Arab J Sci Eng (2017) 42:3357–3373
DOI 10.1007/s13369-017-2497-z

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

A Generic Framework for Building Heterogeneous Simulations of Parallel and Distributed Computing Systems

Taner Dursun[1] · **Hasan Dağ**[2]

Received: 17 September 2016 / Accepted: 7 March 2017 / Published online: 20 March 2017

Abstract There have been many systems available for parallel and distributed computing (PDC) applications such as grids, clusters, super-computers, clouds, peer-to-peer and volunteer computing systems. High-performance computing (HPC) has been an obvious candidate domain to take advantage of PDC systems. Most of the research on HPC has been conducted with simulations and has been generally focused on a specific type of PDC system. This paper, however, introduces a general purpose simulation model that can be easily enlarged for constructing simulations of many of the most well-known PDC system types. Although it might create a new vision for research activities in the simulation community, current simulation tools do not provide proper support for cooperation between software working in real-time and simulation time. In this paper, thus, we also present a promising approach for constructing hybrid simulations that offers great potential for many research areas. As a proof of concept, we implemented a prototype for our simulation model. Then, we are able to rely on this prototype to build simulations of various PDC systems. Thanks to hybrid simulation support of our model, we are able to combine and manage the simulated PDC systems with our previously developed policy-based management framework in simulation runs.

Keywords Parallel and distributed computing · High-performance computing · Policy-based management · Simulation

1 Background

Parallel and distributed computers, considered as a type of multiple instruction multiple data (MIMD) in Flynn's computer classification [1], have been used for decades as high-performance computing architectures, such as super-computers, and clusters. As a sub-type of PDCs, thanks to the increasing speed of the Internet and the improved performance of personal computers, a distributed computing environment (DCE) concept has emerged as a response to the high cost of super-computers. DCE is a virtual computer that is composed of heterogeneous machines sharing their resources for a common purpose. Grids are very large-scale DCE systems whose machines belong to different administrative domains. Recent examples of DCE platforms have emerged in peer-to-peer computing and volunteer computing domains to realise applications such as scientific computing, content-sharing and distributed data storage. Peer-to-peer computing architectures use the Internet to build logical links within their virtual private networks. Volunteer computing systems [2] are composed of millions of personal computers that come together temporarily and voluntarily. Cloud computing platforms [3] are the most recent popular PDC systems that virtualise computing resources such as processing power, storage devices and network connections for subscription-based usage.

Parallel and distributed computing (PDC) systems are generally expensive so simulations are essential for carrying out research experiments with PDC systems. The simulation tools enable construction of repeatable and controllable environments for feasibility and performance studies of PDC.

✉ Taner Dursun
taner.dursun@tubitak.gov.tr

Hasan Dağ
hasan.dag@khas.edu.tr

[1] TÜBİTAK BİLGEM UEKAE, Gebze, Kocaeli, Turkey

[2] Kadir Has University, Cibali, Istanbul, Turkey

Although simulation tools are generally implemented as discrete event-driven programmes [4], there are other alternatives for realisation that Buyya categorised in [5]. There are various simulation tools that combine the concept of objects with the concurrent computation in which simulation entities run concurrently [6] for building discrete event simulations (DESs). These DES simulators have either been used as building blocks or have inspired the development of more sophisticated simulators for application domains like PDC applications (PDCA). There have been many simulators for high-performance computing (HPC), cluster and grid computing fields. However, recent simulators in the PDC community have been proposed for cloud computing. Some of them are originally grid simulators and extended for clouds, whereas others are developed exclusively for clouds. Cloud simulators have to provide different capabilities to grid simulators such as workload virtualisation. The workloads or applications are assigned to virtual machines, and then, these are assigned to computation nodes [7]. Simulators developed for volunteer computing systems provide modelling support for systems composed of large numbers of individually owned computers. Peer-to-peer computing simulators support simulations in which millions of simulation entities related to computers can communicate.

Most of these simulation tools are dedicated only to a specific type of PDC system or a sub-set of its components. Because they are not a result of a research effort to develop a generic model for the implementation of different PDC scenarios. Therefore, we intended to develop a universal simulation model for easy building of simulations of well-known PDC systems in order to fill this gap.

Scalability and accuracy are two important attributes of simulation tools. For more realistic results involving more accurate data, simulations may need to interact with real-time systems. However, existing tools are inadequate to enable easy and fast creation of simulations employing both real-world applications and simulation code. In this paper, especially in order to address this shortcoming, we introduce a model in which interactions are possible between not only real-world systems and simulated entities but also humans and real-world systems. We have accomplished a proof-of-concept implementation of the model.

Then, we built a PDC simulation scenario in which our previously developed policy-based management [8] framework [9,10] plays role of the real-world application that manages PDC systems realised as simulations on our simulation platform.

The rest of this paper is organised as follows. Section 2 describes our simulation model and explains our contribution to facilitate the building of hybrid simulations with a set of execution results. Section 3 includes an evaluation of related work, and the last section presents the conclusion.

2 Proposed Simulation Model

2.1 Generic PDC Simulation Model

The simulators developed for the PDC domain differ from each other by their characteristics in the following attributes introduced in [11]:

- *Simulation model* defines methods to implement simulated resources (e.g. processor, network link storage) and activities. The models for simulated resources can range from simple analytic models to complex ones with regard to the intended level of accuracy and scalability.
- *Platform specification* defines the resources, communication links and topologies to be simulated.
- *Application specification* mechanisms describe the sequence of activities that must be simulated on resources. The realisation alternatives that can be used for this purpose are off-line simulation, formal description and programmatic description.

An abstract model that can be used to define platform specifications for PDC components is already proposed in [12]. It describes a resource concept as a shared entity or capability that could be a machine, a network or a service, whereas a resource management system (RMS) is defined as a service that manages a pool of resources. Figure 1 shows the abstract model of RMS and a sample system with multiple levels of interconnected RMSs. The resource consumers can be either actual applications or another RMS belonging to a higher layer, while the resource provider (broker) can be an actual resource or another RMS that represents a lower layer. The support functions such as naming and security can be accessible through the support interface. The peer interface is intended for interaction with other RMSs and may support protocols such as resource discovery, trading, resolution and co-allocation.

Although many PDC architectures can be built with this abstract model, in order to become more generic it still needs improvements such as adding support for distributed ownership of the resources. In other words, traditional DCEs are generally installed for a single administrative domain, and they serve jobs submitted by users of that domain. However, in some DCEs like grids, supporting different administrative domains is more challenging. Issues such as authorisation of jobs, quality of service (QoS) mechanism, satisfying resource requirements of jobs and constraints originated from semantic relationships between components of PDC systems should be taken into account [13]. Therefore, we developed **HeteroSim** which is a general purpose, policy-based manageable simulation framework for PDC systems by realising an enhanced version of the abstract model of [12] with the following main improvements and simplifications:

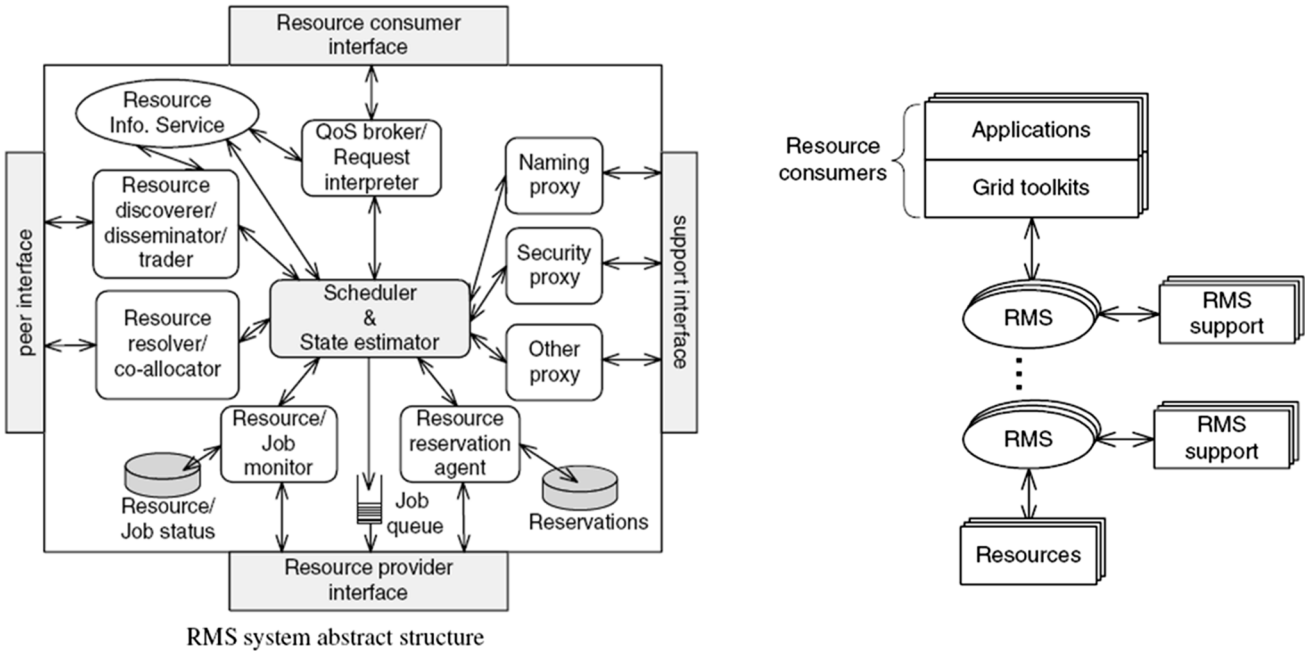

Fig. 1 Abstract RMS model and its use for modelling of Grids

- Adding an inter-domain interface for interaction with RMSs belonging to different administrative domains.
- Adding a PEP (policy enforcement point) [8] interface as a support service in order to achieve policy-based management capability for each component of the RMS. Therefore, each component of the HeteroSim RMS is instrumented for policy enforcement.
- Adding observation capability to the resource/job monitor to observe resources of a peer RMS.
- Merging functionally close modules into one module to obtain an easily implementable model.

Each component of RMS nodes of our framework can be replaced with a different implementation thanks to abstract interfaces. By connecting our RMS nodes either hierarchically or peer-wise and then grouping them into administrative domains, it is possible to construct policy-based manageable simulations of centralised, distributed and hierarchical PDC systems. Figure 2 shows a sample PDC scenario including mixed types of peer and hierarchical RMS relationships that are built with triple RMS nodes. Another significant feature of the HeteroSim framework is its support for hybrid simulations. In this manner, for example, we are able to make our already developed POLICE [9,10] PBM application (a real-world application) communicate with entities of HeteroSim simulations.

