An architecture for Grid-enabled distributed simulation

Ming Jiang and Georgios Theodoropoulos*

School of Computer Science,
University of Birmingham, B15 2TT, UK
E-mail: mzj@cs.bham.ac.uk
E-mail: gkt@cs.bham.ac.uk
*Corresponding author

Rachid Anane

Department of Computer and Network Systems,
Coventry University, CV1 5FB, UK
E-mail: r.anane@coventry.ac.uk

Abstract: The application of distributed simulation techniques to ever more complex problems calls for more computational power. The emergence of the Grid provides an opportunity to address this challenge and thus make advances in the modelling and analysis of large-scale systems by harnessing the power of many computers. These requirements are met by the design and development of frameworks that support the integration of appropriate Distributed Simulation Kernels (DSK) and Grid middleware. This paper presents a framework and a prototype implementation for specifying and executing distributed simulations over the Grid. The framework incorporates a well established DSK with the Globus middleware.

Keywords: Grid computing; distributed simulation.

Reference to this paper should be made as follows: Jiang, M., Theodoropoulos, G. and Anane, R. (2009) 'An architecture for Grid-enabled distributed simulation', *Int. J. High Performance Computing and Networking*, Vol. 6, No. 1, pp.47–55.

Biographical notes: Ming Jiang is a PhD student at the School of Computer Science, University of Birmingham. He holds an MSc in Computer Science from the same school.

Georgios Theodoropoulos is a Reader in Computer Science at the University of Birmingham. He holds a Diploma in Computer Engineering from the University of Patras, Greece and MSc and PhD in Computer Science from the University of Manchester, UK. His current research interests include distributed systems, simulation and Grid computing.

Rachid Anane is a Senior Lecturer in Computer Science at Coventry University. He holds a BSc in Computer Science from the University of Manchester, UK and an MSc and a PhD in Computer Science from the University of Birmingham, UK. His research interests include software engineering and distributed systems.

1 Introduction

Simulation has often placed a heavy premium on highly intensive computation. Applications where 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 and telecommunication networks. 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 problem at hand. As a result, in the last two decades the attention of modellers has focused on parallel and distributed platforms, as the answer to the demand for increased computational power. The term distributed simulation refers traditionally to the execution of discrete event Simulation Models (SM) on parallel/distributed platforms (Fujimoto, 2000). Distributed simulation has recently witnessed an explosion of interest and innovation, not only for speeding up simulations but also as a strategic technology for linking simulation components of various types geographically distributed at multiple locations to

create a common virtual environment (e.g., battlefields, virtual factories and supply chains, agent-based systems, games etc.). This activity, which is generally referred to as Large Scale Distributed Simulation (LSDS), is mainly centred on the High Level Architecture (HLA), a framework originated from an IEEE standard (an offshoot from the USA DoD Defence Modelling and Simulation Office) to facilitate interoperability among simulations and promote reuse of SMs.[1]

Multiprocessors, networks of workstations and cluster computers have served well the computational needs of the simulation community by 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 issue and thus make further advances in the modelling and analysis of large-scale systems by harnessing the power of many distributed computational resources at once. The potential of the Grid as a computational platform is being investigated for both PDES and LSDS.

Work on LSDS is concerned mainly with the integration of HLA and Grid middleware in addressing challenges such service oriented architectures for simulation, model discovery and matching, resource management etc. (Theodoropoulos et al., 2006). A seminal work in this area is XMSF[2], while recently dedicated workshops have also been established.[3] Work on PDES includes Mikler et al. (1998) and Iskra et al. (2003, 2005) and has been concerned with techniques for dealing with increased communication latencies on the Grid.

As part of this endeavour, this paper presents a generic framework for deploying PDES on the Grid. The remainder of the paper is organised as follows. Section 2 gives an introduction to GRID computing and Section 3 covers distributed simulation. Section 4 introduces a generic framework for distributed simulation on a GRID and Section 5 describes a prototype implementation. Section 6 deals with the evaluation of the prototype and in Section 7 some conclusions are drawn.

2 GRID computing

The level and intensity of the research devoted to GRIDs and GRID computing is a testimony to their increasing importance (Foster and Kesselman, 1999). GRIDs are large, dynamic, distributed and heterogeneous resource systems, 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.

