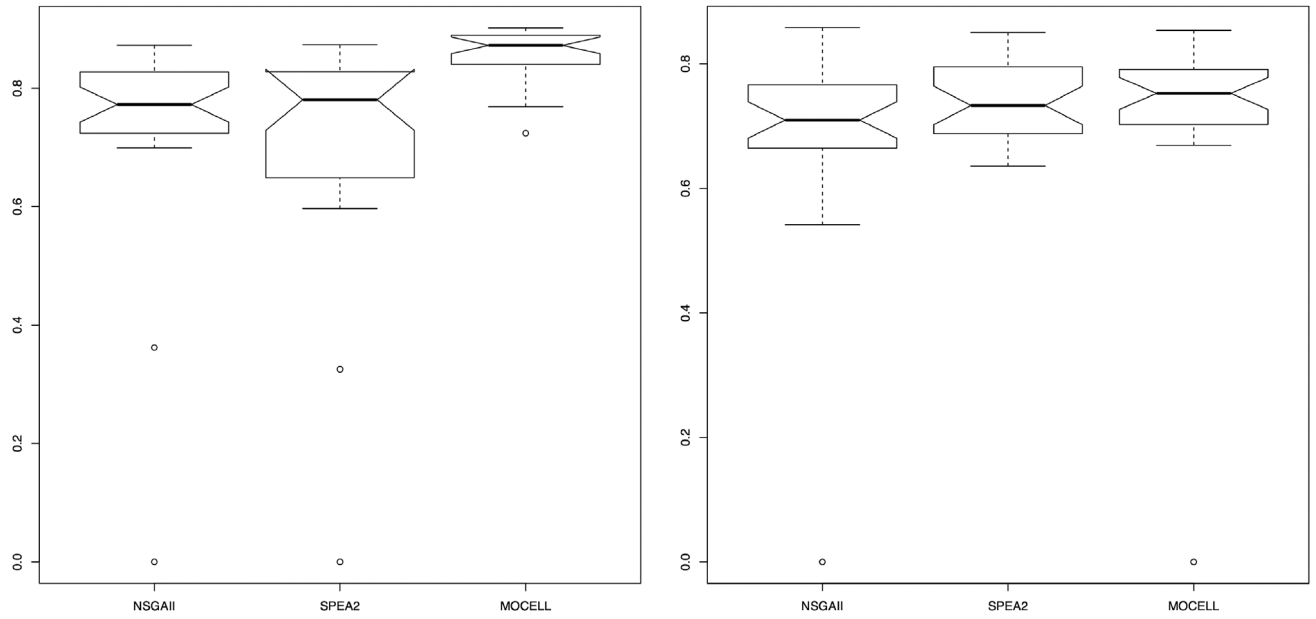
Results for RQ2—MOEA quality comparison: MOEAs are compared with respect to the HV and GD indicators. We use the best configuration of each algorithm as in Table 6. Figs. 7–8 show the notched box plots for both scenarios and both indicators (HV and GD), representing the distribution of data and the extent of the difference among algorithms. Specifically, given the 30 runs of the algorithm on the *x*-axis and the metric on the *y*-axis (i.e., HV or GD), a notched box plot reports: the median value over 30 runs, denoted by the line in the middle; the semi-interquartile range (also known as IQR, including the values from 25 to 75 percentile), denoted by the entire box; the extreme values, denoted by the whiskers, obtained by adding 1.5 times the IQR to the 75 percentile and subtracting 1.5 times the IQR from the 25 percentile; the confidence interval around the median denoted by the notches in the box: if two boxes' notches, referring to two algorithms, do not overlap, there is strong evidence (at 95% confidence) that their medians differ.

