A Framework for Distributed Simulation on a Grid

Ming Jiang
Georgios Theodoropoulos

School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
{mzj | gkt}@cs.bham.ac.uk

Rachid Anane

School of Mathematical &
Information Sciences
Coventry University
Coventry, CV1 5FB UK
r.anane@coventry.ac.uk

Abstract: The last decade has witnessed an explosion of interest in distributed simulation techniques, however the application of simulation to ever more complex problems calls for more computational power. The emergence of the GRID provides an unrivalled opportunity to address this problem and thus make advances in the modeling and analysis of large-scale systems by harnessing the power of many computers at once. This paper presents a framework and an architecture for performing distributed simulations on the GRID.

***Keywords*:** GRID computing, distributed simulation.

1 INTRODUCTION

Simulation has long been placed in the highly computation intensive world. Applications in which the computational requirements of simulations far exceed the capabilities of conventional sequential von Neumann computer systems include health care systems, training, military systems, environment systems, flexible manufacturing systems, aerodynamic simulation, telecommunication networks etc.

The execution of these computationally intensive simulations requires either modelling at higher levels of abstraction in order to reduce the computational load or using more powerful machines. The former is not considered a satisfactory approach as it does not allow the required detail to be incorporated in the model and may thus result in the over-simplification of the investigated problem. As a result, in the last fifteen years, the attention of modellers has focused on parallel machines, as the answer to the demand for increased computational power.

Multiprocessors, networks of workstations and cluster computers have well served the computational needs of the simulation community allowing the simulations of larger models. However, the application of simulation to more complex problems calls for more computational power. The emergence of the GRID provides an unrivalled opportunity to address this problem and thus make advances in the modeling and analysis of large-scale systems harnessing the power of many computers at once.

As a contribution to this endeavour, this paper presents a framework for conducting distributed simulations on the GRID.

2 DISTRIBUTED SIMULATION

The term distributed simulation refers to the execution of discrete event simulation models on parallel/distributed platforms [4]. Various approaches for exploiting parallelism at different levels have been followed. Decentralised, event-driven simulation has the greatest potential for high performance and consequently, has attracted considerable attention from the research community and has almost exclusively been employed for practical simulation applications. Following this approach, a simulation consists of a number of parallel logical processes (LPs), each operating independently and communicating with its peers to exchange information. Each LP maintains a local clock with the current value of the simulated time, Local Virtual Time (LVT). This value represents the process's local view of the global simulated time and denotes how far in simulated time the corresponding process has progressed. An LP will repeatedly accept and process messages arriving on its input links, advancing its LVT and possibly generating, as a result, a number of messages on its output links. The timestamp of an output message is the LVT of the LP when the message was sent.

A fundamental problem in event-driven distributed simulation is to ensure that the LPs always process messages in increasing timestamp order, and hence faithfully and accurately implement the causal dependencies and partial ordering of events dictated by the causality principle in the modelled system. Synchronous approaches utilise global synchronisation schemes (implemented in a centralised or decentralised fashion) to force the LPs to advance together in lock step. This guarantees that the causality principle is not violated but the potential for speedup is limited. In contrast, in asynchronous simulation, LPs operate asynchronously, advancing at completely different rates, simultaneously processing events which occur at completely different simulated times. This approach has greater potential for speedup, but additional synchronisation mechanisms are required to ensure that the LPs adhere to the *local causality constraint* and process messages in increasing timestamp order.

Two main approaches have been developed to ensure that the local causality constraint is not violated in asynchronous simulation, namely *conservative* and *optimistic*. Conservative approaches strictly avoid causality errors but can introduce deadlock problems. In addition, conservative techniques rely heavily on the concept of lookahead, and are thus typically suitable only for applications with good lookahead properties. Conservative protocols also typically require a static partition and configuration of the distributed model, and systems with dynamic behaviour are in general difficult to model. Optimistic approaches allow the processes to advance optimistically in simulated time, detecting and recovering from causality errors by means of a *rollback* mechanism which forces processes to undo past operations. For the rollback of the simulation to be feasible, each process must hold information regarding its recent history (e.g., previous state vectors, processed input events, and previously sent output messages) up to last 'correct time', referred to as the *Global Virtual Time* (GVT). GVT is generally the smallest local clock value amongst all the LPs, and is periodically computed and distributed to all the logical processes. In contrast to conservative approaches, optimistic approaches can accommodate the dynamic creation of logical processes and do not require the prediction of future events for their efficient operation.