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. 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[4] 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. The GRID services provided by Globus include information bases, resource management, data management, communication, security and fault detection.

3 Distributed simulation

Various approaches to exploiting parallelism at different levels have been proposed (Fujimoto, 2000). 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. In 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 constraint 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 (Fujimoto, 2000). 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 LPs. In contrast to conservative approaches, optimistic approaches can accommodate the dynamic creation of LPs and do not require the prediction of future events for their efficient operation. Popular optimistic DSKs include WARPED[5] and GTW.[6]

4 A generic framework for distributed simulation on a Grid

This section introduces an architectural framework for developing, managing, and executing PDES on the Grid. The framework is generic and agnostic with respect to the synchronisation protocols employed. The framework is one manifestation of the simulation process whose components in the context of the Grid are identified and detailed in the following section.

4.1 The simulation process

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 LPs that make up the simulation and their logical organisation. The resources requirements of the simulation, namely CPU or memory, need to be identified at this stage.

2 *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 *Execution*. This step involves running the simulation, coordinating and synchronising the activities of the various LPs. The interaction between LPs identifies a simulation activity. When 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 sequential, there is, however, an overlap between tasks 2 and 3 in addition to 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. Their level of awareness of the underlying GRID may be also one of their distinguishing features. They are concerned with the management of the simulation itself and with the underlying resources:

- *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 SM and the simulation kernel, which is communication-enabled.
- *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.
- *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 (Anane and Antony, 2003; Skillicorn, 2002).

- *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 communication between LPs. This function is supported by an interface between the simulation process and the GRID middleware.

4.2 Architectural framework

At the heart of this framework is a Grid-Enabled Distributed Simulation Platform (GEDSP), which consists of functional modules and strategies for mapping a distributed simulation to Grid resources, and for optimising the simulation performance over heterogeneous and dynamic Grid resources at run time. The proposed framework is designed to enable a distributed simulation to run transparently and efficiently on a computational Grid. In this framework, the general concepts of Grid resource fabric, Grid middleware, application level middleware and application are identified in the context of supporting a distributed simulation.

The design of the framework is open and enables the integration and evaluation of various functional modules for supporting a distributed simulation on a Grid. From the point of view of an application, the framework can be viewed as a blueprint for constructing a distributed simulation Grid, which exhibits the three fundamental properties of a Grid, namely transparency, scalability and conformance to open standards, and in particular standard Grid middleware technologies.

The architecture of the framework is illustrated in Figure 1. It presents a decreasing level of abstraction from the simulation application to a computational Grid resource itself. This hierarchy defines four layers:

1 application layer

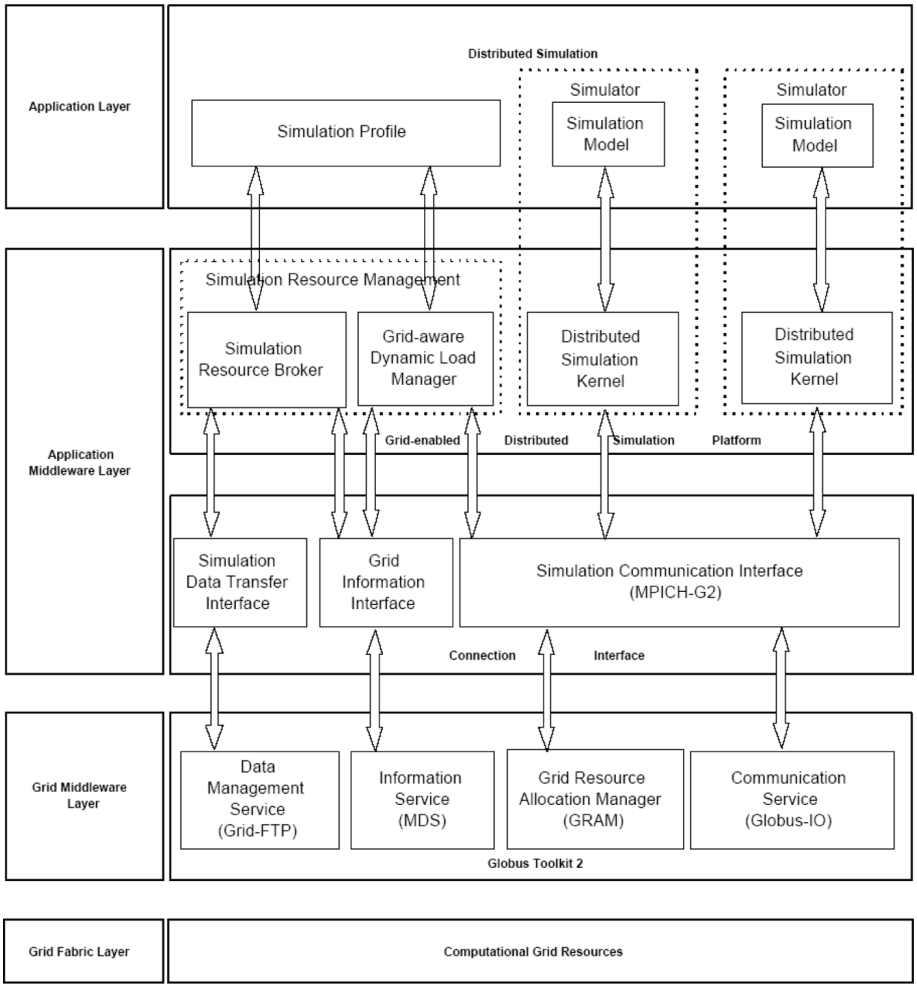
2 application middleware layer

3 grid middleware layer

4 grid fabric layer.

While parts of layer 3 and layer 4 are predefined, the two top layers (1 and 2) represent part of the architectural framework to which the simulation process and the four management functions in particular, can be mapped.

Figure 1 The architecture of the generic framework



4.2.1 Application layer

The application layer includes a SM and a Simulation Profile (SP). An SM specifies the system to be simulated. An SP is used to describe the characteristics of a simulation and the resources required for the simulation of the model. These resource requirements provide crucial information for a simulation resource brokering function (at Layer 2) and help identify the appropriate resources in a Grid before the execution of a simulation program. The characteristics of a simulation, on the other hand, form the basic information for a dynamic load management mechanism (at Layer 2). The provision of dynamic adjustment of the execution of a simulation takes into account both the nature of the simulation and Grid dynamics.

4.2.2 Application middleware layer

The application middleware layer is central to the architecture, and plays a crucial role in the framework. This layer includes two sub-layers: a GEDSP sublayer and a Connection Interface (CI) sublayer between the platform and the underlying Grid. It is the combination of these two sub-layers, which enables distributed simulations to be Grid-aware.

The GEDSP sub-layer insulates a simulation developer or end user from the complexity of the underlying Grid environment. The GEDSP encompasses a DSK and a Simulation Resource Management Module (SRMM). The DSK is responsible for the management of the simulation activity and the synchronisation of simulators distributed on a Grid. The simulation kernel and the model implemented and simulated on it can be viewed together as a simulator (e.g., an LP in PDES). The main functions performed by the SRMM include Grid resource discovery, the mapping of simulators to Grid resources, the allocation and execution of a simulation over the corresponding Grid resources and the run-time load balancing of simulation. The SRMM consists of a Simulation Resource Broker (SRB) and a Grid-aware Dynamic Load Manager (GDLM). The SRB performs two tasks: organisation of resources for a simulation execution and deployment of the simulation onto these resources. The broker selects the most appropriate Grid resources on behalf of the simulation users and transfers the simulation programs to the targeted resources. The GDLM, if enabled, adjusts automatically the simulation to the changes in Grid resources.

The CI sub-layer acts as a bridge between the simulation platform and Grid middleware and, therefore, the Grid resources. It consists of a Simulation Data Transfer Interface (SDTI), a Grid Information Interface (GII), and a Simulation Communication Interface (SCI). The SDTI utilises the Grid-FTP service provided by the Globus Toolkit to transfer simulators to Grid resources. The GII allows SRB to perform the matchmaking between simulators and Grid resources, by extracting information on the Grid through the Monitoring and Discovery System (MDS) of Globus. The SCI enables a DSK to use the Grid as a communication channel between simulators. This enhancement requires the selection of a communication mechanism that is Globus/Grid-aware; in the context of the proposed framework, MPICH-G2 as used as a communication mechanism.

