Abstract

Most of the academic research on High Performance Computing (HPC) systems has been conducted with simulation platforms because they are generally too expensive and hard to construct systems. Moreover, the research has been generally focused on a specific type of HPC system. This paper however, introduces a general-purpose simulation model that can be used for constructing simulations of the most well-known HPC system types.

In this paper, we propose a new approach that leverages usage of simulation systems for constructing hybrid simulations. In order to arrange heterogeneous simulation executions, the simulation tools are required to allow easy and fast creation of simulation sessions by employing real-time software components beside simulation codes. Although there have been considerable amount of research activity in simulation community, the current simulation tools are not capable of supporting such a cooperation between components working in real-time and simulation-time. We introduce HeteroSim, a simulation model that can execute discrete event simulations by employing both simulation entities and real world software entities. This model offers great potential for many research areas. For instance, we are able to rely on this model in order to build simulations that combine both the simulated elements of an HPC system and previously implemented elements of our Policy Based Management (PBM) framework [1][2]. In this way, it may be possible to study the efficiency and usability of PBM concept on the management of HPC systems.

1. INTRODUCTION

Having been the subject of much research, simulation systems provide a way to evaluate distributed systems under different scenarios such as varying the number of resources and users. In real environments, it may not always be possible, to perform evaluation in a repeatable and controllable way for different scenarios. Because the status of entities continuously varies with time and it may not be possible to control activities of them. Simulation systems, however, enable to perform various studies, such as behavior and performance without building the actual system. In contrast to analytic models, simulation models represent the run time behaviors of the real systems being worked on. In some cases as listed in [3] simulation is preferable to analytical modeling.

HPC systems are expensive and generally have a distributed nature. It is not always possible to take the opportunity of academic research and performing experiments with them. Therefore, simulations are essential for carrying out research experiments in HPC systems. Thus, a number of simulation tools have already been developed. On the other hand, most of these simulation tools have dedicated only to a specific type of HPC system or a sub set of its components. Thus, there is no generic tool to support simulations for different types of HPC systems. Moreover, the existing tools are not flexible and modular enough, because they are not a result of a research effort to develop a generic model for the implementation of different HPC scenarios. Therefore, we have developed a universal simulation model for easy building simulations of well-known HPC systems in order to fill this gap. Section 3 explains the proposed model with a set of execution results.

For more realistic results by involving more accurate data, simulations may need to interact with real-time systems. Based on the degree of human and real system participation, there are three main styles of simulation. These styles are described by [4] as follows:

- **Live simulation** involves humans for operating real systems in order to rehearsal or practice with “go-to-war” systems. As in the real world, time is continuous. Testing a real car battery using an electrical tester is an example of live simulation.

- **Virtual simulation** includes simulated systems operated by humans. Time is in discrete steps. Human is in a central role (e.g. a flight simulator).

- **Constructive simulation** refers to classical computerized simulation model that involves simulated people operating simulated systems. Science-based simulations are constructive in nature. Humans just make inputs to such simulations and are driven by the proper sequencing of events, but have no effect on the outcomes.
Most of the research activities in simulation community have generally focused on simulation frameworks in these three styles. As criticized in [5], this categorization suffers from no clear division between categories and a missing category for simulated entities interacting with real equipment (e.g. smart vehicles). Besides, current simulation tools do not sufficiently support such cooperation between real-time and simulation-time software components. Therefore, tools that enable easy and fast creation of simulations employing both real world applications and simulation codes would be useful. In this paper, as a solution to fill this gap, we also introduce a model rather falling into that missing category. In our model, interactions are allowed between not only real world systems and simulated entities but also humans and the real world systems. That’s both simulated people and humans are interacting simultaneously with the same real systems.

A proof-of-concept implementation of the infrastructure for building simulations by involving existing real systems having JMX [6] or JMS-based [7] communication interfaces has already been accomplished. As the second contribution, Section 3 explains our approach to facilitate building of hybrid simulations.

Another aim of our study is to show how the PBM concept [8] responds the needs for effective management of the HPC systems and thus to prove the usability and performance raising effect of our already PBM framework [1],[2]. Therefore, in this study, we have integrated our simulation model with the POLICE PBM framework as described in Section 3.3.