Referring to results of Scenario 1, the box plots report that the median HV of MOCELL is 0.854, while it is 0.743 for NSGA-II and 0.682 for SPEA2. The GD median was: 1.17e−02 for NSGA-II, 1.51e−02 for SPEA2, 7.76e−03 for MOCELL. In Scenario 2, the median values of HV are: 0.628 for NSGA-II, 0.739 for SPEA2, and 0.684 for MOCELL; values of GD are: 1.44e−02 for NSGA-II, 9.16e−03 for SPEA2, 1.55e−02 for MOCELL. According to the outcomes of the Friedman test, there is at least a difference between algorithms in scenario 1 ($p\text{-value} < 0.05$ for both HV and GD), while there is not significant difference in scenario 2 (at 95% of confidence). The Nemenyi test on data of scenari of 1 confirms that MOCELL is the best algorithm in that scenario, for both HV and GD indicators ($p\text{-value} < 0.05$). Considering both scenarios, indications provided tend to suggest a slightly better behavior of MOCELL, whose solutions will thus be considered to argue about the next research question. Finally, regarding the computational performance of the three MOEAs, the mean execution time to get a solution in the two scenarios were: 482 ms and 1349 ms in scenario 1 and 2, respectively, with NSGA-II; SPEA2 took 9175 ms (scenario 1) and 10 849 ms (scenario 2), with relevantly worse performance; MOCELL took 342 ms (scenario 1) and 1280 ms (scenario 2), marking the best performance even in terms of computational time.*Results for RQ3—Usability*: To answer RQ3 we analyzed the Pareto fronts produced by our approach in order to identify useful trade offs among different goals and to discover suitable knee points.

Scenario 1:

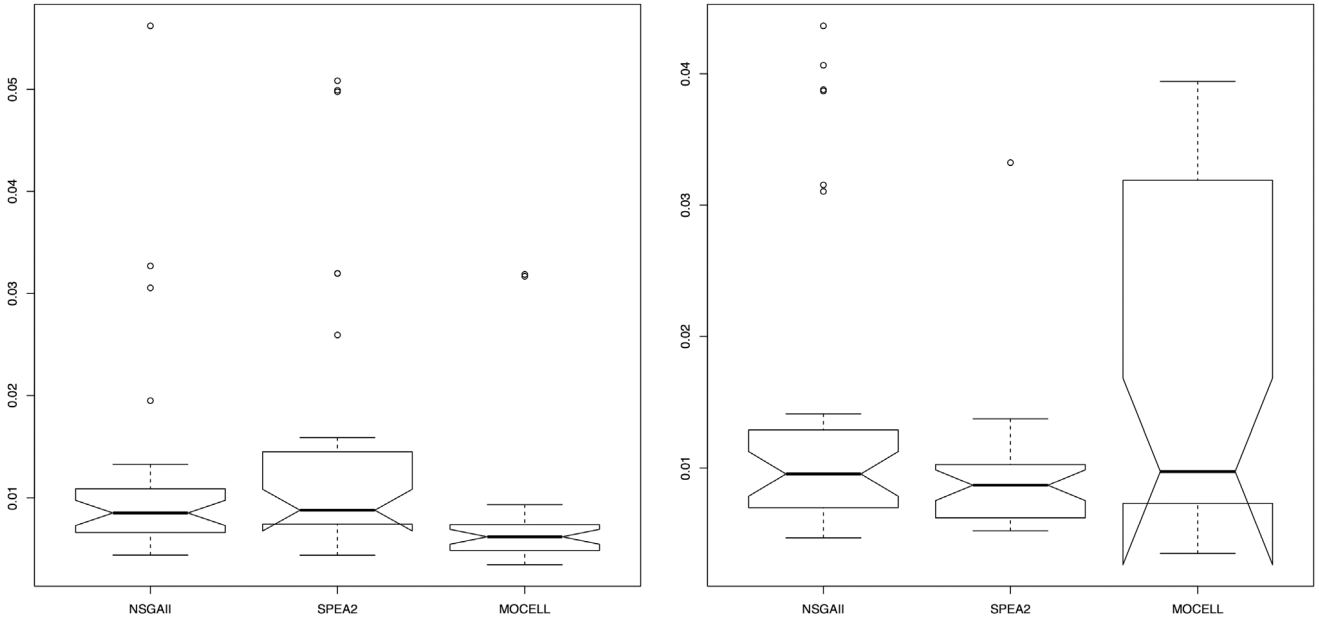
Fig. 9 shows the Pareto front of one of the solution sets along all the dimensions provided by MOCELL, the best performing algorithm. This is the fitness of the non-dominated solutions among which a user can choose. Points in the plot clearly show the trade-off between the dimensions, and, in particular, between the cost and time dimension (it can be noted how to an increase of cost corresponds a decrease of time and vice versa in the *x*–*y* plan). Engineers could be interested in trade-offs among all the objectives, or in just one or two out of the three objectives. As an example of solutions satisfying different trade-offs, let us first suppose that the user is interested only in cost minimization. Fig. 10(a) reports a possible scheduling of tasks that yields the

[4] jMetal is freely available at http://jmetal.sourceforge.net/.