4.2.3 Grid middleware layer

The Grid Middleware Layer mediates between high level applications and low level Grid resources. Although in the proposed framework the Grid middleware layer was implemented in terms of Globus Toolkit 2, other systems, such as OGSA and WSRF standards compliant GT3 or GT4, can be considered as viable alternatives. In the context of the Globus framework, this layer incorporates modules such as Grid-FTP, MDS, Grid Resource Allocation Manager (GRAM) and Globus-IO.

4.2.4 Grid fabric layer

The selection of MPICH-G2 as the SCI in this framework implies that the Grid resources that can be accessed and harnessed must be compatible with MPICH-G2 requirements. These resources range from a single workstation to high performance clusters connected by the internet. In the latter case, the intra-cluster commutation latency is far less than inter-cluster communication and these clusters are likely to be owned by different organisations while their computational capacity is heterogeneous and dynamic.

4.2.5 Simulation deployment

The execution of a simulation on the Grid-aware Time Warp PDES simulation platform involves a number of steps, through the different layers of the framework:

- A user implements a simulation application by following the LP model defined in GTW kernel, generates the binary simulation executable and provides a SP to the SRB.
- Based on the resource requirements specified in the SP the broker identifies the relevant Grid resources by querying the Grid Information Service (MDS) via the Information Interface.
- After identifying the required Grid resources, the broker transfers the binary simulation executable and its associated Resource Configuration File (RCF) and the Resource Specification File (RSF) to the identified remote Grid resource by calling the Grid Data Management Service via the Data Transfer Interface.

- In accordance with the context of simulation in RSF, MPICH-G2 stages the remote simulation of executables automatically and coordinates the execution of the simulation programs.
- When the simulation is complete, MPICH-G2 redirects the simulation results and stores them in user specified output files.

5 A prototype implementation

In order to evaluate the viability of the proposed framework, a prototype was implemented by incorporating specific instances of module components at the Application Middleware Layer. The generic modules that were instantiated were the DSK and the SRB on the one hand, and the SCI on the other. The prototype is based on Globus Toolkit 2. As a SCI mechanism, MPICH-G2[7], a Grid-enabled MPI implementation was selected. MPICH is a high performance and highly portable implementation of MPI standard by Argonne National Laboratory[8] which implements the Abstract Device Interface (ADI) for transferring data between the MPI interface and network subsystem. MPICH-G2 offers another implementation of the ADI, also referred to as the globus2 device provided by the Globus Toolkit.

5.1 A Grid-aware distributed simulation kernel

GTW was used as the DSK in the implementation of the prototype. GTW is a popular optimistic PDES kernel designed to execute on a Network of Workstations (NOW), Symmetric Multiprocessors (SMP), and cluster computers. A GTW simulation consists of a collection of LPs that communicate by exchanging time-stamped messages. The GTW simulation engine can be viewed as consisting of four major elements, simulation starting mechanisms, an event scheduler, simulation termination mechanisms and a SCI to the underlying network communication platform, which connects the GTW instances and coordinates the progress of the distributed simulation.

The communication interface of GTW is based the Parallel Virtual Machine (PVM)[9] platform. PVM only supports inter-process communication within LAN environments.

The original GTW simulation kernel is based on PVM and therefore is limited in LANs (or cluster computers). The integration of GTW into the proposed framework requires as a first step the enhancement of its communication interface by replacing PVM with a Grid-aware communication platform, namely MPICH-G2.

Each instance of GTW kernel is an MPI process. In the context of MPICH-G2, there is no spawn library as in PVM. All instances of the GTW kernel are invoked by the *mpirun* command simultaneously. The program invocation and communication functions of MPICH-G2 are implemented by the Globus Toolkit services. The MPICH-G2 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 SRB and the GDLM by providing access to the GRAM services of the Globus Toolkit. The main modifications in the GTW kernel include replacing the PVM libraries with MPI libraries and the PVM process spawn mechanism with the MPI parallel process invocation mechanism. These modifications involve redesigning and recoding the communication interface component of GTW. Figure 2 illustrates the PVM-based implementation of GTW, and Figure 3 presents the modified version with MPICH-G2 as communication interface component.