Rest of this paper is organized as follows. Section 2 introduces some background for the reader. The model is discussed in Section 3. Section 4 includes evaluation of the related work and the last section presents our conclusions.

2. SIMULATION BACKGROUND

Simulation concept has been used to model and evaluate real world systems and understand their behaviors. The output of the simulation indicates how a real system behaves without need for its actual implementation.

In recent years, simulation has emerged as an important research area and many simulation tools and technologies have been developed. The simulation tools enable constructing repeatable and controllable environments for feasibility and performance studies.

2.1. Simulation Models

A simulation abstracts all the entities and their time dependent interactions in the real system. Simulation models are generally encoded as discrete event-driven programs. Events are time-stamped messages processed in their temporal order as the simulator progresses through simulated time interval [9]. However, there are many other concepts for the realization of the simulation models. Buyya provides a collection of these concepts and a comprehensive categorization of simulation tools according to an inclusive set of criteria [10]. In this section, we emphasize some classifications related to our work.

Each simulation tool has a simulation engine to execute the simulation model. Figure 1 depicts the Buyya’s taxonomy for the mechanics of simulation engines according to their behavior against time. These categories are explained in [10] as follows:

- In a continuous time simulation, state changes occur continuously with time; such systems are usually described and solved by sets of differential equations.
- In a discrete-event/time simulation (DTS/DES), state variables change at instants in time and the system is only considered at these time points (the observation points). A DTS is further subdivided into a trace-driven, time-driven or event-driven simulation. A trace-driven DTS proceeds by reading a set of events that are collected before from another environment for modeling a system that has executed in another environment. A time-driven DTS advances by fixed time increments and is useful for modeling events that occur at regular time intervals. An event-driven DTS advances by irregular time increments and is useful for modeling events that may occur at any time.
- In a hybrid (continuous time-discrete event) simulation, the time is (conceptually) continuous and the observation period is a (equally or arbitrarily spaced) real time interval. The discrete changes in the system state take place at these event times. In between consecutive event times the system state may vary continuously.

The Buyya’s modeling framework criterion [10] describes how the target systems are modeled and how the events are scheduled. Buyya defines three categories for scheduling operations of a simulator as follows:
• An **entity-based** (Process-oriented) modeling involves simulation entity concept in order to logically encapsulate processes of target system. Entities communicate with others via messaging to perform their tasks. HeteroSim falls into this category.

• In an **event-based** modeling framework, there is a procedure associated with each type of event in the system. The system performs the action required to handle that type of event and every time such an event occurs the same action is performed.

• Hybrid models of the categories above are possible.

### 2.2. Simulation Tools

In order to achieve realistic simulations, a simulation model has to be as identical as possible to simulated system. For satisfaction of this requirement, simulation tools should support object-oriented approach, which is well-suited for modeling real world entities in software. There are already various frameworks that combine the concept of objects with the concurrent computation in which simulation entities run concurrently. This process oriented approach is addressed by most of the discrete event simulation tools. Because of its strong support for multi-threaded programming and native schedulers, many simulation tools\(^1\) have been developed with Java language [11].

**JavaSim** [12] is an open source tool developed at the Department of Computing Science, University of Newcastle upon Tyne for building discrete event process-based simulation. JSIM [13] is another Java-based simulation and animation environment supporting Web-Based Simulation. Simulation models may be built using either the event or the process oriented approach. In addition, a visual designer allows process models to be built graphically.

**J-Sim** [14] is a Java-based sequential network simulator developed at the Department of Computing Science of the Ohio State University. It is based on the component-based software architecture, Autonomous Component Architecture (ACA). Unlike JavaBeans, CORBA, and COM/DCOM, the components in J-Sim are loosely coupled, communicate with one another by “wiring” their ports together (Figure 2). J-Sim can be used for both discrete event simulation and real-time process-based simulation. **Silk** [15] and **SimJava** [16][17] are two early Java-based libraries for process-oriented discrete-event simulation. **Silk** is a complete package including JavaBeans components for visual modeling and a variety of methods for entity generation, resource scheduling and output post-processing.