(a) Scenario 1. (b) Scenario 2.

Fig. 7. Boxplots for the HV quality indicator of the 2 scenarios.



(a) Scenario 1. (b) Scenario 2.

Fig. 8. Boxplots for the GD quality indicator of the 2 scenarios.

minimal cost (1617$), but at the expense of total simulation time (which is 2.45 h) and latency (200 ms). In contrast, Fig. 10(b) reports a solution that yields the shortest total simulation time (1.58 h), but with a much higher cost (5620$). One could be even be interested only in latency, in which case all the tasks would be allocated to the machine with lowest latency, regardless the cost and response time (of course, in this case, there is no need of an optimization algorithm).

In the most typical scenario, the user wishes to balance among more objectives. From the provided Pareto front, he can either select the *knee point* visually from the 3D plot of Fig. 9 (or any 2D projection if only two objectives are of interest), or can use an utility function. The utility function we defined previously assigns weights to objectives. Fig. 10(c) reports the solution in the case of user specifying the following weights (i.e., importance) for the objectives: 0.4 for cost, 0.4 for time, 0.2 for latency. Results show that, in this case, the cost is 2310$ (a bit worse than the optimal-cost solution) and total simulation time is 1.73 h (a bit worse than the optimal-time solution); however the point is a good compromise, as the cost is much better than the one of the optimal-time solution and the time is much better than the one obtained under the optimal-cost solution. Several other solutions from the front could be considered, depending on how the user rates the importance of each objective.

Scenario 2:

Insights on the usefulness of such a kind of optimization are, of course, the same in Scenario 2 as in Scenario 1; we hereafter reports synthetically the results in the case of Scenario 2 for optimal-cost,

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

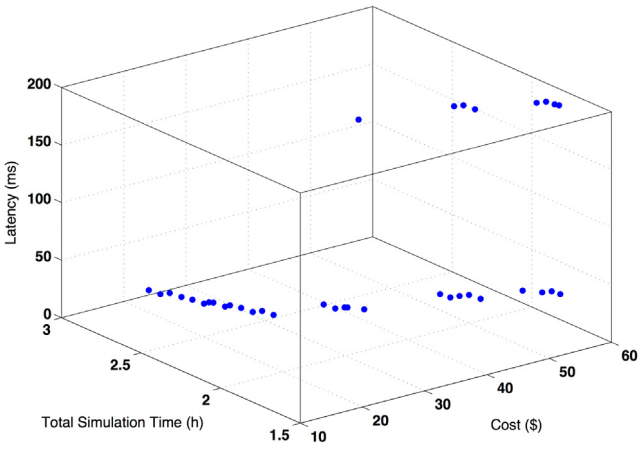


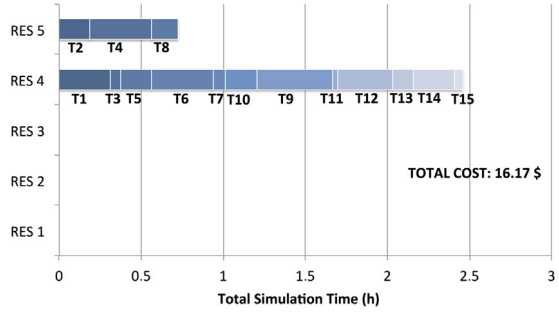
Fig. 9. Scenario 1. Pareto front.

optimal-time, and weighted cost and time solutions. In the case of an optimal-cost solution, i.e., giving weight 1 to cost and 0 to the others in the utility function, the suggested allocation has a cost of 222,41$, a total simulation time of 14.35 h, and a latency of 220 ms. If one is interested in the minimum total simulation time, the suggested solution will cost 333,94$, but yielding a simulation time of 8.47 h (latency is still 220 ms). It means that an extra cost of about 111$ with respect to solution 1 (i.e., +50%) will yield a time saving of 5.88 h (i.e., about −40%). A trade-off solution with the same weights as in Scenario 1 (0.4, 0.4 and 0.2, for cost, time and latency, respectively) yields a simulation planning with: 225,37$, 10.37 h, and 220 ms of latency. With respect to the optimal-cost solution, this adds just an extra cost of 3$ (i.e., +1.3%), but allows saving 4 h (−27%); with respect to the optimal time solution, it worsens the total time by almost 2 h (+22%) but allows saving 108,57$ (−32.5%).