Figure 2 The GTW kernel with PVM

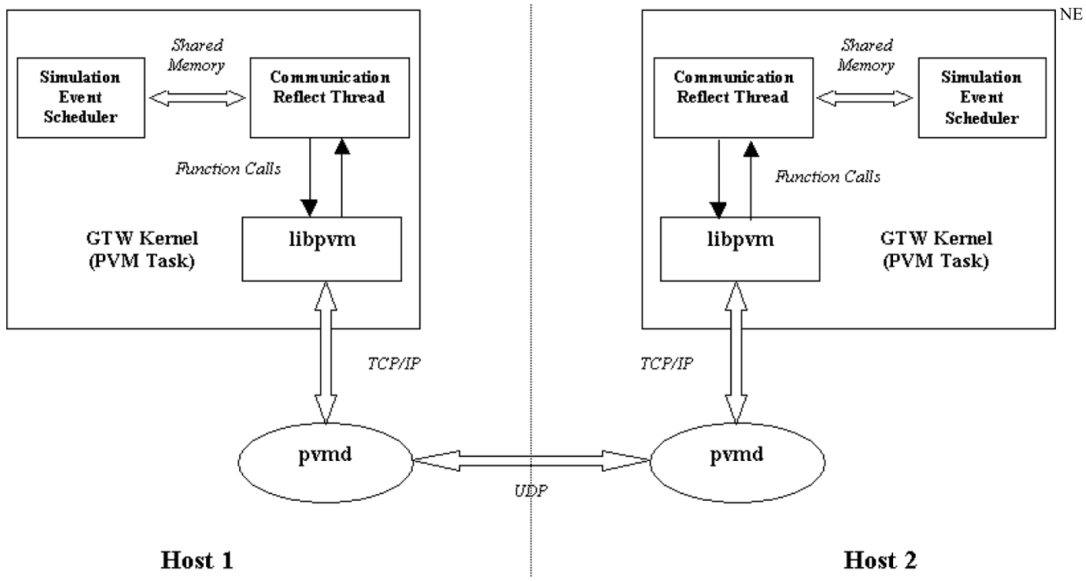
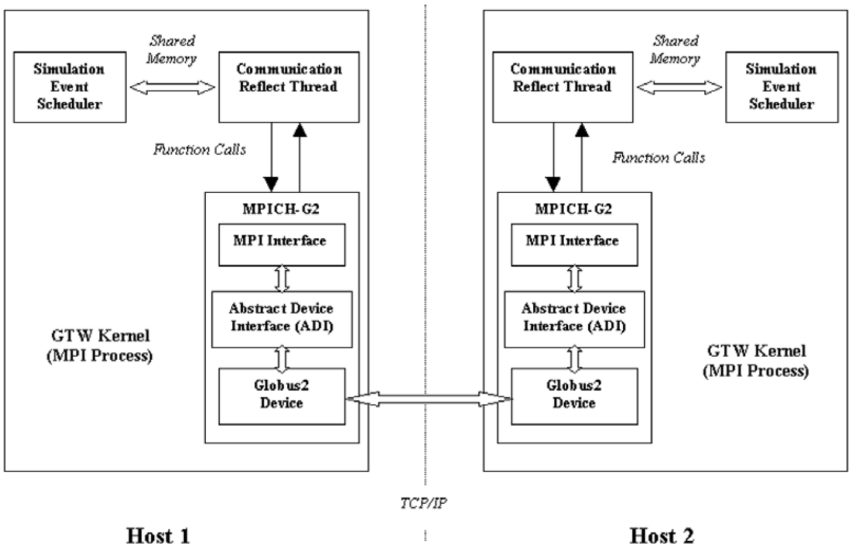


Figure 3 The GTW kernel with MPICH-G2



5.2 Simulation Resource Broker (SRB)

In the context of the proposed framework, the main responsibility of the SRB module is matching the simulation requirements with the available resources and transferring simulation programs to the relevant Grid resources for execution.