As the starting point of our HeteroSim development effort, we selected the GridSim Toolkit [14] which has a largely academic background. Then, we developed our DES toolkit which is a refactored version of GridSim Toolkit from a new point of view. Our refactoring effort results in the following main improvements:

- Capability was added for modelling and simulation of heterogeneous types of entity, from both the simulation and the real world.
- A management interface for PEP was added, and enforcement activities were implemented. Policy-based manageable versions of all RMS components were developed to make it possible to change simulation entity behaviours with policies.
- Support for local and remote job distinction was added.
- New entity types such as Trader, RMS, Dispatcher and Local Information Service were developed. Therefore, the complexity of the scheduler component is decreased by moving the functionalities to where they must exist. Jobs are not submitted directly to the scheduler anymore but to the trader.
- Support for automatic construction of RMS topologies which are deeper than two levels (according to the configuration specified before simulation start) was added.
- Allocation of a single job to multiple resources was made possible.
- A mechanism for adjusting the simulation execution speed was implemented.
- The mechanism for synchronising simulation entities and multi-thread support were improved.
- For resource allocation, a number of predefined strategies such as LONGEST_FIRST, LOCAL_FIRST, EQUAL,

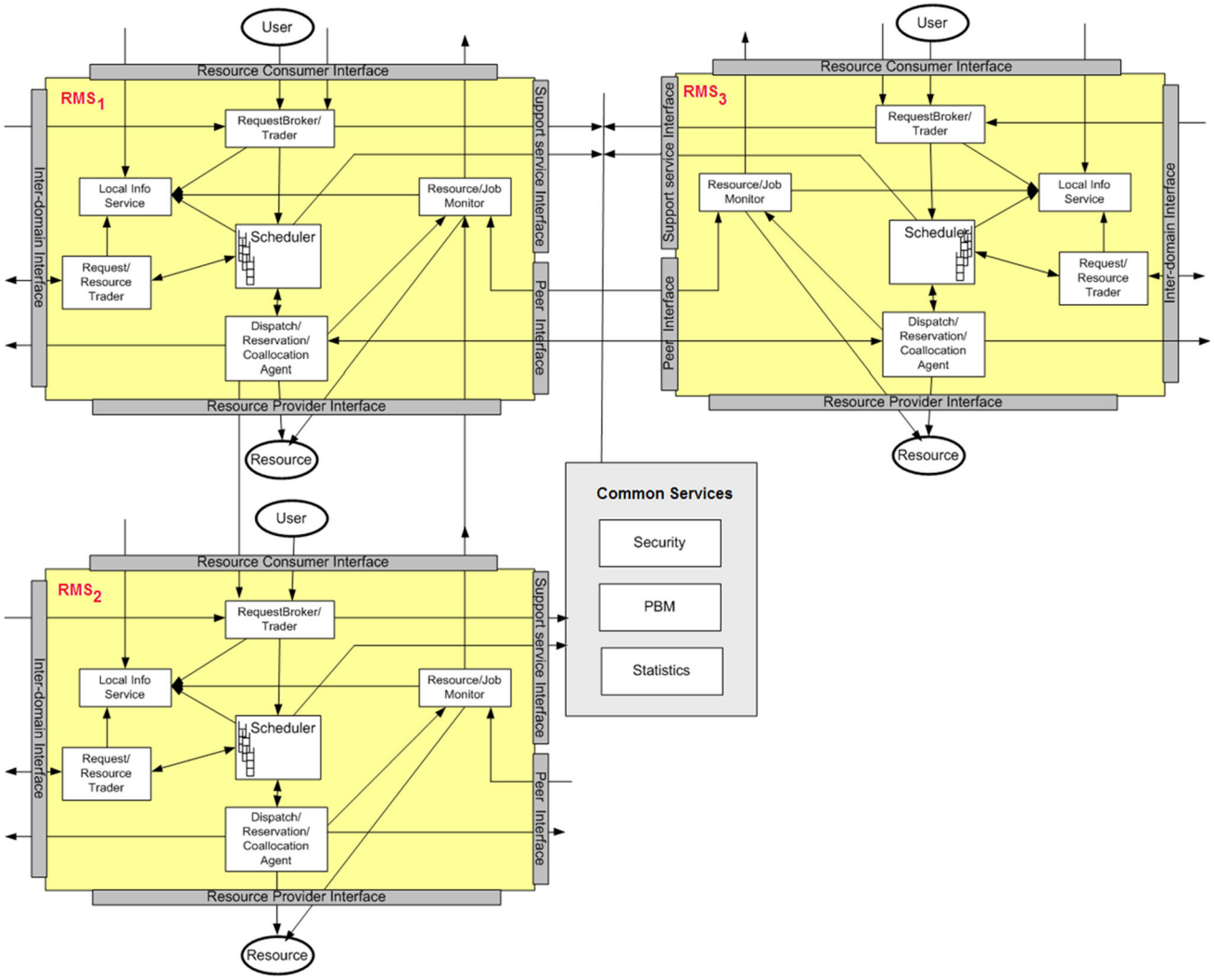

Fig. 2 A sample PDC scenario built with HeteroSim RMS abstraction model (revised version of [39])

LEAST_REMAINING_FIRST and REMOTE_FIRST that can be triggered via policies were made available.

- A new hierarchical resource addressing schema which is required for policies was employed (as in the example of HPC1/RMS2/Resource6).
- A virtual organisation (VO) attribute was added to the Resources and Users entities in order to facilitate multiple administrative domains.
- Resource topology and workload trace loaders were developed and tested with real workload trace files obtained from various HPC centres.
- Mechanisms were implemented for on-the-fly calculation of metrics based on different perspectives of simulations such as whole PDC system, RMS and User (as shown in Table 1).

  Addressing the metrics as parameters within the policies was also made available. Execution results can be shown as charts implemented with the JFreeChart library [15] for each statistical metric. Samples for the resource allocation chart and metric chart are shown in Fig. 3.
- As an important advancement, more detailed resource modelling support was realised. HeteroSim now provides finer granularity in resource modelling compared

Table 1 Implemented metrics in HeteroSim

based on / metrics	RMS	User	HPC
AET (Average Execution Time)	√	√	√
AWT (Average Waiting Time)	√	√	√
AJL (Average Job Length)	√	√	√
TJL (Current/Cumulative Total Job Length)		√	
MakeSpan			√
Max, Min, Average Job length			√
System workload ratio			√

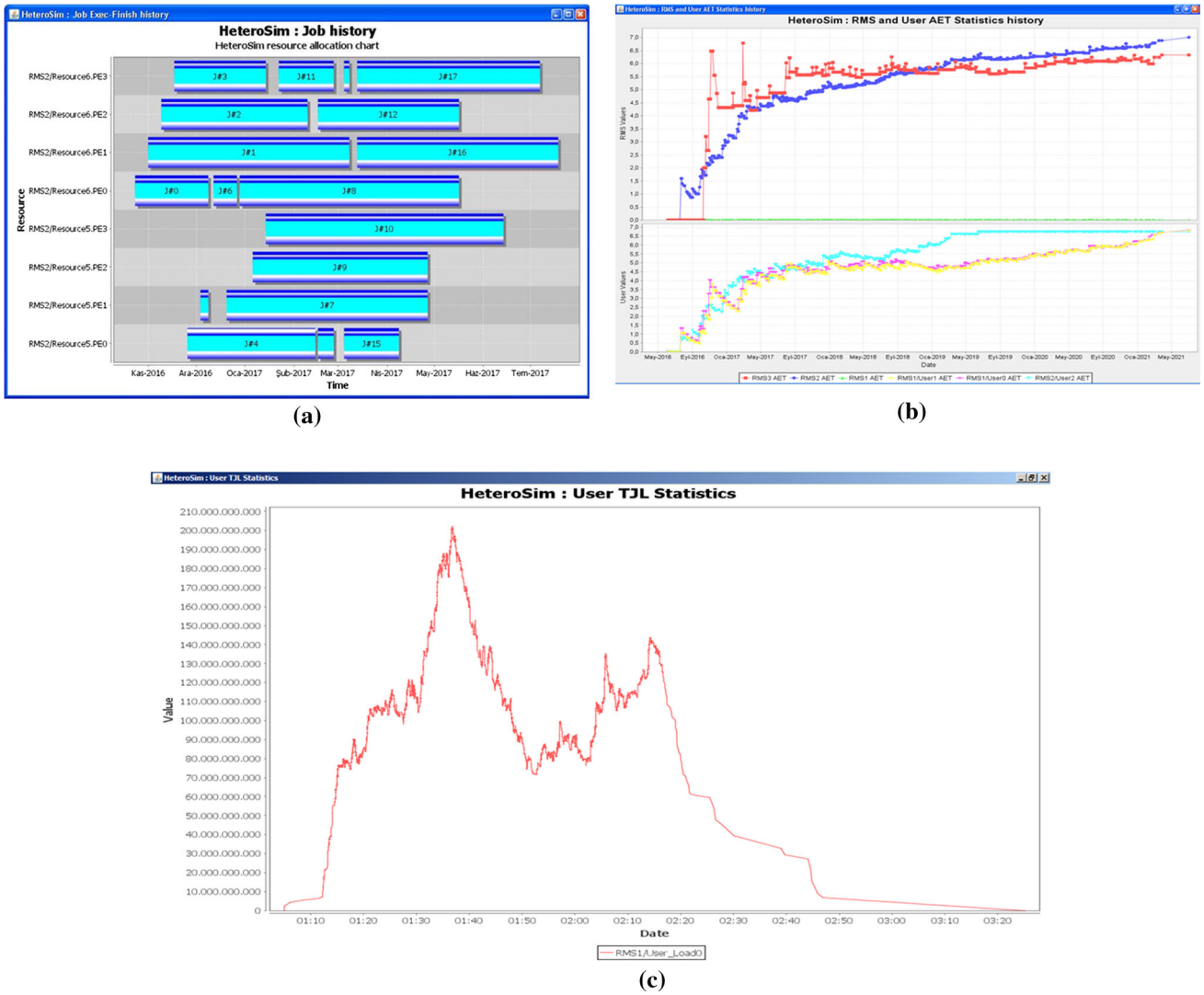

Fig. 3 Sample metric charts. **a** HeteroSim resource allocation history chart. **b** HeteroSim AET metric chart. **c** Result of SDS Blue Gene workload trace replay in HeteroSim (total job length)

to GridSim and its successors. As shown in Fig. 4, HeteroSim can model each computer as a single resource which has processing elements (PEs), whereas GridSim is only able to address a group of computers as a resource. During HeteroSim simulations, each computer can be represented with a simulation entity (SE) so that the behaviour of each computer can be programmed and changed independent of the others. The HeteroSim schedulers calculate simulation state at computer level granularity while allocating resources, whereas in GridSim, a group of computers is treated as a single resource and only one scheduler is assigned to each resource. Figure 4 also shows the other HeteroSim components for facilitating more realistic job processing that GridSim toolkit does not have.