Efficient management of resources, such as CPU and memory, can have a direct effect on the synchronization overhead and thus plays a crucial role in the performance of distributed simulations. For instance, in an optimistic simulation, the amount of memory available can define the degree of optimism of a Logical Process and the amount of rollbacked operations. The synchronization mechanisms involved in distributed simulation render load balancing techniques developed for other, more conventional classes of parallel applications insufficient. In the case of optimistic synchronization, high processor utilization does not necessarily imply good performance as operations could later be undone (rollback), while process migration can affect the efficiency of the synchronization mechanism (e.g. amount of rollbacked computation) [4][8].

3 GRID COMPUTING

The level and intensity of the research devoted to GRIDs and GRID computing is a testimony to their increasing importance [3]. 'GRIDs are persistent environments that enable software applications to integrate instruments, displays, computational and information resources that are managed by diverse organizations in widespread locations' [5]. GRIDs are large, dynamic, distributed and heterogeneous resource systems [7], and GRID computing aims at providing transparent access to scarce resources. Different types of GRID have been identified; among them computational GRIDs are the subject of much research and attention. They are concerned with the efficient execution of tasks on a set of compute servers. In common with other types of GRID, the use and integration of resources in GRID computing involves a set of fundamental middleware services: resource advertisement and discovery, scheduling and transfer of tasks [1].
In advertising its resources a compute resource provider might be required to include the number of nodes in a cluster, the amount of memory, the operating system version and the load average. Resource

discovery by a resource requestor is often achieved by means of a directory service or broker/matchmaker. The main function of this service is to locate remote resources and to help route computational requests to the most suitable computer in a GRID [4]. Suitability can be expressed in terms of static information such as architecture and performance, or in terms of dynamic information such as availability and instantaneous load. This service is usually provided by a global scheduler, and the transfer of tasks requires commitment to service provision. Service provision may involve negotiation as an integral part of scheduling.

The ubiquitous nature of the GRIDs is undoubtedly enhanced by the development and introduction of middleware services. The Globus Toolkit is one significant manifestation of the drive towards providing seamless access to remote and scarce resources. It is an open, middleware architecture, which offers a rich set of services and libraries that facilitate connectivity and mediation to GRID resources [3]. The GRID services provided by Globus include information bases, resource management, data management, communication, security and fault detection.

4 A SIMULATION FRAMEWORK

In this section, the simulation process is considered and its components identified, in the context of the GRID. The role of the proposed framework is to support the simulation process. Three fundamental tasks can be distinguished in distributed simulation:

1. **Simulation requirements**. This initial step is application-dependent and is mainly concerned with the definition and the structure of the logical processes (LPs) that make up the simulation and their logical organization. The resources requirements of the simulation, namely CPU or memory, need to be identified at this stage.

2. **LP Scheduling.** A subsidiary function of the scheduling process, called brokering, is the search and discovery of resources on the GRID, and the matching of these resources to the simulation requirements. The scheduler is mainly responsible for the allocation of LPs to GRID resources. Scheduling decisions may occur at task initiation time or at run-time (e.g. dynamic load balancing).

3. **LP Execution**. This step involves running the simulation and coordinating and synchronizing the activities of the various LPs. The interaction between LPs identifies a simulation activity. If dynamic load balancing is employed, a concurrent but related task concerns the monitoring of the simulation performance; the generation of this information is required by the scheduler.