In the prototype implementation, the SRB consists of a Simulation Initialisation Component (SIC) and a Simulation Launcher (SL). The SIC first reads simulation resource requirements from the SP (specifying initial simulation resource requirements) provided by the simulation users, enquires Grid resource availability information via the GII and generates a Targeted Grid Resource Specification File (TGRSF) and a Simulation Configuration File (SCF) for the simulation. The MDS provided by the Globus Toolkit is used to query resource availability required by the SRB. Once a set of Grid resources are identified as the appropriate resources, the SL reads the TGRSF and generates a RSL for MPICH-G2, in which simulation execution and Grid resources configurations are specified. The SL transfers the executable code of the GTW instances and their associated RSL and SCF to the targeted Grid resources via the SDTI (at the CI sub-layer). The Grid Data Management Service (Grid-FTP) provided in Globus Toolkit is used to transfer these simulation programs.

6 Evaluation

This section deals with the viability of prototype system, presented in terms of experimental results and future enhancement. To this effect, a number of experiments have been conducted in order to obtain an indication of the overheads incurred by the introduction of MPICH-G2 and Globus in the prototype implementation described in the previous section.

At the Grid fabric layer, a Grid-enabled cluster was used consisting of 1 master and 22 client nodes. The master node has dual Intel Xeon 2 GHz CPUs and 2 G memory while each client node has dual Athlon MP 1900+ 1.6 GHz CPUs with 1 G of on-chip memory. The software utilised for the experiments included: PVM 3.4, Globus Toolkit Version 2 and MPICH-G2 1.2.5.

At the application Layer, the SM which was used for the experiments was PHOLD (Fujimoto, 1990), one of the most commonly used synthetic load benchmarks for parallel simulation. A PHOLD model contains a fixed number of LPs and an initially scheduled and distributed simulation event message population, which is kept as constant throughout the whole execution of a simulation. PHOLD has a tightly coupled topology, with many feedback paths. It consists of a network of $N \times N$ nodes (LPs) with each node being interconnected to four neighbouring nodes. An LP executes an event in event queue and sends an event to another selected LP with some specified timestamp increment. 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 experiments we have used the standard configuration of PHOLD distributed with GTW with $N = 32$ (1024 nodes). The initial simulation event messages are equally distributed among these LPs with four events for each LP. The timestamp increment of an event is randomly generated with an exponential probability distribution with a mean value of 0.1. The granularity of simulation event is set to null. Simulation termination is at 10,000 in GVT.

Figure 4 shows the elapsed times of the simulation for the two different implementations of GTW. Clearly, the elapsed times for MPICH-G2 are higher than those achieved with PVM. As the number of processor nodes increases, so does the relative difference in elapsed times between the two implementations. This is due to the additional layers involved in the communication between the nodes, when MPICH-G2 is used (see Figures 2 and 3). This overhead includes also the cost of Globus-IO operations. As the simulation is distributed across more nodes, communication overheads increase and become the dominant factor in the simulation performance.

Figure 4 Performance of the GTW kernel: PVM vs. MPICH-G2 (see online version for colours)