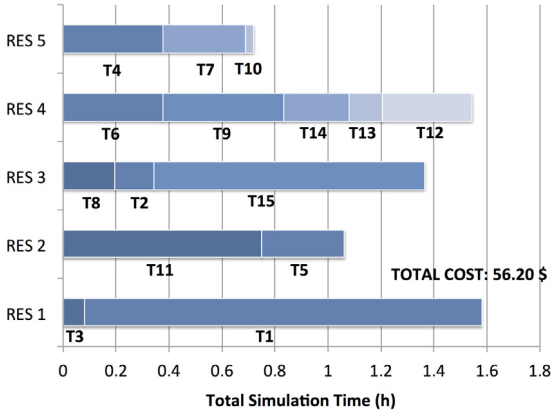
These solutions are just few examples of how the optimal planning can help in hybrid simulation problems. Pareto front solutions propose actionable conclusions for software engineers planning distributed high-intensive simulation. Without the benefit of the insights provided by such an analysis, one would naturally be tempted to look for the minimum-cost or minimum-time allocation, but no suitable trade-off alternative would be considered. Of course, it us up to engineer to make the final decision concerning overtime planning depending on the importance he gives to the objectives and the relative (percentage) gains obtained in the solutions. Using Pareto fronts allows answering questions like "how much the additional cost spent in simulation is effective for saving time?" or "how much should I spend to reduce the time by *x*%?". Such answers provide valuable tools for decision support to the simulation platform manager and engineers using it for their simulation task.

7.6. Threats to validity

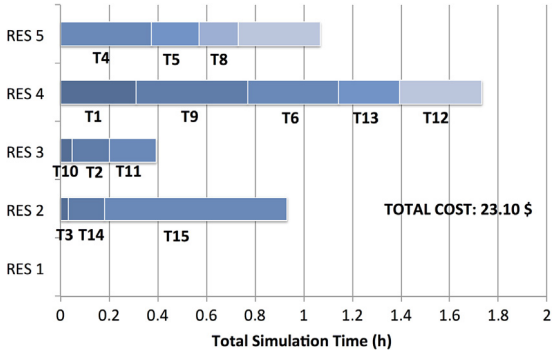
We have evaluated the proposed approach by setting up and executing an empirical study. Such kind of experimentation is naturally subject to "threats", namely to factors that can bias the analysis and need to be taken into account when interpreting the results. In our case, results refer to a specific set of MOEAs on two specific scenarios we used as illustrative examples. As for metaheuristics, although we have tuned the population size and the number of generations by trying several combinations and have adopted common values for (crossover, selection, mutation) operators (and their probabilities), a deeper study varying all the parameters of a metaheuristic could help tuning and improving



(a) A solution optimizing cost.



(b) A solution optimizing time.



(c) A solution optimizing a trade-off between cost, time, latency.

Fig. 10. Example solutions.

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

the obtained results. On one hand, choices different from ours about the parameters of the compared MOEAs could yield different results. On the other hand, the solution as designed offers the possibility to instantiate the concrete approach by using other MOEAs besides the one adopted in this paper. Regarding the proposed scenarios, we have evaluated the approach on the two typical sample scenarios we encounter in VTS systems. The characterization of tasks and resources used in such scenarios could, indeed, be affected by uncertainty (e.g., task processing size or data exchange requirements are estimates of the real value, potentially affected by a margin of error); a deeper analysis will be required to figure out to what extent the proposed algorithms are robust with respect to uncertain input parameters (namely, how uncertainty in the inputs propagates to the results – a topic we are going to investigate by Montecarlo simulation approaches [56]). Additionally, while these scenarios are very representative of what we experience in the practice when using distributed simulation, we cannot claim that our results generalize to all possible scenarios. An exhaustive empirical studies to improve generality will be subject of our future work.

8. Conclusions

In this work, we presented an open-source middleware for implementing local and distributed testbeds, that can be used to simulate large-scale systems in mission-critical scenarios, like air and vessel traffic control systems. The middleware is being implemented in the context of the DISPLAY project, whose main aim is to implement distributed and hybrid simulation platform for efficiently engineering ATC and VTS systems of the next generation. Regarding the architectural solution, an HLA-based platform and the adoption of the cloud paradigm allowed us to solve tricky issues in terms of interoperability among heterogeneous entities and integration of entities of different types (e.g., emulated, simulated, real). Regrading the efficiency of the platform usage – a requirement of primary importance for our industrial partner in the project – the best exploitation of resources is targeted by means of optimal planning of tasks allocation.