In the rest of this section, we first outline the system architecture of HeteroSim, followed by a detailed explanation of the hybrid simulation mechanism through which the interactions take place between simulation and real-world entities. We then present how to employ HeteroSim for PBM study of HPC systems with simulations.

2.2 Heterogeneous Simulation Architecture

The details of HeteroSim architecture which are shown by Figs. 5 and 6 can be found in [16]. HeteroSim architecture consists of four types of component. The real-world entities (REs) represent external real-time systems and are able to interact with the simulation entities (SEs). The actions of both REs and SEs can influence the outcome of simulations.

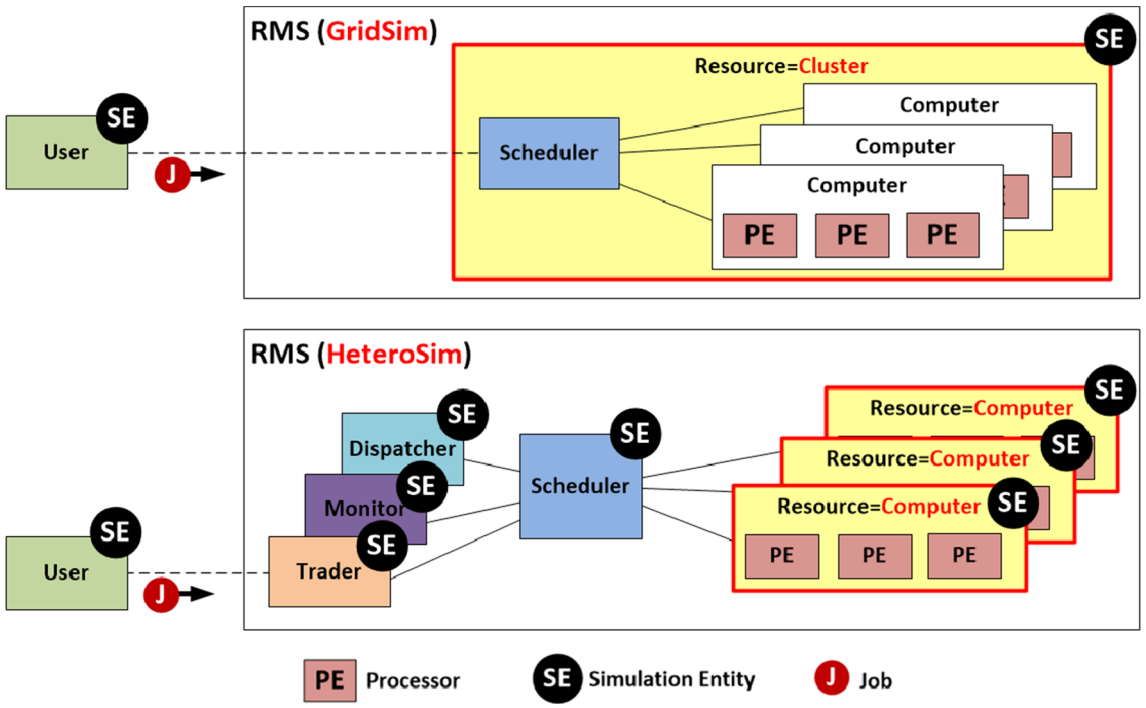

Fig. 4 Comparison of resource modelling granularity levels of HeteroSim and GridSim

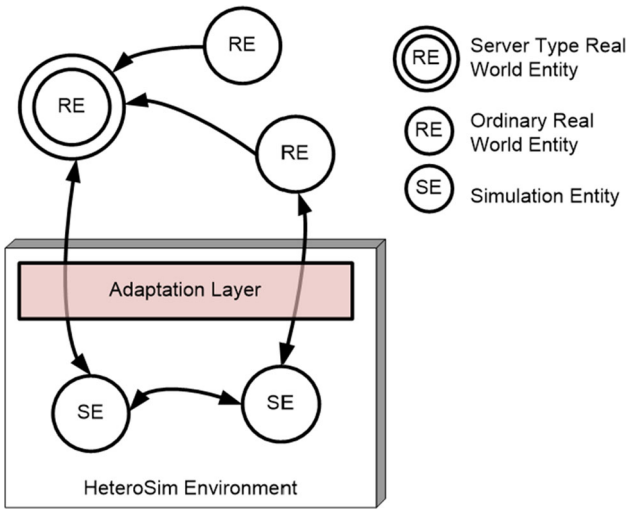

Fig. 5 HeteroSim simulation architecture

Bi-directional interactions of SEs and real-system components occur through a communication mechanism, called the adaptation layer (AL). The AL acts as a proxy between SEs and real-world applications to facilitate expansion of simulations to external real-world applications without modifying them. Similar to SEs, the REs are also part of the simulation sessions. The REs can be either an ordinary type or a server type application. The server type RE (SRE) can be any kind of real-world application so that SEs or other REs can consume its service. For use with different types of real-world application, the AL can support various communication technologies as shown in Fig. 6. A proxy entity (a so-called adapter entity) is automatically created for each RE. While the adapter entities communicate with REs via proper technology such as SMTP, JMS [17] and remote procedure call (RPC), on the other side they act as pseudo-SEs in order to interact with other SEs on behalf of REs. Thanks to the adapter entities, the SEs don't include any code specific to the communication technologies related to the real-world applications.

HeteroSim follows the process-oriented approach where each SE can be considered as a separate process. The SEs have an independent thread of control (pseudo-parallel execution), and they use event-based messaging. There is a central event queue, called a future event list (FEL) that contains timestamp ordered events. The scheduler observes the FEL and finds the event with the smallest time stamp and invokes the entities related to that event. After all entities have been executed for the current instance of simulation time, the scheduler pops the next event off the queue and advances the simulation clock. This flow continues until no more events are generated.

For more accurate simulations, communication links and topologies must also be included in platform specification model of simulator. However, a few of simulators provide a complete modelling capability for network type of resources. Although HeteroSim does not include a complete network model, for accuracy it can employ the communication mech-

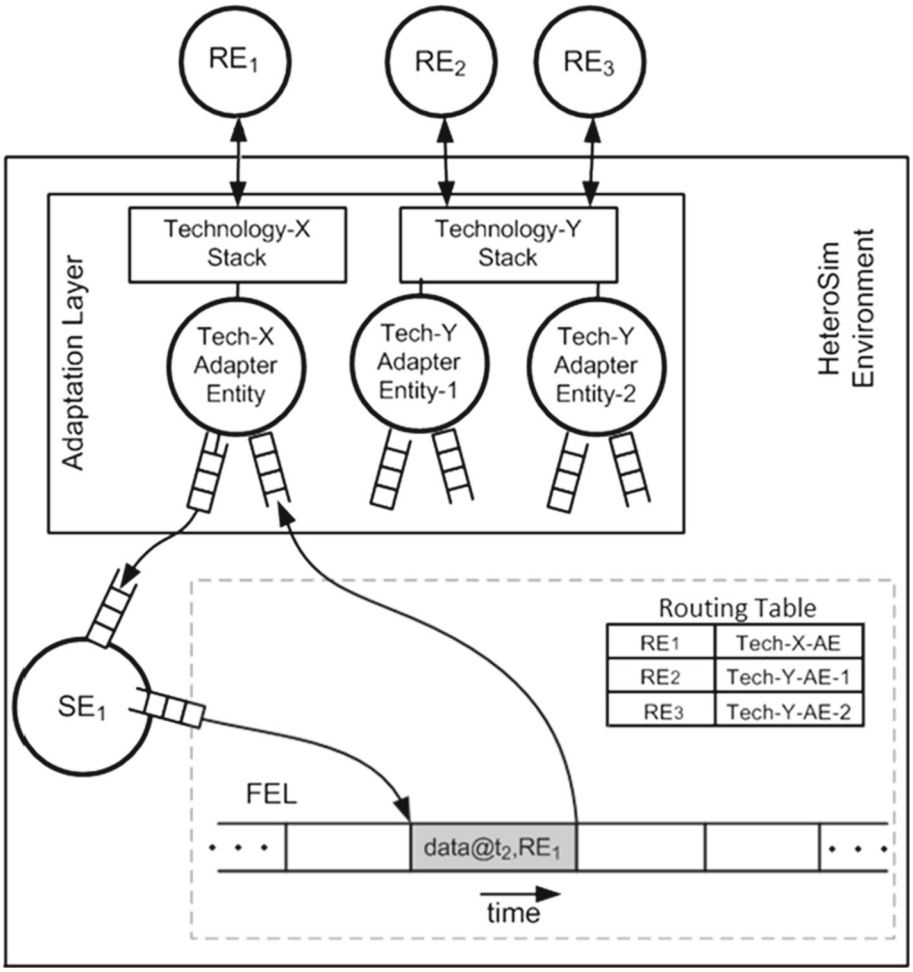

Fig. 6 Interaction between SEs and REs through AL

anisms in simulation via the AL, as depicted in Fig. 7. The AL can already employ the original protocol stacks of communication technologies within simulation sessions. A sample simulation scenario for the case of Fig. 7a can be arranged as shown in Fig. 7b. In this sample scenario, the SE_2 can select an indirect path over AL to send messages to SE_1 via communication layers associated with Technology-X.