![Component Diagram](image)

**Figure 2.** Simulation elements

**SimJava** is a general-purpose package that is a Java implementation of a C++ library called HASE++ (Hierarchical computer Architecture design and Simulation Environment). A SimJava simulation contains a number of simulation entities each of which runs in parallel in its own thread. As in J-Sim, these entities are connected to each other via ports and can communicate by sending and receiving passive event objects through these ports.

### 3. PROPOSED SIMULATION MODEL

#### 3.1. Generic HPC Simulation Model

The first members of HPC family are super-computers (or parallel systems), which are generally composed of a great number of processors, large memory modules and a high speed communication bus. Due to high cost of supercomputers, an alternative family, named as Distributed Computing Environment (DCE), has been emerged. As being member of DCE family, networked virtual supercomputers (meta-computers) and distributed network computing (NC) systems are virtual computers that are composed with heterogeneous machines for sharing their resources for a common purpose. Grids, however, are very large scale, Internet wide, distributed NC systems, whose machines belong to different administrative domains. They combine characteristics of both distributed and parallel systems.

A generic, abstract model that can be used for definition of both distributed computing environments (DCEs) and Grids is already proposed in [18]. This abstract model contains several functional units and four interfaces: resource consumer interface (RCI), resource provider interface (RPI), resource manager support interface (RMSI), and resource manager peer interface (RMPI). The resource is a shared entity or capability that is employed to fulfill scheduled job or resource requests. It could be a machine, network, or some service that is under control of a Resource Management System (RMS). An RMS is defined as a service that is provided by a distributed NC system that manages a pool of resources. The resource consumers that

---

\(^1\) In the academical simulation world relying on Java language, there have been naming ambiguity. Many tools have names including words of “Java” and “Sim”, such as JavaSim, SimJava, and J-Sim.
interface via RCI can be either actual applications or another RMS that represents a ‘higher’ layer. It is an agent that controls the consumer. The resource provider (broker) that interfaces via the RPI 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 RMSI. The RMPI is intended for interaction with other RMSs and may support several protocols including resource discovery, resource dissemination, trading, resolution, and co-allocation. Figure 3 shows the abstract model and a sample system with multiple interconnected multiple levels RMSs.

![RMS interface diagram](image)

**Figure 3.** An abstract RMS model and its use for Grid modeling [18]

Although many HPC architectures could be specified with this abstract model, to be generic enough it needs some improvements such as support for distributed ownership of the Grid resources. Traditional DCEs are generally established in a single administrative domain and accept jobs of clients that belong to the same organization. Thus, job priorities could be simply used as QoS mechanism in traditional DCEs. However, in a Grid, providing QoS to clients from different administrative domains is more challenging. The issues such as access privileges of jobs, and resource requirements shall also be taken into account [19]. Thus, different administrative domains concept and handling semantic relationships between components of HPC systems should also be supported in this model.

Therefore, as the first contribution, we developed a general purpose, policy-based manageable simulation framework for HPC systems by realizing an enhanced version of the abstract model above. The following main improvements and simplifications are applied to this abstract model:

- Adding observation capability to Resource/Job Monitor in order to observe resources of a peer RMS.
- Adding an Inter-Domain Interface for interaction with RMSs belonging to different administrative domains by using various protocols.
- For simplification purpose, by combining some functionally close modules into one module, an easily implementable model has been obtained.
- To achieve policy based management capability for each component of RMS, adding PEP (Policy Enforcement Point) interface as a support service. Therefore, each component of the HeteroSim RMS is instrumented for policy enforcement.

Each component of our abstract RMS simulation model shown in Figure 4 can be replaced with a different implementation thanks to the abstract interfaces we implemented. Furthermore, by connecting our RMS nodes, hierarchically or peer-wise and then grouping them into administrative domains, it is possible to construct policy based manageable simulations of centralized, distributed and hierarchical HPC systems. Figure 5 shows a sample HPC scenario including mixed types of peer and hierarchical RMS relationships that is built with triple RMS nodes.
Figure 4. RMS abstraction model (revised version of the model proposed by [18])

Figure 5. A sample HPC scenario established with HeteroSim RMS nodes
Another significant feature of the HeteroSim framework is its support for hybrid simulation in which both simulation codes and real application software can participate. In this manner, for example, we are able to make our already developed POLICE PBM application (real world application) communicates with entities of HeteroSim simulations.