Overall, the presented solution, that is going to be exploited with real systems, is expected to lead to (i) a significant reduction of costs in all the system development life-cycle, and to (ii) an increased dependability and security, due to better integration and system test practices. It can bring a deep innovation into industrial design and development processes, especially with respect to the next generation of critical systems, constituted by very large integrations of desperate systems due to an increased exploitation of facilities from various several domains. The presented platform is subject to continuous improvement within the context of the project: in the next future, further algorithms for scalability with respect to the number of simulated scenarios and their increase of complexity will be explored. A tight interaction with the industrial partner is already in act, with the main objective of introducing such a kind of optimized solutions in the daily industrial practice, integrated with practices already in act for requirement verification [57–59], development [60], testing and analysis [61–63], and maintenance [64]. We also foresee to validate and refine the platform by looking at other domains of mission-critical systems, such as military and civil applications, which can highlight additional different requirements for hybrid distributed simulation.

Acknowledgment

This work has been partially supported by the Italian Ministry for Education, University, and Research (MIUR) under Project PON02_00485_3487784 "DISPLAY" of the public–private laboratory "COSMIC" (PON02_00669).

References

[1] D. Cotroneo, R. Pietrantuono, S. Russo, K. Trivedi, How do bugs surface? A comprehensive study on the characteristics of software bugs manifestation, J. Syst. Softw. 113 (2016) 27–43.
[2] D.G. Cavezza, R. Pietrantuono, S. Russo, J. Alonso, K.S. Trivedi, Reproducibility of environment-dependent software failures: An experience report, in: 2014 IEEE 25th International Symposium on Proc. of the Software Reliability Engineering, ISSRE, 2014, pp. 267–276.
[3] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), Federate Interface Specification, Std 1516.2-2000. New York: Institute of Electrical and Electronics Engineers, Inc.
[4] S. Feng, Y.D. Yuanchang, Z.X. Meng, Remodeling traditional RTI software to be with PaaS architecture, in: Proc. of the 3rd IEEE Int. Conf. on Computer Science and Information Technology, ICCSIT, 2010, pp. 511–515.
[5] M. Clune, P. Mosterman, C. Cassandras, Discrete event and hybrid system simulation with simevents, in: Proc. of the 8th International Workshop on Discrete Event Systems, July 2006, pp. 386–387.
[6] E. Kofman, M. Lapadula, E. Pagliero, PowerDEVS: A DEVSbased environment for hybrid system modeling and simulation, school of Electronic Engineering, Universidad Nacional de Rosario, Tech. Rep. LSD0306, 2003, Available at: http://www.fceia.unr.edu.ar/~kofman/files/lsd0306.pdf.
[7] H. Zheng, Operational semantics of hybrid systems (Ph.D. dissertation), Adviser-Edward A. Lee, Berkeley, CA, USA, 2007.
[8] M. Wetter, P. Haves, A modular building controls virtual test bed for the integration of heterogeneous systems, in: Proc. of the 3rd National Conference of IBPS, SimBuild, July 2008, pp. 69–76.
[9] F. Bouchhima, G. Nicolescu, E.M. Aboulhamid, M. Abid, Generic discrete-continuous simulation model for accurate validation in heterogeneous systems design, J. Microelectron. 38 (6–7) (2007) 805–815.
[10] A. Borshchev, A. Filippov, From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools, in: Proc. of the 22nd International Conference on the System Dynamics Society, 2004, pp. 25–29.