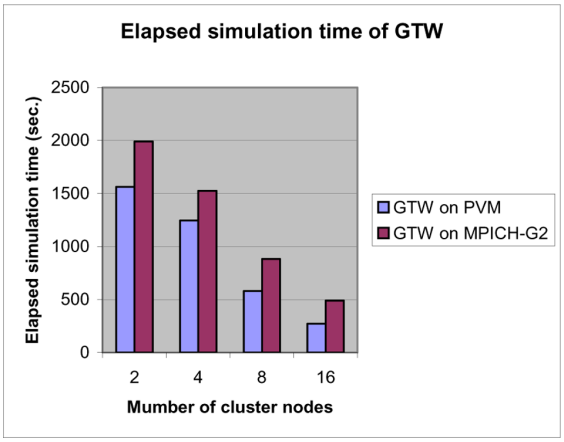


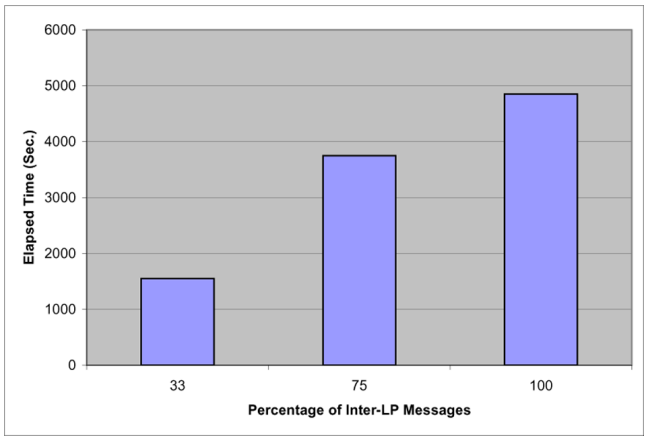
Figure 5 shows the impact of communication overheads on the performance of the simulator on a Grid. For this particular experiment, a Grid was emulated by changing the communication latencies between the nodes of the cluster. The nodes were divided into four fully interconnected groups and a different latency value was associated with each of the six links. The LPs are equally divided into four communication groups. When an LP generates an event message, the destination of the message can be itself (intra-LP message) or another LP (inter-LP message) within its communication group. The model is configured with 3% of inter-LP messages:

- *33% configuration*. The amount of inter-LP communication is not significant and the simulation time is mainly dominated by the events processing costs within a single LP. The processors are busy with computation rather than communication.
- *75% configuration*. The level of inter-LP communications is high and requires a significant amount of processor time.
- *100% configuration*. This represents the extreme case where all simulation events are inter-LP messages. Simulation time is dominated by the transmission of events over distributed processors.

These results suggest that for communication-bound simulations, the overheads introduced by the Grid middleware may be prohibitive and may negate the benefits afforded by harnessing increased computational power.

This issue points to the need for the introduction of appropriate load balancing mechanisms, such as the GDLM component in the proposed framework, to improve locality of reference, and therefore reduce communication delays. Conventional load balancing techniques are unsuitable for this class of parallel applications, because of the specific characteristics of the synchronisation mechanisms involved in distributed simulation. For instance, in the case of optimistic synchronisation, high processor utilisation does not necessarily imply good performance as operations could later be undone (rollback), while process migration can affect the efficiency of the synchronisation mechanism (e.g., amount of roll backed computation) (Fujimoto, 2000; Theodoropoulos and Logan, 2000). As a result, load balancing has been studied extensively in the context of both conservative and optimistic parallel simulation and several load balancing algorithms have been developed, (Burdorf and Marti, 1993; Carothers and Fujimoto, 1996). Most, if not all of these algorithms assume parallel/distributed platforms with negligible inter-processor communication delays (such as multiprocessors, networks of workstations and cluster computers). Thus their feedback loop is based mainly on processor load and they tend largely to ignore communication latencies.

Figure 5 Performance of the GTW kernel on a Grid (see online version for colours)



In a Grid, however, communication delays are substantial and can be the decisive factor of the simulation performance, as shown by the results Figure 4. This issue has been a focal point of research in this area. In (Mikler et al., 1998) a simulation event flow control mechanisms was proposed for adaptively adjusting resource demands and optimism as a function of the underlying network performance. A multi-level LP structure is employed which clusters LPs on individual Grid nodes. Optimistic synchronisation within a cluster inter-cluster synchronisation is achieved through a conservative protocol. In (Iskra et al., 2003, 2005) the authors investigate the use of lazy cancellation and message aggregation techniques for improving the performance of an optimistic PDES over a high latency Grid.

These considerations have identified load balancing as the main focus of future work within the framework proposed in this paper. In particular, investigations are being carried out on determining the most appropriate way of using communication latencies as a guiding factor in the dynamic migration of LPs (or state) between remote nodes (e.g., clusters) in a Grid. In the generic framework , this task is performed by the GDLM. Experiments are being conducted with an algorithm which, first decides dynamically on a usable set of processors, namely processors that are not overloaded and therefore can be used for the simulation (the processor allocation policy) and then employs a load balancing policy to migrate LPs from overloaded processors to less loaded ones.