As the first step of the development of HeteroSim platform, we have surveyed academic and open-source simulation tools that offer process-based discrete event simulation base, and then we have selected GridSim [20] toolkit as the starting point of our development work. Having larger academic background, the GridSim has the benefits of relative simplicity, extensibility, and Grid based HPC domain support. Then, we have developed a Java-based discrete-event simulation toolkit called HeteroSim that is a redesign of GridSim Toolkit [20] from a new point of view. The refactoring results in the following main improvements:

- Support for modeling and simulation of heterogeneous types of entities, from both simulation world and real world was added.
- More detailed resource modeling (finer granularity). The resource represents a set of machines in GridSim framework and is managed by a Scheduler. However, in HeteroSim each machine can be modeled as a separate resource. By means of this capability, the resource concept is changed from “Resource including Machines with Scheduler” to “the machine with a number of PEs (Processing Elements)”.
- Management interface for POLICE PEP was added and enforcement activities were implemented. Policy based manageable versions of all RMS components were developed. Therefore, it is possible to change behaviors of simulation entities with policies.
- Local and remote job separation was added
- Added new entity types of Trader, RMS, Dispatcher, and Local Information Service. Therefore, complexity of the Scheduler is decreased by moving the functionalities to where they must exist. Jobs are not submitted to directly Scheduler anymore but to the Trader.
- Automatic RMS topology construction according to configuration specified before simulation start.
- Support for RMS hierarchy deeper than two levels was added.
- Mechanism for adjusting the simulation execution speed mechanism was implemented
- Entity synchronization and multi-thread support were improved.
- Allocation of jobs to multiple PEs belonging to different resources is now possible. Moreover, instant mapping a job to multiple RMS resources is also possible.
- For resource allocation, a number of predefined strategies such as LONGEST_FIRST, LOCAL_FIRST, EQUAL, LEAST_REMAINING_FIRST and REMOTE_FIRST can be triggered via policies.
- A new resource addressing schema is used such as HPC1/RMS2/Resource6.
- Virtual Organization (VO) attribute was added to the Resources and to the Users entities.
- ResourceLoader was developed and tested with real workload traces from various HPC centers.
- Mechanisms implemented for on the fly calculation of metrics based on HPC system, RMS and users:

<table>
<thead>
<tr>
<th>metrics</th>
<th>based on</th>
<th>RMS</th>
<th>User</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AET (Average Execution Time)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>AWT (Average Waiting Time)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>AJL (Average Job Length)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>TJL (Current/Cumulative Total Job Length)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MakeSpan</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max, Min, Average Job length</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System workload ratio</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Calling the added metrics within the policies was supported.
- Execution results are shown with charts implemented with JFreeChart [21] library for each statistical metric defined above. In addition, ResourceAllocation chart was also implemented. The sample charts are shown in Figure 6 and Figure 7.

Figure 6. HeteroSim resource allocation history chart

HeteroSim follows the process-oriented approach of SimJava, where each SE (Simulation Entity) can be considered as a separate process. The SEs are the Java
objects which have an independent thread of control associated with them (pseudo-parallel execution) and they use event-based messaging.

There is a central event queue, called FEL (Future Event List) that contains timestamp ordered events and is operated by a scheduler. The scheduler observes the FEL and finds the event with the smallest time stamp and invokes the entity 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.

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

### 3.2. Heterogeneous Simulation Architecture

As shown in Figure 8, HeteroSim architecture consists of four types of components: an adaptation layer, a simulation runtime, simulation entity and real world entity [22]. The real world entities (REs) represent external real time systems and are able to interact with the simulation entities (SEs) during a run. The actions of REs can influence the outcomes of the simulation. Controversially, the SEs may also cause state changes in REs. This bidirectional interaction ability makes it possible, for example, to create SEs including test logic for testing the REs or to exploit existing REs to build simulations quickly instead of implementing them as SE.

The interactions between simulation entities and real system components occur through a common communication mechanism, so called Adaptation Layer (AL) in our model. The AL facilitates expansion of simulation to external real world applications. Instead of establishing direct connection from SEs to the real world applications, AL communicates with the real world applications on behalf of SEs. It supports multicast, broadcast and unicast communications.