[11] R. Aversa, B. Di Martino, M. Ficco, S. Venticinque, A simulation model for localization of pervasive objects using heterogeneous wireless networks, Simul. Model. Pract. Theory 19 (8) (2011) 1758–1772.
[12] J. Krozel, N. Doble, Simulation of the national airspace system in inclement weather, in: Modeling and Simulation Technologies Conference, 2007, pp. 20–23.
[13] A. Agogino, K. Tumer, Regulating air traffic flow with coupled agents, in: Proc. of the 7th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, Vol. 2, 2008, pp. 535–542.
[14] V. Gorodetsky, O. Karsaev, V. Samoylov, V. Skormin, Multi-agent technology for air traffic control and incident management in airport airspace. in: Proc. of the Int. Workshop on Agents in Traffic and Transportation, 2008, pp. 119–125.
[15] David Sislak, Premysl Volf, Michal Jakob, Michal Pechoucek, Distributed platform for large-scale agent-based simulations, in: Agents for Games and Simulations, Springer, 2009, pp. 16–32.
[16] S.Y. Lim, T.G. Kim, Hybrid modeling and simulation methodology based on DEVS formalism, in: Sunner Computer Simulation, 2001, pp. 188–193.
[17] A. Borshchev, Y. Karpov, V. Kharitonov, Distributed simulation of hybrid systems with AnyLogic and HLA, Future Gener. Comput. Syst. 18 (6) (2002) 829–839.
[18] M. Ficco, G. Avolio, F. Palmieri, A. Castiglione, An HLA-based framework for simulation of large-scale critical systems, Concurr. Comput.: Pract. Exper. (2015) http://dx.doi.org/10.1002/cpe.3472.
[19] M. Ficco, G. Avolio, L. Battaglia, V. Manetti, Hybrid Simulation of Distributed Large-Scale Critical Infrastructures, in: Proc. of the Int. Conf. on Intelligent Networking and Collaborative Systems, September 2014, pp. 616–621.
[20] S. Smanchat, K. Viriyapant, Taxonomies of workflow scheduling problem and techniques in the cloud, Future Gener. Comput. Syst. 52 (2015) 1–12.
[21] E.N. Alkhanak, S.P. Lee, S.U.R. Khan, Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities, Future Gener. Comput. Syst. 50 (2015) 3–21.
[22] Wei-Jen Wang, Yue-Shan Chang, Win-Tsung Lo, Yi-Kang Lee, Adaptivescheduling for parallel tasks with QoS satisfaction for hybrid cloud environments, J. Supercomput. (2013) http://dx.doi.org/10.1007/s11227-013-0890-2.
[23] J. Moses, R. Iyer, R. Illikkal, S. Srinivasan, K. Aisopos, Shared Resource Monitoring and Throughput Optimization in Cloud-Computing, in: Proc. of the Int. Conf. on Datacenters, Parallel & Distributed Processing Symposium, IPDPS, May 2011, pp. 1024–1033.
[24] S. Ghanbari, M. Othman, A priority based job schedulingalgorithm in cloud computing, in: Proc. of the Int. Conf. on Advances Science and Contemporary Engineering, Vol. 50, 2012, pp. 778–785.
[25] H. Lawrance, S. Silas, Efficient QoS based resource scheduling using PAPRIKA method for cloud computing, in International Journal of Engineering Science and Technology, IJEST, Vol. 5, no. 3, March 2013, pp. 638–643.
[26] Jianfeng Zhao, Wenhua Zeng, Min Liu, Guangming Li, Min Liu, Multiobjective optimization model of virtual resources scheduling under cloud computing and it's solution, in: Proc. of the Int. Conf. on Cloud and Service Computing, CSC, December 2011, pp. 185–190.
[27] R. Raju, R.G. Babukarthik, D. Chandramohan, P. Dhavachelvan, T. Vengattaraman, Minimizing the make span using Hybrid algorithm for cloud computing, in: Proc. of the IEEE 3rd Int. Conf. on Advance Computing Conference, IACC, February 2013, pp. 957–962.
[28] R. Gogulan, A. Kavitha, U. Karthick Kumar, An multiple pheromone algorithm for cloud scheduling with various QoS requirements, Int. J. Comput. Sci. Issues 9 (3) (2012) 232–254.