Although these tasks may be considered as consecutive, there is, however, an overlap between tasks 2 and 3 and also some iteration over them. The fundamental tasks and the sub-tasks they spawn are sustained by a number of functions that contribute to the management of the simulation process. These functions can be performed locally at node level or globally at GRID level. They may also be distinguished by their level of awareness of the underlying GRID. They are concerned with the management of the simulation itself and with the underlying resources:

1. **Management of the simulation processes.** This function is performed locally at node level and revolves mainly around the execution of an OS process. It is therefore a GRID-unaware function, which may represent the extreme case of a simulation activity made up of a single LP. The LP is made up of two parts, the simulation model and the simulation kernel, which is communication-enabled.

2. **Management of the simulation activity.** This is a logical and application-oriented function performed at global level. One important operation that pertains to this function is synchronisation, as an instance of the exchange of information between LPs. Since it is concerned with the execution and the semantics of the simulation, the GRID is transparent. This function requires a communication channel between LPs. The communication itself is GRID-aware.

3. **Management of resources.** This is carried out by two main sub-functions, brokering and scheduling, which are performed at global level and are GRID-aware. While the broker is concerned with the search and discovery of specific resources, expressed in terms of CPU, memory or OS requirements, the scheduler is responsible for the allocation and transfer of LPs to remote nodes. An additional task that can be performed on behalf of this function is dynamic load balancing. Attempts at addressing the inherent inefficiencies of load imbalance in distributed simulations can take two forms: migration of the state of the simulation application (logical process), or migration of both the logical process and its associated state. These two types of migration occur at user level. The process migration will have to be non-preemptive, a requirement imposed by the heterogeneity of the GRID [2][6].

4. **Management of the GRID services**. This function works at global level and is GRID-aware. It requires access to information on resources on the GRID, in order to support the brokering activity, and connectivity in order to facilitate the communication activity between LPs. This function is supported by an interface between the simulation process and the GRID middleware.

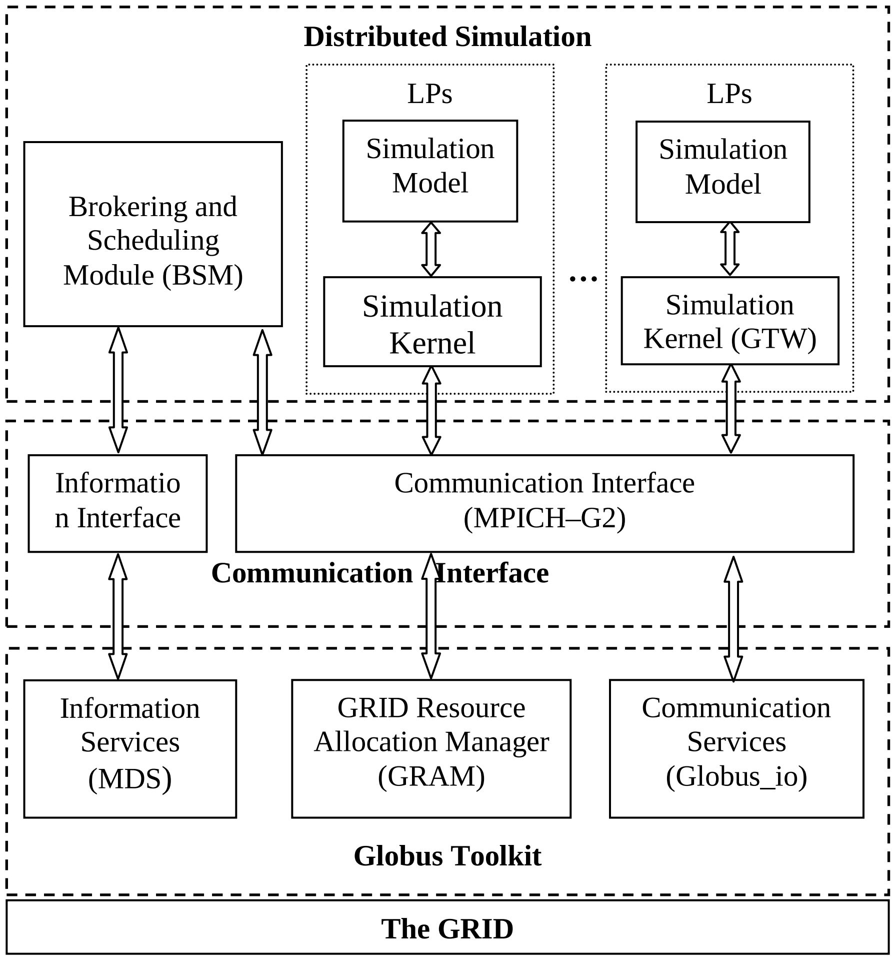
Distributed Simulation
LPs
Simulation Model
Simulation Kernel
LPs
Simulation Model
Simulation Kernel (GTW)
Brokering and Scheduling Module (BSM)
...
Informatio n Interface
Communication Interface (MPICH–G2)
Communication Interface
Information Services (MDS)
GRID Resource Allocation Manager (GRAM)
Communication Services (Globus_io)
Globus Toolkit
The GRID