Similar to SEs, the REs are also parts of the simulation sessions. The REs can be either an ordinary or server type entity. The Server type RE (SRE) can be any kind of server application such as an RDBMS Server, an SNMP Server, a Web Server, a Messaging Server, and so on that SEs or other REs can consume its service. For use with different types of external applications, the AL provides various alternative communication technologies for the SE and RE interaction as shown in Figure 9.
In order to obtain a communication path, a proxy entity (so-called Adapter Entity) for each RE is automatically created and registered in a routing table when the simulation starts. The proxy entities are responsible for translating the communication between SEs and REs transparently. While the proxy entities communicate with REs via proper technology such as SMTP, JMS [23], and RPC (Remote Procedure Call), they also play SE role in order to interact with SEs (i.e. via messaging). Thus, the codes of SEs that interact with REs do not include any code which is specific to the communication technology.

A sample simulation scenario for the case of the Figure 10 can be arranged as shown in Figure 11. In this sample scenario, the SE$_2$ should select the path over AL so that the messages are to be transmitted to SE$_1$ via communication layers associated to the Technology-X.

**Figure 10.** Obtaining a more accurate simulation

For simplicity purpose, the simulation frameworks generally focus on modeling only application-layer level activities of real world applications. However, accurate simulations are achieved only by simulating as much communication layer as possible, in addition to application-layer behaviors, as depicted in Figure 10. For increasing accuracy of simulations, the AL can also be used for communication between SEs.

Because the AL can already employ the original protocol stacks related to communication technologies within simulation sessions, HeteroSim can also implicitly help to simulate the communication mechanisms through which the REs communicate without writing any additional code.

Another capability provided to users by HeteroSim is the BeanShell [24] scripts based graphical user interface for interacting with simulation run.

For the validation purpose, modeling results must be compared with the corresponding field observations to ensure that the model realistically represents the real system. Therefore, we have used various real workload traces obtained from well-known supercomputing centers to test our HPC simulation model.

### 3.3. Issue of Simulation Time

As summarized in [9] and [5], there are two main types of simulation execution model according to time treatment:

- **Logical-time simulations** explicitly model the passage of time inside and allow the rate of their advancement to be dictated by the granularity of simulated events. This category is also named as simulation time execution (dual/application clock) in which the progress of time depends on the progress of the application [9]. The simulation time does not advance to the next discrete time point unless all codes for the current simulation time have been executed. HeteroSim falls into this category. Logical-time simulations have the classical discrete event or continuous simulation data structures and algorithms.
Real-time simulations using the real clock values to drive their execution are further divided into two subcategories [9]:

- In the actual time execution (standard Java execution) simulations, the progress of the application and the passing of time are independent. Due to unpredictable parameters such as interrupts, cpu load, and I/O events, the program can advance at a variable rate. Moreover, for the interpreted languages, program is not guaranteed to progress timely due to survival activities.

- Real time execution: the application progress depends on the passing of time. The runtime guarantees that instructions will meet given deadlines. Real-time execution simulations resemble real-time systems and their execution is measured by hertz frequency.

The HeteroSim simulations permit interactions with the real-world applications and the events related to them are processed by REs in the simulation time. Therefore, the real time clock should also be taken into account, in addition to the simulation clock. Otherwise, the execution of simulation may cause inconsistencies for REs. During an interaction between external applications and SEs, any delay in simulation side may affect the operations of real world systems negatively, or vice versa. As an example, the enforcement of scheduling policies including Date/Time constraints may not be performed in a consistent way due to two different time domains.

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, changing execution rate at some ratio to real time (for example, by injecting specialized events on the event list or by adjusting frame rate), degrading or abandoning next-state computation, ignoring the delay and, if capability exists, attempting to catch up later by running faster are the possible choices for keeping synchronization with real time.

Simulation time and real time combination is still an open issue for HeteroSim. However, in the current implementation, the events related to the 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 interaction with real time applications.

On the other hand, the authors of [9] propose a new approach, called JiST (Java in Simulation Time) for constructing discrete-event simulator by embedding simulation time semantics into the standard Java language to achieve performance. Thus, it can be possible to write simulations by using simplified JiST API wherein the simulation codes may contain functions related directly to real time. Then, the JiST framework modifies the standard Java-byte codes of the simulation programs written in Java and embeds its execution semantic. As a future work, we plan to study to involve the actual time concept mentioned in [9] to solve the possible performance problem and to help the solution of dual-time-domains problem.