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022

[29] A.V. Lakra, D.K. Yadav, Multi-objective tasks scheduling algorithm for cloud computing throughput optimization, Procedia Comput. Sci. 48 (2015) 107–113.
[30] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds, Future Gener. Comput. Syst. 48 (2015) 1–18.
[31] Jinn-Tsong Tsai, Jia-Cen Fang, Jyh-Horng Chou, Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm, Comput. Oper. Res. 40 (12) (2013) 3045–3055.
[32] Mihaela-Andreea Vasile, Florin Pop, Radu-Ioan Tutueanu, Valentin Cristea, Joanna Koodziej, Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing, Future Gener. Comput. Syst. 51 (2015) 61–71.
[33] Li Xu, Zhibin Zeng, Xiucai Ye, Multi-objective optimization based virtual resource allocation strategy for cloud computing, in: Proc. of the IEEE/ACIS 11th Int. Conf. on Computer and Information Science ICIS, 2012, pp.56–61.
[34] Y. Ding, X. Qin, L. Liu, T. Wang, Energy efficient scheduling of virtual machines in cloud with deadline constraint, Future Gener. Comput. Syst. 50 (2015) 62–74.
[35] S. Selvarani, G. SudhaSadhasivam, Improved cost-based algorithm fortask scheduling in Cloud computing, in: Proc. of the IEEE Int. Conf. on Computational Intelligence and Computing Research, ICCIC, 2010, pp. 1-5.
[36] S. Kianpisheh, N.M. Charkari, M. Kargahi, Reliability-driven scheduling of time/cost-constrained grid workflows, Future Gener. Comput. Syst. 55 (2016) 1–16.
[37] R. Pietrantuono, S. Russo, K.S. Trivedi, Software reliability and testing time allocation: An architecture-based approach, IEEE Trans. Softw. Eng. 36 (3) (2010) 323–337.
[38] G. Carrozza, R. Pietrantuono, S. Russo, Dynamic test planning: a study in an industrial context, Int. J. Softw. Tools Technol. Trans. 16 (15) (2014) 593–607.
[39] poRTIco RTI tool, available at: http://www.porticoproject.org/index.php?title=Main_Page. Last update April 2013.
[40] J. Ahrenholz, Comparison of CORE network emulation platforms, in: Proc. of IEEE Military Communications Conference, MILCOM, 2010, pp. 864-869.
[41] C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory, IEEE Trans. Comput. http://dx.doi.org/10.1109/TC.2015.2389952.
[42] OpenNebula, An user-driven cloud management platform for sysadmins and devops, Available at: http://opennebula.org/.
[43] Host sFlow, Host sFlow: Data center wide server performance monitoring, 2014. available at http://host-sflow.sourceforge.net/relatedlinks.php.
[44] Y. Xue, T. Busch, M. Gacek, H. Neema, G. Karsai, J. Sztipanovits, A model-based integration of network emulation with hla-based heterogeneous simulation environments. Technical Report ISIS-10-107, 2010, Available at: http://www.isis.vanderbilt.edu/node/4396.
[45] W. Zhang, L. Feng, J. Hu, Y. Zha, An approach to service provisioning of HLA RTI as web services, in: Proc. of the 7th Int. Conf. on Asia Simulation Conference, 2008, pp. 1-6.
[46] W. Zebin, W. Huizhong, L. Weiqing, Z. Xu, Extending distributed simulation's run-time infrastructure with web services, in: Proc. of the IEEE Int. Conf. on Automation and Logistics, 2007, pp. 1528–1532.
[47] AIS—Automatic identification system, available at: https://www.itu.int/rec/R-REC-M.1371/en.
[48] M.1317: Considerations for sharing between systems of other services operating in bands allocated to the radionavigation-satellite and aeronautical radionavigation services and the global navigation satellite system (GLONASS-M), available at: https://www.itu.int/rec/R-REC-M.1317-0-199710-W/en.
[49] R. Shah, B. Veeravalli, M. Misra, Estimation based load balancing algorithm for data-intensive heterogeneous grid environments, in: Proc. of the High Performance Computing, in: LNCS, vol. 4297, 2006, pp. 72–83.
[50] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[51] M. Ficco, R. Pietrantuono, S. Russo, Using multi-objective metaheuristics for the optimal selection of positioning systems, Soft Comput. (2015) 1–24.
[52] F. Ferrucci, M. Harman, J. Ren, F. Sarro, Not going to take this anymore: multi-objective overtime planning for software engineering projects, in: Proc. of the Int. Conf. on Software Engineering, ICSE '13, 2013, pp. 462-471.
[53] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.
[54] Van Veldhuizen, G.B. Lamont, Multiobjective evolutionary algorithm research: A history and analysis, Tech. Rep. TR-98-03, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Institute Technology, Wright-Patterson, AFB, OH, 1998, Available at: http://delta.cs.cinvestav.mx/~ccoello/EMOO/veldhuizen98.ps.gz.
[55] Janez Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[56] I. Meedeniya, A. Aleti, L. Grunske, Architecture-driven reliability optimization with uncertain model parameters, J. Syst. Softw. 85 (10) (2012) 2340–2355.
[57] G. Gigante, F. Gargiulo, M. Ficco, A semantic driven approach for requirements verification, Stud. Comput. Intell. 570 (2015) 427–436.
[58] F. Gargiulo, G. Gigante, M. Ficco, A semantic driven approach for requirements consistency verification, High Perform. Comput. Netw. 8 (3) (2015) 201–211.
[59] G. Zazzaro, G. Gigante, E. Zaccariello, M. Ficco, B. Di Martino, Supporting development of certified aeronautical components by applying text analysis techniques, in: Proc. of the 8th Int. Conf. on Complex, Intelligent, and Software Intensive Systems, CISIS'14, Birmingham, UK, 2014, pp. 602-607.
[60] G. Carrozza, M. Faella, F. Fucci, R. Pietrantuono, S. Russo, Engineering air traffic control systems with a model-driven approach, IEEE Softw. 30 (3) (2013) 42–48.
[61] G. Carrozza, M. Faella, F. Fucci, R. Pietrantuono, S. Russo, Integrating MDT in an industrial process in the air traffic control domain, in: Proc. of the Software Reliability Engineering Workshops, ISSREW, 2012 IEEE 23rd International Symposium on, 2012, pp.225-230.
[62] G. Carrozza, M. Cinque, U. Giordano, R. Pietrantuono, S. Russo, Prioritizing correction of static analysis infringements for cost-effective code sanitization, in: Proc. of the Second International Workshop on Software Engineering Research and Industrial Practice, SER&IP'15, 2015, pp. 25-31.
[63] D. Cotroneo, R. Pietrantuono, S. Russo, Combining operational and debug testing for improving reliability, IEEE Trans. Reliab. 62 (2) (2013) 408–423.
[64] G. Carrozza, R. Pietrantuono, S. Russo, Defect analysis in mission-critical software systems: a detailed investigation, J. Softw. Evol. and Proc. 27 (1) (2015) 22–49.