7 Conclusion

In this paper a framework for the specification and deployment of distributed simulation was introduced and described. The main advantage of the framework is that it is able to harness Grid resources thanks to the adaptation and integration of a DSK, GTW and the Globus middleware. The evaluation of the framework has however underlined the high level of the communications costs of the CI, when compared to traditional PVM implementations. It has also brought to the fore the fact that its relevance and usefulness need to be enhanced by the introduction of adaptive policies such as load balancing. The provision of this adaptive feature is currently the main focus of this research.

References

Anane, R. and Anthony, R.J. (2003) 'Implementation of a proactive load sharing scheme', *Proceedings of the 18th ACM Symposium on Applied Computing (SAC 2003)*, March, Melbourne, USA, pp.1038–1045.

Burdorf, C. and Marti, J. (1993) 'Load balancing strategies for time warp on multi-user workstations', *The Computer Journal*, Vol. 36, No. 2, pp.168–176.

Carothers, C. and Fujimoto, R. (1996) 'Background execution of time-warp programs', *Proceedings of 10th Workshop on Parallel and Distributed Simulation*, Society for Computer Simulation, (PADS '96), May, Philadelphia, PA, USA, pp.12–19.

Foster, I. and Kesselman, C. (1999) *The GRID: Blueprint for a New Computing Infrastructure*, Morgan Kaufman, USA.

Fujimoto, R.M. (1990) 'Performance of time warp under synthetic workloads', *Proc. SCS Multiconf. Distributed Simulation*, San Diego, January, California, USA, Vol. 22, No. 1, pp.23–28.

Fujimoto, R.M. (2000) *Parallel and Distributed Simulation Systems*, John Wiley and Sons, Inc., New York.

Iskra, K.A., van Albada, G.D. and Sloot, P.M.A. (2003) 'Time warp cancellation optimizations on high latency networks', *Proceedings Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pp.128–135.

Iskra, K.A., van Albada, G.D. and Sloot, P.M.A. (2005) 'Towards grid-aware time warp', *Simulation: Transactions of the Society for Modeling and Simulation International Journal*, Vol. 81, No. 4, pp.293–306.

Mikler, A.R., Das, S.K. and Fabbri, A. (1998) 'Distributed simulation for large communication infrastructures across loosely coupled domains', *Proceedings of the 6th International Conference on Telecommunication Systems: Modeling and Analysis*, pp.561–569.

Milojicic, D.S., Douglis, F., Paindaveine, Y., Wheeler, R. and Zhou, S. (2000) 'Process migration', *ACM Computer Surveys*, Vol. 32, No. 3, pp.241–299.

Skillicorn, D.B. (2002) 'Motivating computational GRIDs', *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the GRID*, Berlin, May, pp.401–406.

Theodoropoulos, G. and Logan, B. (2000) 'Interest management and dynamic load balancing in distributed simulation', in Moeller, D. (Ed.): *Proceedings of 12th European Simulation Symposium (ESS'2000)*, Society for Computer Simulation International, ISBN 1-56555-190-7, University of Hamburg, 28–30 September, Hamburg, Germany, pp.111–115.

Theodoropoulos, G., Zhang, Y., Chen, D., Minson, R., Turner, S., Cai, W., Yong, X. and Logan, B. (2006) 'Large scale distributed simulation on the Grid', *International Workshop on Large Scale Distributed Simulations on the Grid, (DS-Grid 2006), 6th IEEE International Symposium on Cluster Computing and the Grid*, 16–19 May, Singapore.

Notes

[1]High Level Architecture, www.dmso.mil/public /transition/hla/

[2]XMSF Home Page, www.movesinstitute.org/xmsf/

[3]For instance, the International Workshop on Large Scale Distributed Simulations on the Grid, (DS-Grid 2006), in conjunction with the 6th IEEE International Symposium on Cluster Computing and the Grid, 16–19 May 2006, Singapore.

[4]Globus Project, www.globus.org/

[5]WARPED Home Page, www.ececs.uc.edu/~paw/warped/

[6]GTW Home Page, www.cc.gatech.edu/computing/pads/teddoc.html

[7]MPICH-G2 Home Page, www3.niu.edu/mpi

[8]MPICH Home Page, http://www-unix.mcs.anl.gov/mpi/mpich/index.htm

[9]PVM Home Page, www.csm.ornl.gov/pvm/pvm_home.html