Figure 1: The Proposed Architectural Framework

5 AN EXAMPLE ARCHITECTURE

The proposed framework is designed to enable the execution of simulations transparently on the GRID. Figure 1 shows an architecture that exemplifies and implements the framework.

The architecture presents a decreasing level of abstraction from the application (logical process) to the GRID itself (physical resource), as mediated by Globus. This hierarchy defines four layers:

1. Simulation
2. Communication
3. GRID middleware
4. GRID

While layers 3 and 4 are predefined, the two top layers (1 and 2) represent part of the architectural framework on which the simulation process and the four management functions in particular, identified above in section 4, can be mapped. In our implementation, Layer 1 utilises the GTW optimistic distributed simulation kernel [9] while Layer 3 is implemented by the Globus Toolkit.

Layer 1: Simulation. Two separate components are defined at this layer. The brokering and scheduling module (BSM) is concerned with the management of resources as described above (function 3). This framework implements a centralized approach to resource management and load balancing. The decisions of brokering, scheduling and balancing strategies of each above functions are based on the analysis of the GRID resource availability information and simulation application performance information. The other component is the set of LPs, distributed at different nodes on the GRID. Their execution on a node by the OS is part of the simulation activity. The simulation kernel is responsible for the management of the simulation activity and of synchronization in particular.

Layer 2: Communication. This communication layer is a bridge between the simulation activity and GRID middleware and therefore the GRID. The Information Interface module allows to BSM to perform the matchmaking task, by extracting information on the GRID through the Information Services (MDS) of Globus. The communication interface, on the other hand, allows the simulation kernel to use the GRID as a communication channel between LPs. This enhancement requires the selection of a communication mechanism that is Globus/GRID-aware; for this implementation, MPICH-G2 is utilized, a GRID-enabled MPI implementation library, which is based on Globus services. The original GTW kernel has been modified to replace PVM with MPICH-G2 and make GTW MPICH-G2 aware. The communication layer enables, on one hand, the modified GTW to make use of the communication services of Globus (Globus_io), and on the other hand supports the scheduling function of the BSM by providing access to the GRAM. The GRAM is used for the allocation of computational resources and for monitoring and control of computation on those resources.

6 RESULTS

We are currently evaluating the performance of the architecture described above by executing typical benchmark distributed simulations on a Globus-enabled network of workstations running Linux. Figure 2 shows the performance and the speedups achieved for the PHOLD synthetic benchmark [4]. PHOLD is one of the most commonly used benchmarks for parallel simulation. It has a tightly coupled topology, with many feedback paths. It consists of a network of N x N nodes (LPs) with each node being interconnected to four neighboring nodes. Every node is initialized with *m* jobs (messages) at the start of simulation. The distribution of the computation time per event, the timestamp increment and the LP to which the message is to be sent are the parameters of the simulation. For the results depicted in figure 2, we have used the standard configuration of PHOLD distributed with GTW [9] with N=32 (1024 nodes) and m=256.