3.4. Using HeteroSim in PBM simulation

The number of parameters that should be configured in an optimal way and consequently management overhead increase in parallel with capabilities and size of traditional HPC systems. Especially, due to highly heterogeneous components, distributed ownership of the resources, and very dynamic conditions, Grids have much more management requirements. In order to deal with this need, employing PBM tools should be the first alternative that comes to mind. The easily handling of processes such as access control for resources, allocating jobs, and sharing resources, according to management goals can only be realized with policy based management of HPC systems. In fact, for the management of HPC systems many tools have been developed as a built-in or supplementary mechanism. Despite policy and strategy words are addressed frequently by these tools, they do not actually fulfill the characteristics of a real PBM tool as defined in Literature. Therefore, for the management of HPC systems in conformance with PBM concept, we have intended to use the output of our previous study which is a general purpose PBM framework, so-called POLICE [1] [2] whose architecture is shown in Figure 13.

The PBM offers an effective way for management of systems wherein the desired system behavior is specified as policies by administrators. Containing high-level, human-friendly terms, policies are rules to administer, manage, and control the resources. The PBM system automatically translates policies into commands and configuration parameters that are understandable to the managed devices.

![Figure 12. General Architecture of a PBM System](image-url)
The information model and architectural blocks published by IETF to describe network policies, and services [8], [25] have been commonly accepted in the PBM community (Figure 12).

The policy management applications allow administrators to specify the policies, translate the input into a common format and store them in the policy repository. A PDP, on the other hand, retrieves policies from the policy repository, interprets the policies, sends them to the PEPs for enforcement and replies the policy decision requests from PEPs with policy decisions. PEPs (residing on managed resources) act according to the PDP’s decisions and actual system conditions in order to enforce policies.

HPC systems need to be evaluated under different scenarios such as varying the number of resources and users. However, in a real HPC environment, it is hard, and perhaps even impossible, to perform evaluation in a repeatable and controllable manner for different scenarios. Because the availability of resources and their loads continuously vary with time and it is impossible to control activities of users especially the ones belonging to different administrative domains. Moreover, HPC systems are very expensive and it is not always possible to have opportunity for academic research with them. That’s why, we selected to use simulation instead of real HPC systems. Therefore, we have developed the generic simulation model for HPC systems whose details are given in Section 3.1.

Then, thanks to our simulation model allowing simulation entities to interact with real software applications, we have been able to exploit our PBM framework in order to manage the simulated HPC components with policies. In this simulation setup, POLICE components, as the real application elements, interact with the simulation entities representing HPC components. In this manner, we are able to show the effectiveness of POLICE framework on HPC throughput and the usability of POLICE for management of HPC systems.

RMS nodes typically contain various functionally related components. That’s why, with our RMS model, not only scheduling components but also all functions and components of an HPC system can be managed as a whole. This is a must for consistent management of HPC systems. Otherwise various inconsistencies may occur. For example, in case for which access control is performed by policies and scheduling is not, if the jobs are assigned the resources that they do not have access privilege, this fact remains unknown until the jobs are sent to the related RMS. All management functions shall be performed with enterprise wide policies which are collection of access control policies, scheduling policies, and so on.

Thanks to this PEP expansion, RMS hierarchies belonging to different administrative domains can be constructed. By this way, site autonomy can be achieved. Each management server (PDP) allows specification of policies for its own administrative domain. Inter-domain resource sharing can be performed according to the policy negotiation, SLA, etc. This negotiation concept is out of scope of this paper but it is noted as a future work.

By employing the mechanisms mentioned above, only resource allocation policies can be defined. However, semantic relationships (acyclic task graph, etc.) between resources and jobs that may pose limitations for resource allocation and job execution also must be considered. Handling this kind of semantic relationships for an error free resource sharing can be performed with E-Code concept which is already provided in POLICE PBM framework [1]. For this purpose, HPC system administrators can specify the actions for semantic constraints as E-Code that must be executed during resource management.