2.3 Scalability of HeteroSim

Scalability is one of the most important characteristics that simulation tools must provide. To be counted as scalable, simulators should support modelling millions of simulation entities. For DES-oriented simulators, this is not so easy due to limitations such as thread or process counts, memory size and overheads like context switching. Most of the Java-based simulators allow only thousands of SEs due to JVM's thread limitations. An analysis given in [18] compares scalabilities of grid simulators, GridSim, GES and SimGrid. The authors performed all tests on the CalcUA cluster at the University of Antwerp which hosts 256 Opteron 250 nodes running 64-bit Linux distribution. During our study, we implemented the same tests with HeteroSim. However, our testbed includes 64-bit JVM running on a 64-bit Windows 8 computer with 8 GB of RAM and a 1.8 GHz Intel i7 processor. Although our test platform is different, we compared the results of our tests with their results [18] just to give an idea of the scalability level of HeteroSim which is derived from scalable GridSim tool. These tests are quite important to figure out whether our effort to provide more granularity with additional HeteroSim components, interfaces and Java Thread implementation way dose not undermine the scalability.

During the *general scalability test* (Test-I), the number of consumers (users) is scaled from 1 to 10,000 while changing the number of resources from 1 to 1000 so that a maximum of 10 consumers can use each resource. As shown in Fig. 8, GES, which uses a single-thread implementation, scales up best and HeteroSim scales similar to GridSim. However, GridSim was unable to simulate 10,000 consumers because of its need to create more than 32,000 threads that a

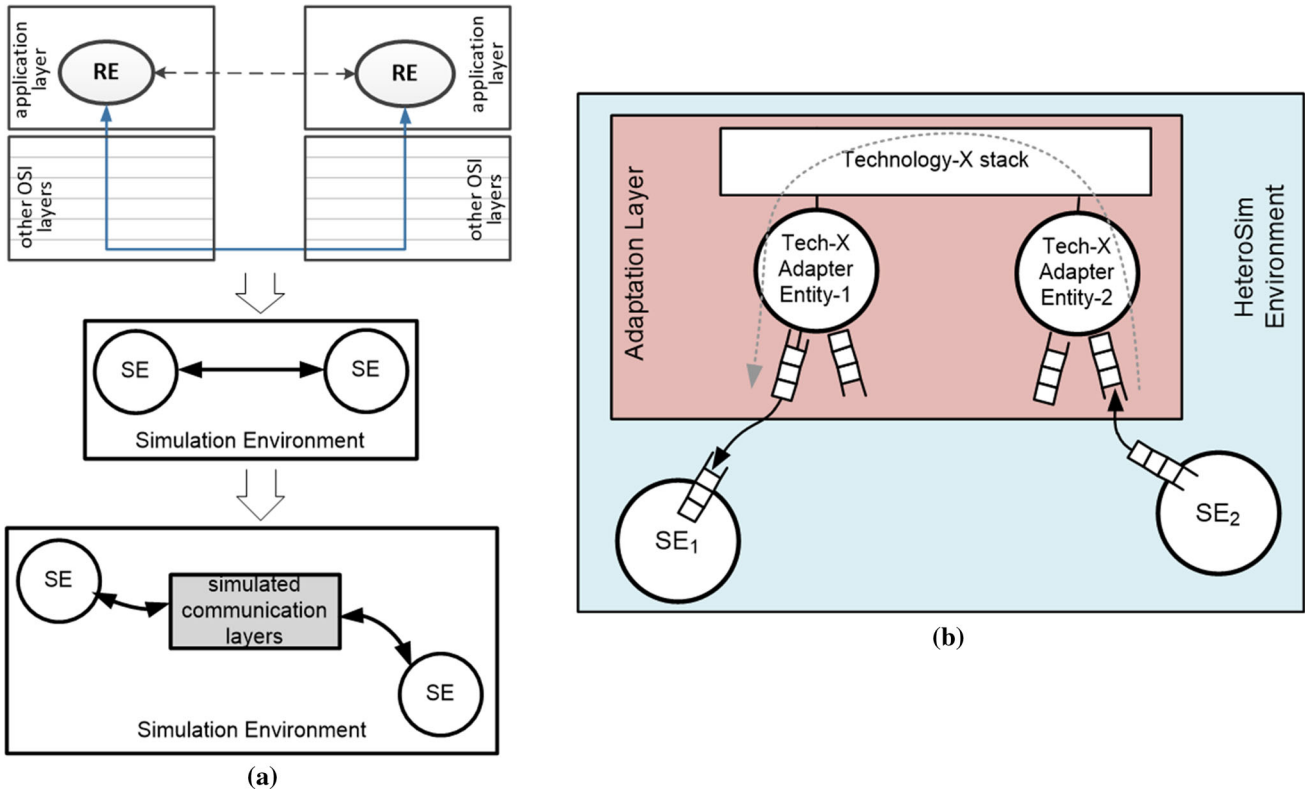

Fig. 7 Obtaining more accurate simulations

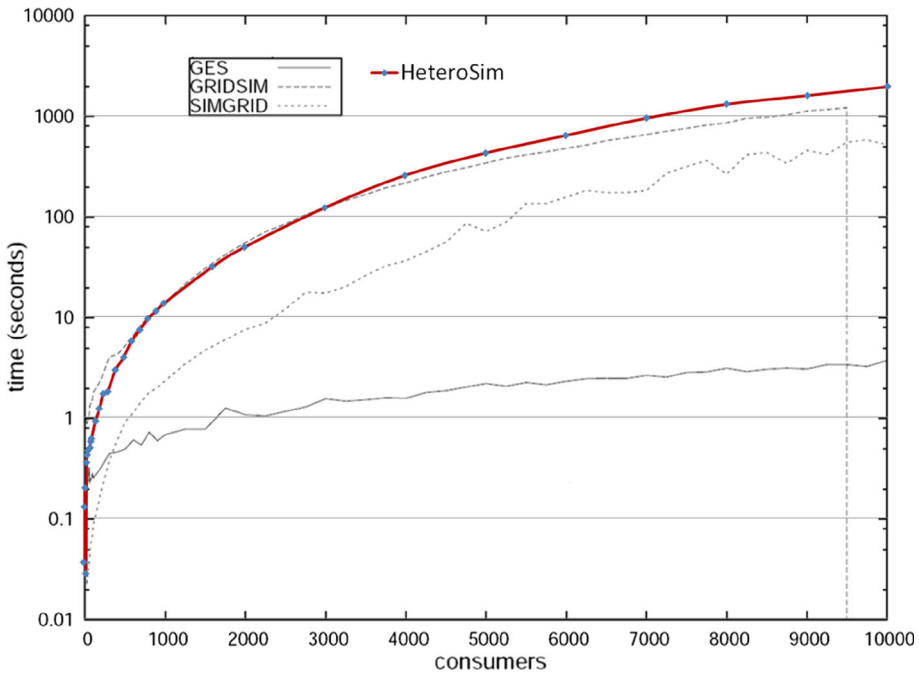

Fig. 8 Simulation time as a function of the number of consumers on logarithmic scale

normally configured Linux cannot handle. SimGrid can overcomes these limitations via light-weight, non-preemptive threads, called a continuum. In our tests, however, we did not encounter any problem due to thread count because it was only limited by the memory size of the Windows 8 computer on which the 64-bit JVM runs.

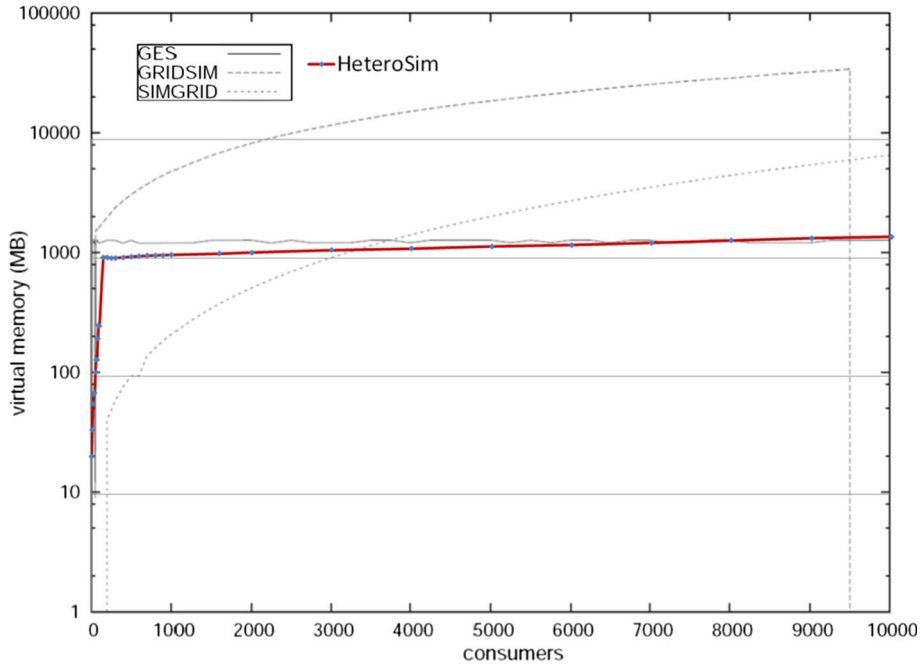

Fig. 9 Maximum virtual memory allocation as a function of the number of consumers

Figure 9 shows the virtual memory usage related to the number of consumers. GES and HeteroSim scale linearly. The nonlinearity in GridSim and SimGrid is because of the high stack size value used in tests. Actually, simulators use less virtual memory than allocated. Because threads relate to simulation entities using a small portion of their stacks, we run the tests with a smaller stack size (128 KB).

Figure 10a shows the maximum thread count in a function of the number of consumers. According to the charts, it is easy to find that the maximum thread count is equal to 1.5 times the number of consumers. In fact, this is consistent with the granularity of the HeteroSim model in which 5000 threads for 1000 RMSs (including 1 Resource, 1 Scheduler, 1 Treader, 1 Dispatcher and 1 LIS) and 10,000 threads for consumers are created. Actually, these test scenarios are not suitable to reflect the actual capacity of HeteroSim due to assigning one RMS for each resource and thus using HeteroSim with coarse granularity.

The Job Scaling test (Test-II) analyses the impact of the number of jobs on the simulation time. In this test, each consumer sends 100 jobs, and the number of consumers is increased from 1 to 100. From the results shown in Fig. 10b, it can be seen that the simulation time in GridSim, SimGrid and HeteroSim scales linearly with the number of jobs, as expected due to using DES-oriented simulation implementation. However, GES, which is a discrete time simulator, is virtually unaffected by the number of jobs. In contrast, discrete time simulators scale linearly with the size of jobs while DES simulators are unaffected. Until a consumer count of 60, HeteroSim performs better than SimGrid. However, after this point, the increasing number of entities cause longer times for statistical calculations and then HeteroSim falls behind SimGrid. Therefore, metric calculation mechanism of HeteroSim needs to be improved further.