Massimo Ficco is Assistant Professor at the Department of "Ingegneria Industriale e dell'Informazione" of the Second University of Naples (SUN). He received the degree in Informatics Engineering in 2000 from the University of Naples "Federico II", and his Ph.D. in "Information Engineering" from the University of "Parthenope" in 2010. From 2000 to 2010, he was senior researcher at the Italian University Consortium for Computer Science (CINI). From 2004, he taught master courses in "Software Reliability and Security", "Software Engineering", and "Data Base", "Software Programming". His current research interests include security and reliability of critical infrastructure, cloud computing, and mobile computing. Massimo Ficco has been involved in national and EU funded research projects in the area of security and reliability of critical infrastructures and cloud computing, in which he has covered different positions of responsibility.

Beniamino Di Martino is Full Professor of Information Systems at the Second University of Naples (Italy) since 2005, and vice-Head of Dip. Di Ingegnaria Industriale e dell' Informazione. From 1994 till 1998 he was Researcher at the Institute for Software Technology and Parallel Systems at the University of Vienna (Austria). In 1998 he moved at the Second University of Naples—Assistant Professor till 2002, Associate Professor till 2005. He is author of 8 international books and more than 220 publications in international journals and conferences. He has been Project Coordinator of the FP7-ICT-2010-256910 Project mOSAIC—Open-Source API and Platform for Multiple Clouds. He has been vice Chair of the Executive Board of the IEEE CS Technical Committee on Scalable Computing, and member of the IEEE Working Group for the IEEE P3203 Standard on Cloud Interoperability, of the IEEE Intercloud Testbed Initiative, of the Cloud Standards Customer Council, of the Cloud Computing Experts' Group of the European Commission.He acts as Co-Chair of the Nomination Committee for the IEEE "Award of Excellence in Scalable Computing".

Roberto Pietrantuono, Ph.D., IEEE Member, received the B.S. and M.S. degrees in computer engineering in 2003 and 2006, respectively, from the Federico II University of Naples, Italy. He received the Ph.D. degree in computer and automation engineering from the same university in 2009. He is currently a post-doc researcher at the same university. His main research interests are in the area of software engineering of critical systems and software reliability evaluation. In this context, he collaborated in several national and international projects. He published in the field of software testing, dependability assessment, and reliability modeling and analysis.

Stefano Russo is Professor of Computer Engineering at the Federico II University of Naples, where he teaches Software Engineering and Distributed Systems, and leads the distributed and mobile systems research group (www.mobilab.unina.it). He co-authored over 140 papers in the areas of software engineering, middleware technologies, software dependability, mobile computing. He is Associate Editor of the IEEE Transactions on Services Computing.

Please cite this article in press as: M. Ficco, et al., Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.01.022