In order to enforce policies, we have integrated HeteroSim environment with POLICE framework by using the AL mechanism of HeteroSim. Figure 15 shows interconnection schema with a sample scenario including an RMS and its dedicated PPEP. In this way, the real world components of POLICE interact with the SEs representing HPC components. Figure 15 includes more details on integration schema of POLICE PEP and HeteroSIM.
Figure 14. The information model for HPC modeling

Figure 14 shows the summary information model of our HPC simulation framework. Each RMS is associated with a PAE (PEP Adapter Entity) that plays proxy entity role for the communication with POLICE PEP which is designed for this RMS. The managed node to be controlled by PEP is the RMS and the Managed Objects (MOs) are the components of this RMS. During simulation startup, the PAE registers the MOs with the PEP. The primary actor objects are the users and virtual organizations whereas the target objects are the resources. With this model, the following components and processes can be referenced within POLICE policies:

- Job (job start, stop, remove, migrate, priorities and scheduling)
- Job queues (strategy, priority, etc.)
- Consumers (Users)
- Resources
- Reservations
- Scheduler
- Broker (for QoS/SLA)
- Trader
- Resource allocations
- Access Control

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

- `rms1.scheduler.awt(user)`
- `rms1.scheduler.awt()`
For example, the following POLICE policy changes the job selection strategy followed by the scheduler component of the RMS3 when the average job waiting time for User1 is greater than zero so that the longest jobs could have priority.

```
setJobSelectionStrategy("LocalFirst")
```

![Sample policy for HPC management](image)

**Figure 16. Sample policy for HPC management**

4. RELATED WORK

In recent years, modeling and simulation has emerged as an important research area and various standards, methods, tools and technologies have been developed. However, there have been quite a few studies [11][26] dealing with bringing simulation and real world codes together in a simulation session. Except few tools which are already hard to integrate with external real-time systems, most of the simulation tools contain only simulation codes. A mature solution has not appeared yet. Distributed SimJava [26] is the model proposed to make distributed simulation sessions possible. It extends SimJava with distributed simulation capability by employing the Java RMI (Remote Method Invocation). In this way, the SEs belonging to different simulation nodes may interact with each other. It is also possible for the SEs to interact via RMI interface with the real world components which are actually running the real application codes. The main difference of this model to our approach is that it requires the real-world participants of the simulation to implement the RMI interface. Moreover, the real applications can only be seen as external systems providing services to be called by SEs and they cannot participate in a simulation as a simulation entity.

Another study [27] also proposes a distributed simulation model including a DES-based remote simulation service and a representation service interface for real applications to interact concurrently with that system. Simulation objects interact with remote, asynchronous subscribed clients in order to produce representations of the simulated system. Although this model provides mechanisms to external users for either passive (the user only visually monitors the output of the simulation) or active (the user is able to interact with the model during a run and will then influence the outcomes) interaction with a simulation, it does not provide any software interface for bi-directional communication between simulation codes and existing real applications. The simulation service accesses external information systems just to obtain data related to the simulation.

On the other hand, there are several works in Literature which bring heterogeneous and simulation words together as we do. However, none of them does it for the same purpose as ours. The authors, for instance, proposed [28] a simulation model consisting of a heterogeneous simulation environment (HSE) that integrates a variety of simulation and analytical models of a manufacturing system. The HSE executes each model sequentially so that the output from one model can be used in the input for the next model.

Another work [29] describes a heterogeneous simulation framework in which a collection of heterogeneous simulation models such as conventional simulation models and the DEVS (Discrete Event System Specification) models communicate with each other via DEVS bus. It includes a DEVS/CSIM simulation protocol converter for communication between DEVSS model and conventional model. This work only focused on simulation time domain but it didn’t deal with the adaptation problem of time domains.

The last example [30], which uses the same name as our model, describes a simulation model named, HETERO SIM to replicate the field conditions of heterogeneous (car) traffic flow that does not follow traffic lanes. That’s why they named their model as heterogeneous.

For simulation of HPC systems, a number of simulation tools have already been developed, such as GridSim [20], SimGrid [31], and MicroGrid [32]. However, most of these simulation tools have dedicated only to a specific type of HPC system or a sub set of its components such as scheduler, broker, and so on. Thus, there is no generic tool to support simulations for different types of HPC systems.