2.4 Using HeteroSim in PBM Simulation

We claim to be able to build heterogenous simulations thanks to the hybrid simulation support of our HeteroSim prototype. As a proof of concept, we integrated our previously developed policy-based management (PBM) framework with HeteroSim simulations. This sub-sections describes the integration effort.

Matters such as lots of parameters that should be configured properly by administrators, highly heterogeneous components and distributed ownership of resources make the management of PDC systems more complicated. Moreover, PDC systems obviously cannot tolerate interruptions caused by administrators for re-configuration. Employing PBM tools for management of PDC systems can respond to these matters. These tools use policies to specify the desired system behaviour. Policies are automatically translated by the PBM system into commands and configuration parameters that are understandable to the managed devices. According to the commonly accepted Internet Engineering Task Force's (IETF) PBM model [8] (shown on the left side of Fig. 11), policies defined by administrators are stored in a policy repository. A PDP retrieves policies from the policy repository, interprets and sends them to the PEPs for enforcement and answers the decision requests from PEPs.

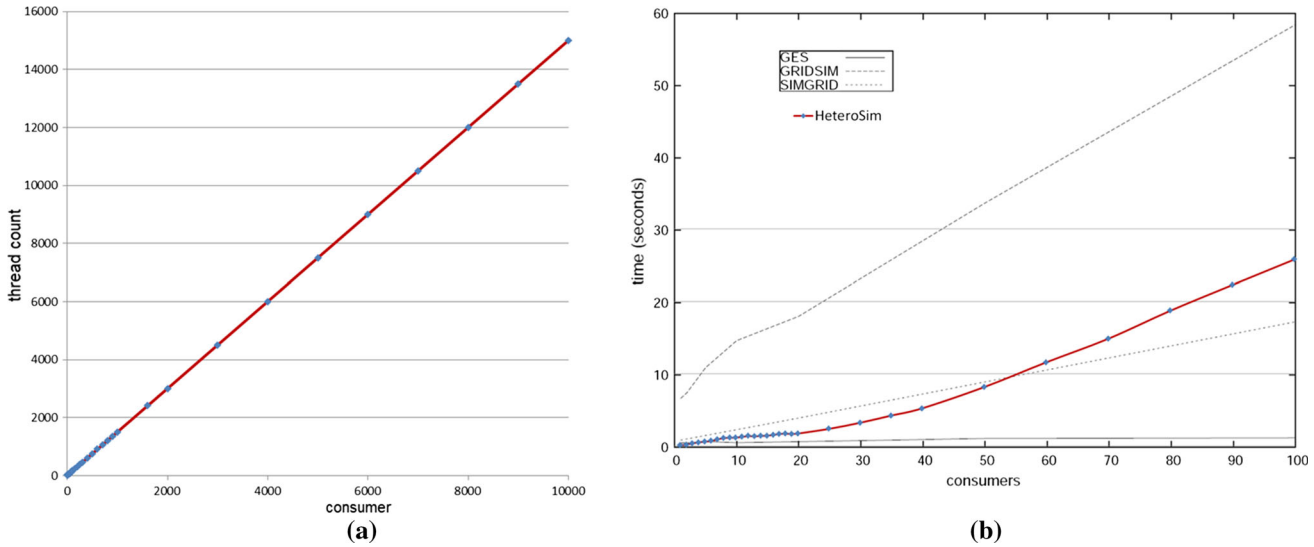

Fig. 10 Test results. **a** Maximum number of threads as a function of the number of consumers. **b** Simulation time as a function of the number of consumers