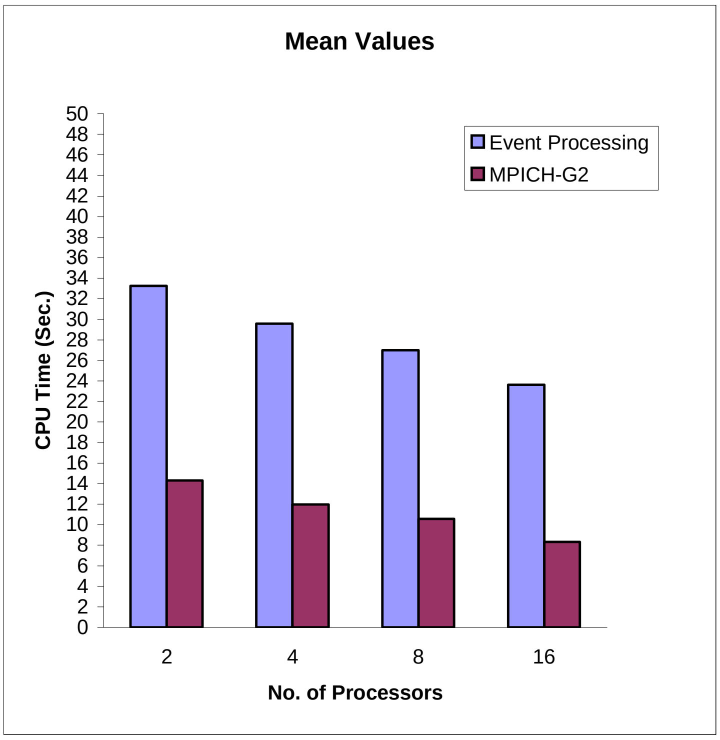

Figure 2: Performance of the Globus-aware GTW Kernel (PHOLD benchmark)

The figure shows the times spent in the Simulation (event processing, Layer 1) and the Communication and GRID middleware (MPICH-G2 Layers 2 and 3). Despite the relatively small number of processes simulated, the speedup achieved is apparent, suggesting that the overhead introduced by the Grid-middleware is acceptable. We are currently performing more experiment of to obtain a more complete performance analysis of the proposed architecture. The results of these experiments will be the subject of a subsequent publication.

REFERENCES

[1] Abramson D., Buyya R. and Giddy J., A Computational Economy for GRID Computing and its Implementation in the Nimrod-G Resource Broker, *Future Generation Computer Systems, 18 (2002),* pp1061-1074.

[2] R. Anane and R.J. Anthony Implementation of a Proactive Load Sharing Scheme. Proceedings of the 18th ACM Symposium on Applied Computing (SAC 2003), Melbourne, USA, March 2003, pp1038-1045.

[3] Forster I. And Kesselman C. (Eds.), *The GRID: Blueprint for a new Computing Infrastructure*, Morgan Kaufman, 1999.

[4] R.M. Fujimoto, "*Performance of Time Warp under Synthetic Workload*", Proceedings of the SCS Multiconference on Distributed Simulation, 22, 1, 1990.

[5] R. M. Fujimoto. Parallel and Distributed Simulation Systems. John Wiley & Sons, Inc., New York, 2000.

[5] The Globus Project, http://www.globus.org/

[6] Milojici, D.S., Douglis, F., Paindeveine, Y., Wheeler, R. and Zhou, S., Process Migration, *ACM Computing Surveys, 32(3*), September 2000, pp241-299.

[7] Skillicorn D.B., Motivating Computational GRIDs, *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the GRID*, Berlin, May 2002, pp401-406.

[8] Theodoropoulos, G., Logan, B., " Interest Management and Dynamic Load Balancing in Distributed Simulation" Proceedings of 12th European Simulation Symposium (ESS'2000), 28-30 September 2000, University of Hamburg, Hamburg, Germany, pp. 111-115, Society for Computer Simulation International, ISBN 1-56555-190-7, editors: Dietmar Moeller

[9] The GTW Homepage, http://www.cc.gatech.edu/computing/pads/teddoc.html