For example, GridSim[20] and ChicSim [33] data grid
simulation system, were developed only for simulation of Grid systems. Ganglia discrete event simulator [34], popular Ganglia Cluster Monitor [34], and GangSim simulator [35] are all tools having rather focus on the cluster-wide scheduling. Moreover, they are not flexible and modular enough, because they are not a result of a research effort to develop a generic model for the implementation of different HPC scenarios.

5. CONCLUSION AND FUTURE WORK

Simulation designers may need to involve real-time systems into simulations for several purposes such as reducing building time of simulations, achieving more realistic simulations, arranging simulations of partially implemented systems, and so on. Therefore, it would be useful to have simulation models that enable easy and fast creation of simulation sessions by employing real software components besides simulation codes. Current simulation tools are not sufficiently capable of supporting such cooperation. Our proposition in this paper, however, is able to overcome this shortage with its extensible model for heterogeneous simulations. When preparing a HeteroSim simulation, the implementation of the interactions between SEs and REs are hidden from the modeler’s view. Unlike the other proposed systems, in our model, starting connections from both simulation and real world sides is possible.

Having an extensible Adaptation Layer makes our proposition suitable to be used with any real world application. For the time being, we have only implemented and tested a JMS (Java Message Service) [23] based proxy entity for real world applications. Other types of proxy entities can be easily added to the architecture.

Many research areas can take the advantage of the infrastructure proposed in this paper. Scenario-based software testing [36] would be an interesting area to apply our model. For a typical use, all testing scenario can be easily implemented within simulation entities (SEs). Then, through the AL, the SEs can interact with the real world applications to be tested.

Simulations are essential for research experiments in HPC systems. We have developed a universal simulation model for building simulations of HPC systems. With this model, well-known HPC architectures can be created easily either by specifying the configuration of the target architectures manually or by providing workload trace files. Beside the manual experiments, we have also successfully performed simulations with the workload traces of various supercomputing centers. We have been able to execute simulations and observed their results on the graphical charts produced automatically during the simulations. This universal model also contains structures related to policy based management so that by defining policies in POLICE PBM system, we have investigated the effects of them on the behaviors of the simulation entities. Our model can be used for future research on both PBM and HPC domains.

Cloud computing is a new and emerging form of distributed resource sharing. Thus, the same challenges as the Grids have to be overcome. The following list is not exhaustive, but for the majority issues to be handled in cloud domain:

- Cloud service auditing, monitoring, and metering.
- Mobility management in cloud scenarios.
- New models and paradigms for cloud service management.
- Novel and emerging standards for interoperability between clouds and management of cloud federations.
- QoS/QoE and SLA management in the cloud.
- Secure and private management of cloud data.

Thus, enlarging our model to allow modeling the cloud computing architectures would be a good research opportunity.

On the other hand, our model can be extended and investigated further. We plan to concentrate on especially in the following issues:

- Developing more types of proxy entities for the AL related to mostly used technologies area such as RMI-IIOP, JDBC, and so on,
- Adding distributed simulation support,
- Performance analysis of Adapter Entities,
- Further exploration of the problem of integrating applications running in different time-domains, and investigating solution alternatives,
- Finding a way to synchronize the events occurred in different time domains such as real clock and simulation clock.
- Testing how the proposed model meets the requirements of various domains and applications,
- Improving existing monitoring tools in order to follow visually the activities performed within AL.
- Developing SE templates and structures for scenario based software testing,
- Taking the advantages of heterogeneous simulation sessions for software testing might be a study target. If the test logic is prepared as the simulation codes interacting with the real-time application being tested, it could be possible to benefit from the simulation tools.
- Real world implementation of the RMS node whose simulation model described in this paper.
- Providing a visual tool for preparation of HPC simulation scenarios.
• Using outcomes of this study for developing SLA management capability for Load Sharing Facility (LSF) [37],
• Comparing the performances of the policy based manageable RMS with those of traditional RMSs, and those of heuristic based RMSs such as exploiting evolutionary algorithms and genetic programming. For this purpose, comparing the RMSs via experiments that use the same workload traces
• Developing the mechanisms for SLA negotiations between administrative domains into HeteroSIM and extending POLICE model for SLA management,
• Improving statistical metrics charts so that they do not affect the performance of HeteroSIM simulations.

REFERENCES