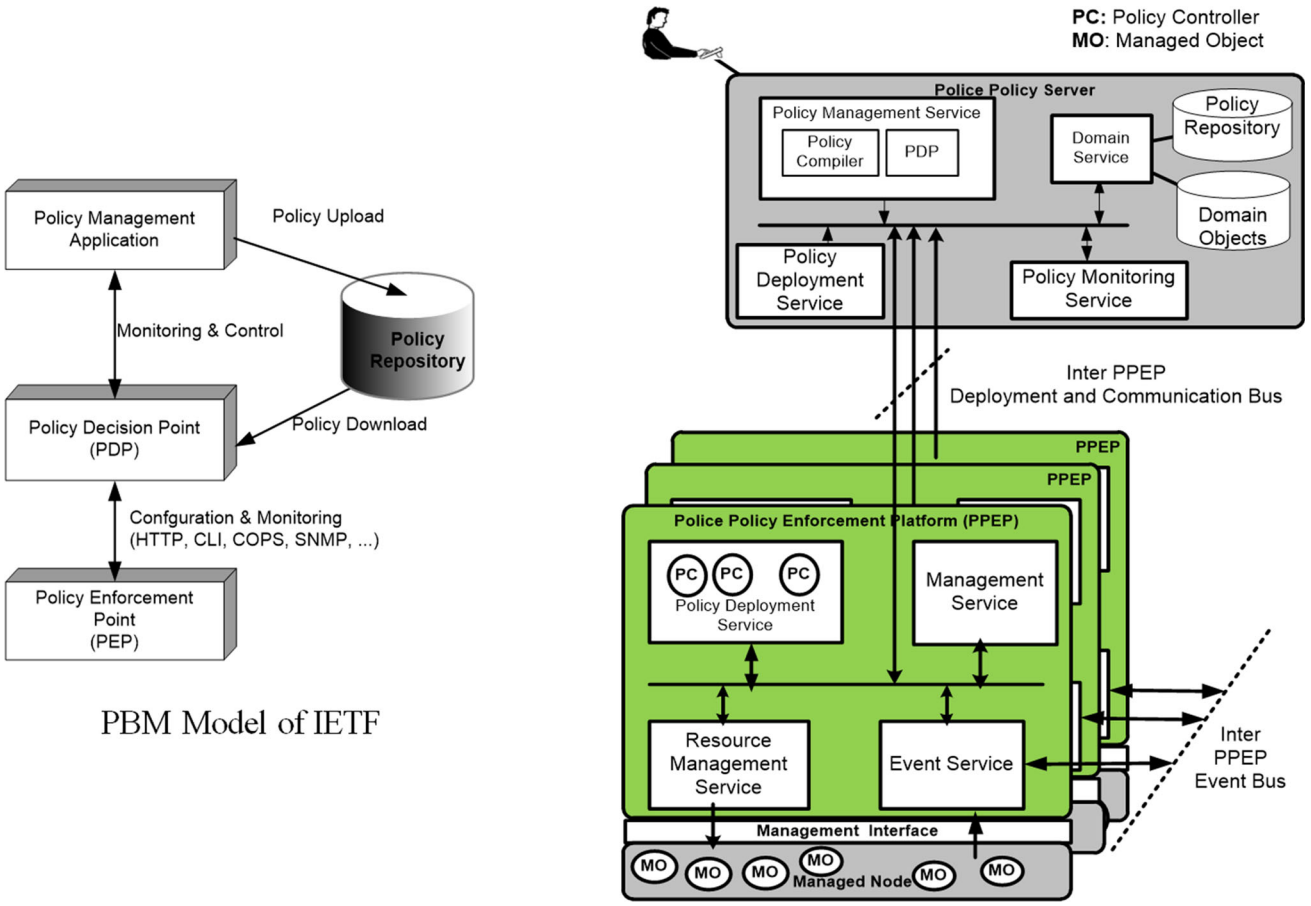

Fig. 11 General architecture of PBM system and architecture of POLICE framework

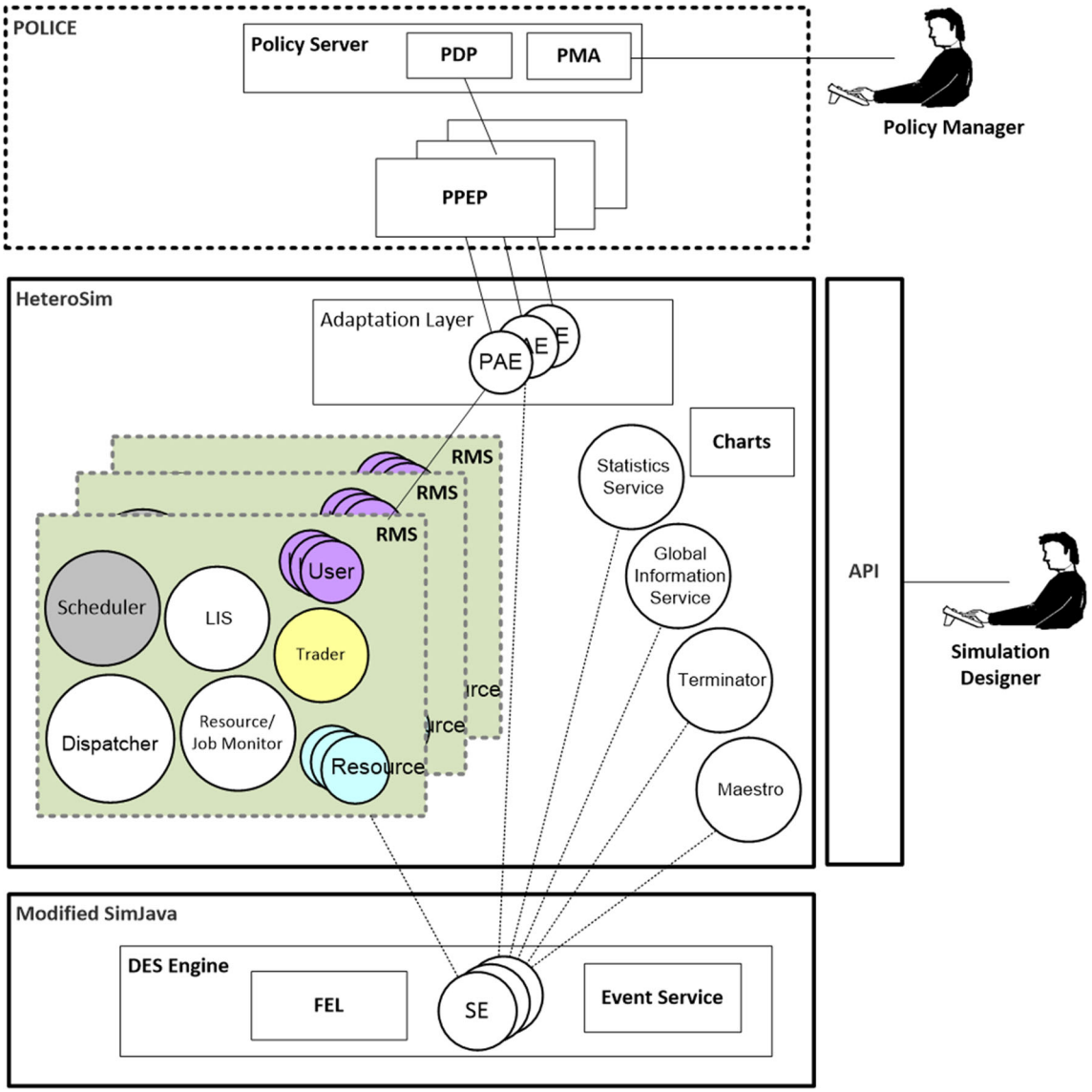

Fig. 12 Integration POLICE PBM and RMS within HeteroSim

In order to prove the effectiveness of a real PBM tool for the management of PDC systems, we conducted a study with our general purpose PBM framework, called POLICE, whose architecture is shown in Fig. 11. However, during such a study, in order to observe the effects of the policies on PDC systems, the PDC systems need to be evaluated under different scenarios such as varying the number of resources, workloads and users while different policies are enforced. In a real PDC environment, it is hard and perhaps impossible to perform evaluation of different scenarios in a repeatable and controllable manner because the status of resources (availability and loads) varies continuously, and it is impossible to control user activities. Moreover, PDC systems are very expensive, and it is not always possible to have an opportunity for academic research with them. In fact, in recent years, testbeds such as Grid'5000 and FutureGrid have been accessible for researchers to perform their studies. However, modifying the source code of the testbeds to integrate a PBM is obviously not permitted.

For this reason, instead of using real PDC systems or testbeds, we preferred using PDC simulations within HeteroSim for our study. However, the problem with this choice was the implementation burden of modelling behaviour of a PBM tool within simulation. Thanks to the ability of HeteroSim to involve real software applications in simulations, we were able to employ our PBM framework to manage the simulated PDC components [19]. In this simulation setup, POLICE components (as the real application elements) can interact with the simulation entities representing the PDC system's components.

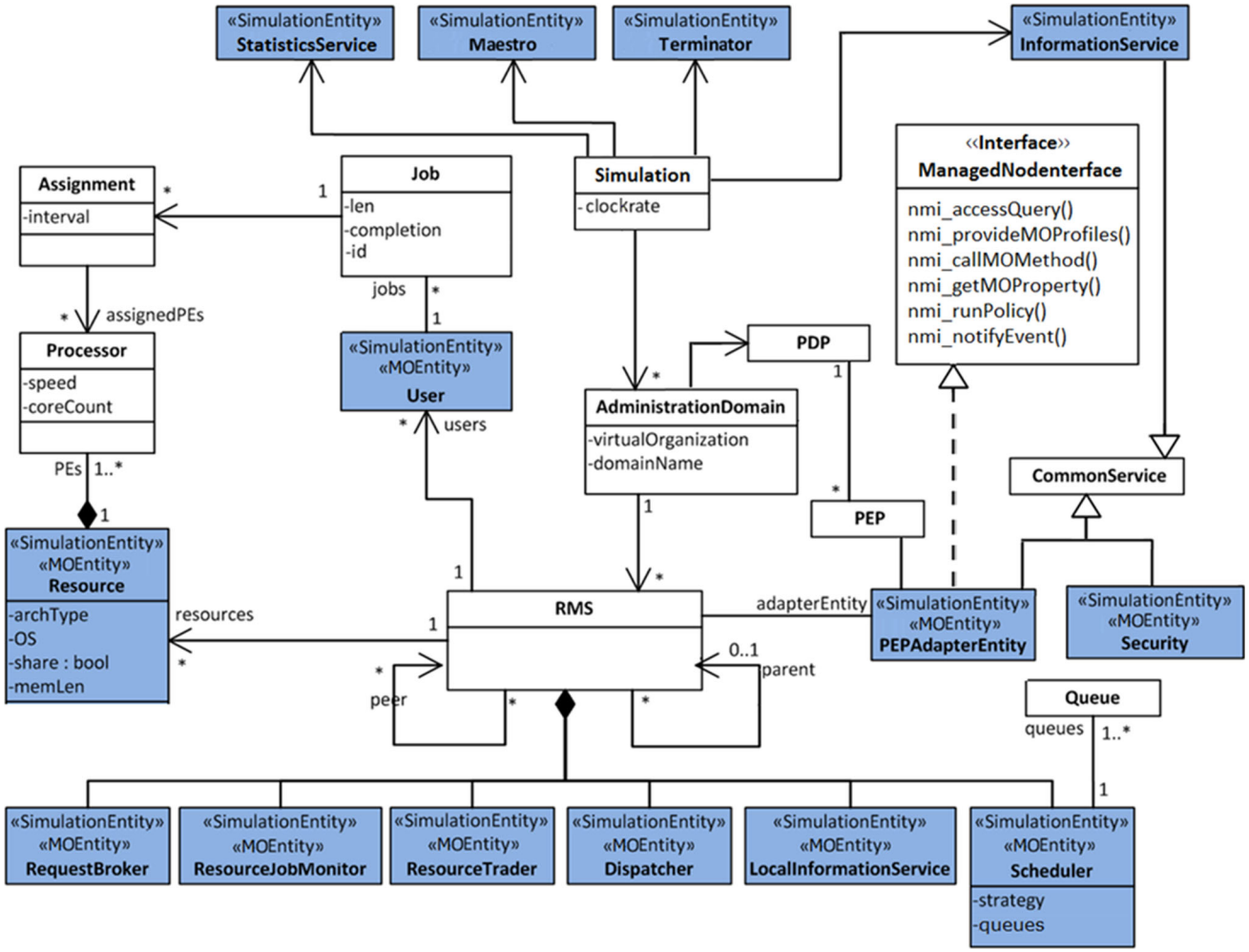

Fig. 13 Information model of HeteroSim for PDC modelling

In order to enforce policies, we integrated POLICE PEP with PDC model simulations by using the AL mechanism of HeteroSim. Figure 12 shows the interconnection schema with a sample scenario including three RMSs and their dedicated PEPs. Both application and platform specifications can be defined by the simulation designer via an application programming interface (API).

The functionalities of an RMS's components are usually dependent on each other. Therefore, in our RMS model, not only the scheduler but all components of a PDC system can be managed to achieve a holistic management. Otherwise, various inconsistencies may occur. For example, in a case for which access control is managed by policies but scheduling is not, if the jobs are assigned to restricted resources, this fact remains unknown until the jobs are really sent to the resources.

Assigning a PEP for each RMS and a PDP for each domain makes it possible to construct RMS hierarchies belonging to different administrative domains. Each PDP allows specification of policies for its own administrative domain. Inter-domain resource sharing can be performed according to a policy negotiation, service level agreement (SLA) and so on. However, specifying policies may not be enough by itself for effective management. Sometimes, semantic limitations on resource allocations and job executions must also be considered along with policies. In our PBM model, handling this kind of semantic relationship can be performed with the E-Code concept [8].

Figure 13 shows the information model of our PDC simulation framework. Each RMS communicates with its PEP via the associated AL entity, called PAE (PEP adapter entity). The managed node to be controlled by PEP is the entire RMS while the managed objects (MOs) are the components of RMS.

The primary actor objects in the policies are the users, whereas the target objects are the resources. With this model, the following components and processes can be addressed within POLICE policies:

• Job	• Scheduler
• Job queue (strategy, priority and so on.)	• Broker
• User	• Trader
• Resource	• Resource allocations
• Reservations	• Access control

For this study, Police policy language is extended with the keywords of HPC, Statistics, RMS and Scheduler. Sample language statements that can be used in policies are as follows:

• rms1.scheduler.awt ('user4')	• rms1.scheduler.awt()
• rms1.scheduler.awt ()	• rms1.scheduler.hpcAWT()
• rms1.hpc.awt()	• rms1.scheduler.userAWT('user9')
• getResponsiblePE-PofMO('rms1.hpc')	• hpc.awt()
• findQueueName ('rms1/hpc')	• hpc.loadRatio()
• rms1.user.awt('user3')	• rms1.scheduler()
• hpc.rms2.user1.aet()	• rms1.scheduler.setJobSelection Strategy('LocalFirst')

For example, the following policy (shown in Fig. 14) changes the job selection strategy of the scheduler component of RMS_3 when the average job waiting time for $User_1$ is greater than zero so that the local jobs can have priority. Figure 15 shows the enforcement details of the policy specified in Fig. 14.

3 Related Work

In recent years, modelling and simulation have emerged as important research areas and for the simulation of PDC systems, a number of simulation tools have already been developed. JavaSim [20] and JSIM [21] are tools for building Discrete Event Simulations (DESs). J-Sim [22] is another Java-based network simulator that is based on the components communicating with each other by wiring their ports together. J-Sim can be used for both DES and real-time process-based simulation. Silk [23] and SimJava [24] are two early Java-based libraries for process-oriented DES.

These basic level simulators either have been used as building blocks or have inspired the development of more sophisticated simulators for application domains like PDC Applications (PDCA). The most well-known simulation tools for PDCA are analysed in depth in [11].

There have been a lot of simulators for Cluster and Grid computing fields. ChicSim (Chicago Simulator) [25], GangSim [26] and its successor, virtual organisation-centric Ganglia [27] are grid simulators. GridSim [13] and SimGrid [11,28] are the most popular and widely used simulators. GridSim [13], which is built on SimJava, can models clusters, users and network communications. Its distinctive feature is economical resource allocation [29]. GES (Grid Economics Simulator) [18] is another significant simulator that allows distributed simulation executions. It is a single-threaded simulator. SimGrid [11,28] is one of the few tools that can be used for simulation of more than one type of PDC. It has a DES-oriented simulator infrastructure which its developers claim is generic and versatile. SimGrid has a single thread (core context) to calculate the states of the simulation models. A more widely used alternative is the multi-threaded approach in which the states of the simulation models are composed of threads representing simulation entities. The latter approach is used in GridSim and HeteroSim. SimGrid also provides a special kind of asynchronous execution mechanism called continuation to defeat thread count limitations.

Recent simulators in the PDC community have been proposed for cloud computing. Some of them are originally grid simulators extended for clouds, whereas others are developed exclusively for clouds. CloudSim [30] is a very popular cloud simulator built on GridSim infrastructure. Many simulators have been derived from it; TeachCloud, for example, adds service level agreement (SLA) and graphical user interface (GUI) components into CloudSim. DynamicCloudSim [31] tries to increase the strength of CloudSim against failures and dynamic changes. GroudSim [32] is a DES simulator that supports both Grids and Clouds. It employs an event-based model instead of multi-threads for scalability. iCanCloud [33] is another cloud simulator that supports large storage systems.

Simulators developed for volunteer computing systems provide modelling support for systems composed of large numbers of individually owned computers. Most of the simulators such as SimBA [34], EmBOINC [35] and SimBOINC [36] try to simulate the capabilities of BOINC which is the most popular volunteer computing system.

Peer-to-peer computing simulators are summarised in [37]. PeerSim [38] is the most commonly addressed peer-to-peer simulator developed in Java. OverSim [39] is a DES-oriented simulator that provides more realistic network models. P2PSim [40] and PlanetSim [41] support peer-to-peer protocols such as Chord, Kademlia Koorde and Gia.

Most of these simulation tools are dedicated only to a specific type of PDC system or a sub-set of its components such as scheduler, broker and so on. Few of them are the result of a research effort to develop a generic model for the implementation of different types of PDC system. Moreover, none of them supports heterogenous simulations as HeteroSim does.

Accuracy and scalability are the most required attributes of simulators. The analysis given in Sect. 2.3 provides scalability test results of various well-known grid simulators and HeteroSim. According to the test results, the scalability of HeteroSim is reasonable. Although we involved a different

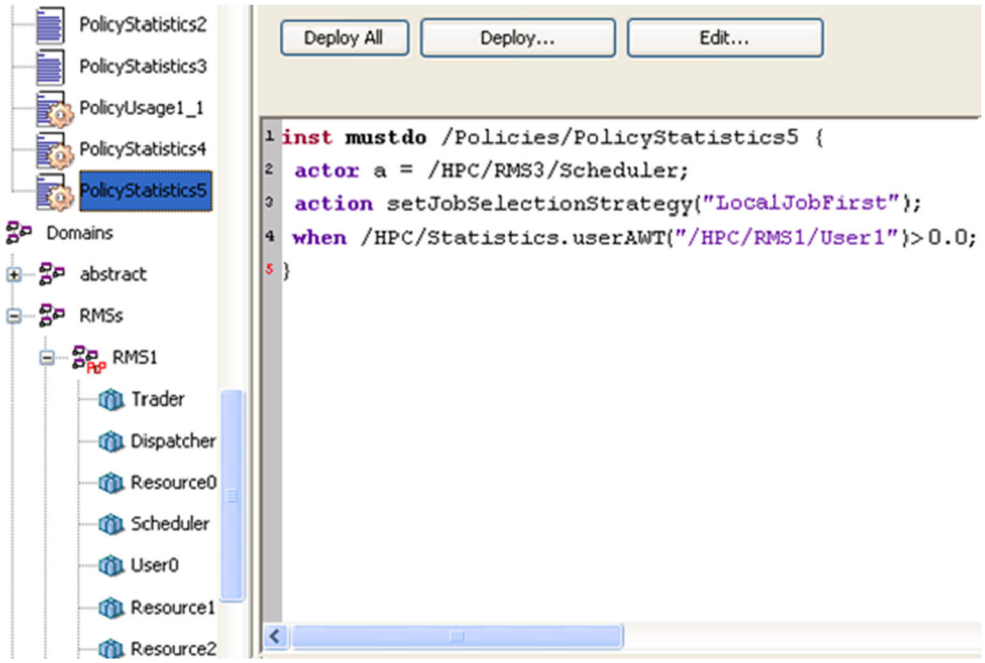

Fig. 14 Sample policy for HPC management

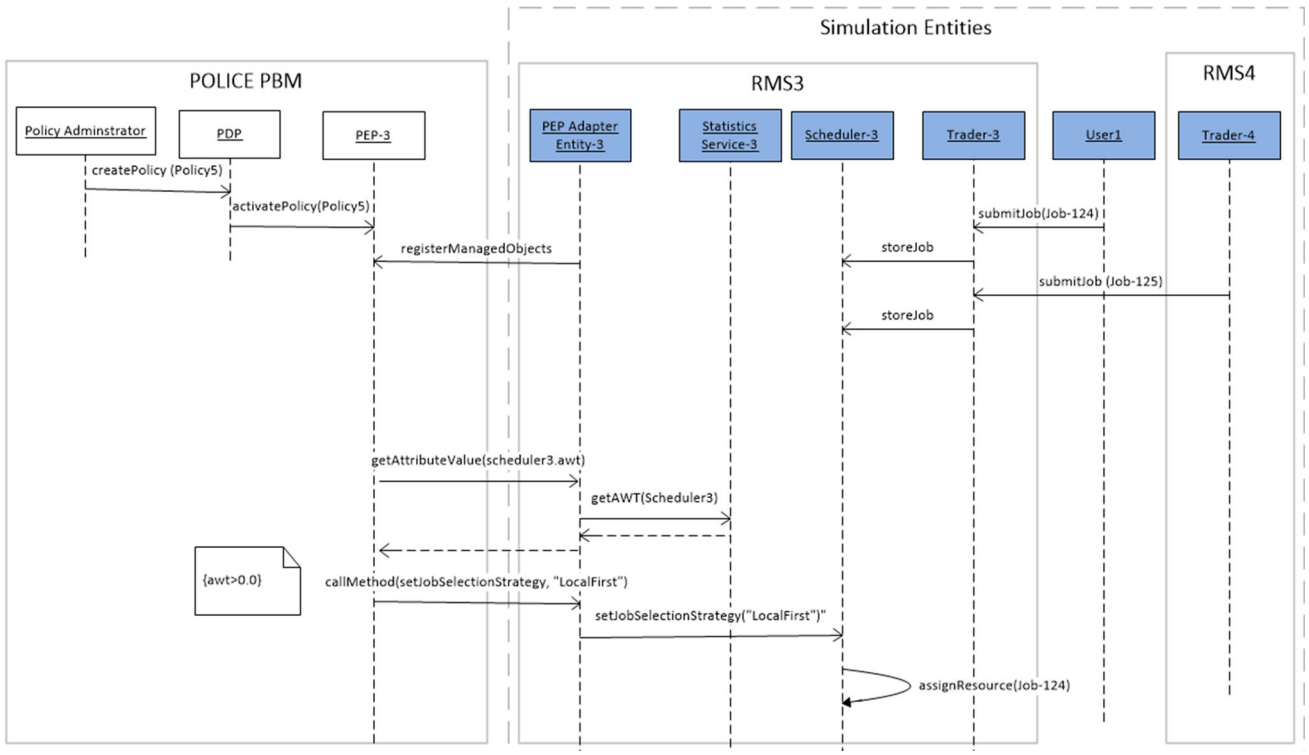

Fig. 15 Enforcement steps of the sample policy in HeteroSim simulation

test platform, by comparing the results of our tests with the results given in [18] readers may get an idea of the scalability level of HeteroSim. However, readers must be aware that the analysis results in [18] were achieved using former releases of GridSim, GES and SimGrid with custom configurations. The newer releases of the tools may provide more scalability. Therefore, for more accurate comparison, the tests must be repeated with the latest releases of the tools.

Because of its versatile resource and network traffic modelling capabilities, SimGrid finds itself occupying a privileged position in the PDC simulation community. As a PDC simulator, HeteroSim does not currently support modelling of network traffic and storage resources. It does not directly support modelling of either cloud computing or peer-to-peer computing platforms. The performance of HeteroSim can be seen as reasonable if its advantages mentioned above are considered. Although HeteroSim falls behind some of the simulators in the tests given in Sect. 2.3, in many cases the memory consumption and simulation time may not be as important when compared to accuracy. In addition to its fine-grain PDC model, HeteroSim improves the accuracy of simulations by involving real-world elements and communication protocols as they are into simulations.

In order to increase accuracy, real-world systems should be as much involved in simulations as possible. However, there have been several studies [42,43] dealing with bringing simulation and real-world codes together in a simulation session. A mature solution has not appeared yet. Distributed SimJava [43] allows the SEs of different simulations to interact with each other and with real-world applications. The main difference compared to our approach is that the real applications cannot participate in a Distributed SimJava simulation as an SE. Furthermore, it requires the real-world applications to provide an RMI interface.

Another study [44] also proposes a distributed simulation model including an interface for interaction with the real world. Although this model provides mechanisms to external users for either passive interaction (the user only visually monitors the output of the simulation) or active interaction (the user is able to interact with the model during a run and then influences the outcomes) with a simulation, there is no bi-directional communication interface between simulation codes and existing real applications. The simulation entities access external information systems just to obtain data related to the simulation.

On the other hand, there are several works [45,46] in the literature which use heterogeneous and simulation worlds together. However, none of them are aiming for the same goal as us.

The HeteroSim permits interactions between SEs running in simulation time and REs running in real-world time. Therefore, the real-time clock should be taken into account in addition to the simulation clock. During an interaction between external applications and SEs, any delay on the simulation side may affect the operation of real-world systems negatively, or vice versa. The co-existence of simulation time and real-time is still an open issue for HeteroSim. In fact, no system can guarantee simultaneous faithfulness to both simulation time and real-time. When the required time to compute the next state exceeds, the amount of real-time available before the next state should occur, alternative methods such as changing execution rate at some ratio to real time (for example, by injecting specialised events into the event list or by adjusting frame rate), degrading or abandoning next state computation, ignoring the delay and attempting to catch up later by running faster may be used for keeping synchronisation with real-time.

In the current HeteroSim implementation, the events related to real-world systems are given the highest priority and are processed immediately. Additionally, as a workaround, it is possible to define bi-directional timeout values for interactions with real-time applications. A mechanism that can be used as another workaround is proposed in [47], called JiST (Java in simulation time). The simulation codes may contain commands related directly to real-time. Then, the JiST framework modifies the simulation program Java-byte codes and embeds its execution logic to achieve performance.

4 Conclusion and Future Work

We have designed a universal model for building simulations of PDC systems and then accomplished a proof-of-concept implementation. With this tool, many well-known PDC architectures can be created easily, either by specifying the configuration of the target architectures manually or by providing workload trace files. Beside manual experiments, we have also successfully performed simulations with the workload traces of various supercomputing centres.

Simulation designers may need to involve real-time systems into simulations for several purposes such as achieving greater accuracy, reducing the building time of simulations, arranging simulations of partially implemented systems and so on. Hybrid simulation support of our model allows easy and fast creation of simulations by employing real software components besides simulation codes. As a proof of concept, we are able to involve successfully our previously developed policy-based management (PBM) framework into simulations as a real-world application. Thus, it is possible to simulate and investigate effects of policies on the behaviours of PDC systems.

Many research areas other than PDC can also take advantage of the HeteroSim infrastructure. Scenario-based software testing [48] would be an interesting area to apply our model. For typical use, all testing scenarios can be easily implemented within SEs. Then, through the AL, the SEs can interact with the real-world applications to be tested.

In order to enlarge our PDC model, we plan to concentrate on the following topics:

- To be sufficiently generic, improving our universal model by adding modelling elements for network traffic and storage resources.

- Adding support for modelling recently emerged PDC platforms such as cloud computing and peer-to-peer computing.
- In addition to existing support for JMS [17] and JMX [49], enhancing the framework by adding more types of interfaces with real-world applications such as WebService, RMI-IIOP, JDBC and so on.
- Adding distributed simulation support.
- Further exploration of the problem of integrating applications running in different time domains and investigating solution alternatives.
- Study to involve the actual time concept mentioned in [47] to solve the possible performance problem and to help the solution of the dual-time domains problem.
- Providing a visual tool for preparation of PDC simulation scenarios.
- Comparing the performances of policy-based manageable RMS with those of traditional heuristic-based RMSs such as exploiting evolutionary algorithms and genetic programming.
- Improving statistical metrics charts so that they do not affect the performance of simulations.

Authors' Contributions TD prepared source codes of the study, carried out the simulation experiments and drafted the manuscript. HD conceived of the study, participated in its design and coordination and helped to draft the manuscript. All authors read and approved the final manuscript.

References

1. Flynn, M.J.: Some computer organizations and their effectiveness. IEEE Trans. Comput. **C–21**(9), 948–960 (1972)
2. Berkeley Open Infrastructure for Network Computing. http://boinc.berkeley.edu/ (2014)
3. Weiss, A.: Computing in the clouds. NetWorker **11**(4), 16–25 (2007)
4. Page, E.H.; Smith, R.: Introduction to military training simulation: a guide for discrete event simulationists. In: Winter Simulation Conference (1998)
5. Sulistio, A.; Yeo, C.S.; Rajkumar, B.: A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. Softw. Pract. Exp. **34**, 653–673 (2004). doi:10.1002/spe.585
6. Banks, J.; Carson, J.S.; Nelson, B.L.; Nicol, D.M.: Discrete-Event System Simulation. Prentice Hall, Englewood Cliffs (2001)
7. Banerjee, S.; Adhikari, M.; Kar, S.; et al.: Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud. Arab. J. Sci. Eng. **40**, 1409 (2015). doi:10.1007/s13369-015-1626-9
8. Policy Based Management, IETF, Internet Engineering Task Force, Policy Working Group. http://www.ietf.org/html.charters/policy-charter.html (2009)
9. Dursun, T.; Örencik, B.: POLICE: A Novel Policy Framework. Lecture Notes in Computer Science, LNCS 2869, pp. 819–827 (2003)
10. Dursun, T.: A Generic Policy Conflict Handling Model. Lecture Notes in Computer Science, LNCS, 3733, pp. 193–204 (2005)
11. Casanova, H.; Giersch, A.; Suter, F.: Versatile, scalable, and accurate simulation of distributed applications and platforms. J. Parallel Distrib. Comput. **74**(10), 2899–2917 (2014)
12. Krauter, K.; Buyya, R.; Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. Softw. Pract. Exp. **32**, 135–164 (2002)
13. Sinha, P.K.: Distributed Operating Systems: Concepts and Design. IEEE Press, New York (1997)
14. Buyya, R.; Murshed, M.: GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr. Comput. Pract. Exp. **14**, 1175–1220 (2002)
15. JFreeChart graphic library web portal. http://www.jfree.org/jfreechart (2009)
16. Dursun, T.; Dağ, H.: HeteroSim: heterogeneous simulation framework. In: Communications and Networking Simulation Symposium, CNS (2009)
17. Java Message Service Concept, Oracle (2016). http://docs.oracle.com/javaee/6/tutorial/doc/bnceh.html (2016)
18. Depoorter, W.; De Moor, N.; Vanmechelen, K.; Broeckhove, J.: Scalability of grid simulators: an evaluation. In: 14th EuroPar Conference, Vol. 5168 of LNCS, pp 544–553. Springer (2008)
19. Dursun, T.; Dağ, H.: A generic simulation model for high performance computing systems. In: International Conference on Modeling, Simulation and Visualization Methods (MSV14), Las Vegas (2014)
20. Little, M.C.: JavaSim User Guide, pub.rel. 0.3 (2004). http://javasim.ncl.ac.uk/ (2009)
21. Miller, J.A.; Nair, R.S.; Zhang, Z.: JSIM: A JAVA-based simulation and animation environment. In: 30th Annual Simulation Symposium (ANSS97), Atlanta (1997)
22. Tyan, H.Y.; Hou, C.J.: J-Sim JavaSim: a component based compositional network simulation environment. In: Western Simulation Multiconference (2001)
23. Healy, K.J.; Kilgore, R.A.: Silk: a Java-based process simulation language. In: Winter Simulation Conference, pp. 475–482 (1997)
24. Howell, F.; McNab, R.: SimJava: a discrete event simulation package for Java with applications in computer systems modelling. In: First International Conference on Web-Based Modelling and Simulation, San Diego (1998)
25. Ranganathan, K.; Foster, I.: Decoupling computation and data scheduling in distributed data-intensive applications. In: 11th IEEE International Symposium on High Performance Distributed Computing, HPDC-11, 23–26 July, pp .352–358 (2002)
26. Dumitrescu, C.L.; Foster, I.: GangSim: a simulator for grid scheduling studies. In: IEEE International Symposium on Cluster Computing and the Grid, CCGrid, vol. 2, 9–12 May, pp. 1151–1158 (2005)
27. Dumitrescu, C.L.; Foster, I.: Usage policy-based CPU sharing in virtual organizations. In: Fifth IEEE/ACM International Workshop on Grid Computing, pp. 53–60 (2004)
28. Legrand, A.; Marchal, L.; Casanova, H.: Scheduling distributed applications: the SimGrid simulation framework. In: 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2003), Tokyo, 12–15 May (2003)
29. Syed Abudhagir, U.; Shanmugavel, S.: A novel dynamic reliability optimized resource scheduling algorithm for grid computing system. Arab. J. Sci. Eng. **39**, 7087 (2014). doi:10.1007/s13369-014-1305-2
30. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011)
31. Bux, M.; Leser, U.: DynamicCloudSim: simulating heterogeneity in computational clouds, DynamicCloudSim. In: SWEET '13,

2nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies (2013)
32. Ostermann, S.; Prodan, R.; Fahringer, T.: Dynamic cloud provisioning for scientific grid workflows. In: 11th ACM/IEEE International Conference on Grid Computing, pp. 97–104 (2010)
33. Nunez, A.; Poletti, J.V.; Caminero, A.: Design of a new cloud computing simulation platform. In: 11th International Conference on Computational Science and Applications, pp. 582–593 (2011)
34. Taufer, M.; Kerstens, A.; Estrada, T.; Flores, D.; Teller, P.J.: SimBA: a DES for performance prediction of volunteer computing projects. In: 21st International Workshop on Principles of Advanced and Distributed Simulation, pp. 189–197 (2007)
35. Estrada, T.; Taufer, M.; Reed, K.; Anderson, D.P.: EmBOINC: an emulator for performance analysis of BOINC projects. In: Workshop on Large-Scale and Volatile Desktop Grids (PCGrid) (2009)
36. Kondo, D.: SimBOINC: A Simulator for Desktop Grids and Volunteer Computing Systems. http://simboinc.gforge.inria.fr/ (2007)
37. Basu, A.; Fleming, S.; Stanier, J.; Naicken, S.; Wakeman, I.; Gurbani, V.K.: The state of peer-to-peer network simulators. ACM Comput. Surv. **45**(4), 46:1–46:25 (2013)
38. Montresor, A.; Jelasity, M.: PeerSim: a scalable P2P simulator. In: 9th International Conference on Peer-to-Peer, pp. 99–100 (2009)
39. Baumgart, I.; Heep, B.; Krause, S.: OverSim: a flexible overlay network simulation framework. In: 10th IEEE Global Internet Symposium, pp. 79–84 (2007)
40. Gil, T.M.; Kaashoek, M.F.; Li, J.; Stribling, J.: P2PSim, A Simulator for Peer-to-Peer Protocols. http://pdos.csail.mit.edu/p2psim/ (2005)
41. Garcia, P.; Pairot, C.; Mondejar, R.; Rallo, R.: PlanetSim: a new overlay network simulation framework. In: 4th International Workshop on Software Engineering and Middleware (SEM), Vol. 3437 of LNCS, pp .123–136. Springer. (2004)
42. Kilgore, R.A.; Healy, K.J.; Kleindorfer, G.B.: The future of Java-based simulation. In: Winter Simulation Conference, pp. 1707–1712 (1998)
43. Page, E.H.; Moose, R.L.; Griffin, S.P.: Web-Based simulation in SimJava using remote method invocation. In: Winter Simulation Conference, Atlanta, 7–10 Dec (1997)
44. Jacobs, P.H.M.; Lang, N.A.; Verbraeck, A.: D-SOL; a distributed Java based discrete event simulation architecture. In: Winter Simulation Conference, pp. 793–800 (2002)
45. Herrmann, J.W.; Conaghan, B.F.; Lecordier, L.H.: Understanding the impact of equipment and process changes with a heterogeneous semiconductor manufacturing simulation environment. In: Winter Simulation Conference (2000)
46. Kim, Y.J.; Kim, T.G.: A heterogeneous simulation framework based on the DEVS BUS and the high level architecture. In: Winter Simulation Conference, 13–16 Dec (1998)
47. Barr, R.; Haas, Z.J.; Van Renesse, R.: JiST: An efficient approach to simulation using virtual machines. Softw. Pract. Exp. **35**(6), 539–576 (2005)
48. Kaner, C.: Cem Kaner on scenario testing: the power of "what if" and nine ways to fuel your imagination. In: Software Testing & Quality Engineering (STQE) (2003)
49. Java Management Extensions (JMX) Technology, Oracle. http://www.oracle.com/technetwork/articles/java/javamanagement-140525.html (2016)