Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

SUPPORTING SIMULATION IN INDUSTRY THROUGH THE APPLICATION OF GRID COMPUTING

Navonil Mustafee

Operational Research and Management Sciences
Warwick Business School
University of Warwick, Coventry, CV4 7AL, UK

Simon J E Taylor

Centre for Applied Simulation Modelling
School of Information Systems, Computing and Maths
Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

ABSTRACT

An increased need for collaborative research, together with continuing advances in communication technology and computer hardware, has facilitated the development of distributed systems that can provide users access to geographically dispersed computing resources that are administered in multiple computer domains. The term *grid computing*, or *grids*, is popularly used to refer to such distributed systems. Simulation is characterized by the need to run multiple sets of computationally intensive experiments. Large scale scientific simulations have traditionally been the primary benefactor of grid computing. The application of this technology to simulation in industry has, however, been negligible. This research investigates how grid technology can be effectively exploited by users to model simulations in industry. It introduces our desktop grid, *WinGrid*, and presents a case study conducted at a leading European investment bank. Results indicate that grid computing does indeed hold promise for simulation in industry.

1 INTRODUCTION

Grid computing has the potential to provide users on-demand access to large amounts of computing power, just as power grids provide users with consistent, pervasive, dependable and transparent access to electricity, irrespective of its source (Baker, Buyya, and Laforenza 2002). It has been identified that simulation modelling can potentially benefit from this as computing power can be an issue in the time taken to get results from a simulation (Taylor and Robinson 2006). Furthermore, development in simulation has been closely allied to the advances in the field of computing (Robinson 2005) and it is expected that it will continue to rely on the latest advances in computing to support increasingly large and complex simulations (Pidd and Carvalho 2006).

Grid computing is a significant advancement in the field of distributed computing and it is very likely that, like previous beneficial developments in computing adopted by simulation users, this technology may provide an opportunity to further improve the use of simulation in industry. This is supported by the observation that the use of grid computing in scientific simulation has certainly proved beneficial. For example, the role it plays in increasing the speed of simulations, store vast amounts of data generated, provide users secure access to data and application, etc. is certainly true in disciplines such as particle physics, climatology, astrophysics and medicine, among others. The question therefore is - can the same benefits be passed on to the use of simulation modelling as practiced in industry? In this paper we are primarily concerned with how computing resources made available through grid computing can be effectively used to execute simulation experiments faster (and therefore the opportunity to do more!).

For the benefit of the readers, the next section of this paper presents an involved discussion on grid computing. It includes the various definitions of grid computing, an overview of grid software (also referred to as grid middleware) and a discussion on the use of grids for research and in industry. The subsequent sections of this paper will be introduced at the end of section 2, since, by then, it is expected that the readers will have a good understanding of grid computing.

2 GRID COMPUTING

The grid vision of providing users continuous access to computing resources can be traced back to the *Multics* (Multiplexed Information and Computing Service) system that arguably discussed this in the context of time-sharing of a CPU among jobs of several users (Corbato and Vyssotsky 1965). The term "grid computing" was itself preceded by the term *metacomputing* which also advocated transparent user access to distributed and heterogeneous computing resources by linking such resources by software and an underlying network (Smarr and Catlett 1992).

Grid computing was first defined by Ian Foster and Carl Kesselman in their book *"The Grid: The Blueprint for a New Computing Infrastructure"* as a hardware and software infrastructure that provides access to high-end computational resources (Foster and Kesselman 1998). It was further stated that this access should be dependable, consistent, pervasive and inexpensive. This definition of grid computing has since been modified twice by the grid

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

veterans; once by Foster, Kesselman and Tuecke in their paper titled *"Anatomy of the Grid"* (Foster, Kesselman, and Tuecke 2001), and again by Foster and Kesselman with the publication of the second edition of their book *"The Grid: The Blueprint for a New Computing Infrastructure"* (Foster and Kesselman 2004). Re-definition of the term "grid computing" twice over the period of nearly 5 years suggests that this is still an evolving field. However, all the three definitions are consistent in terms of their focus on large-scale computing. Thus, Foster and Kesselman (1998) mention "access to high-end computational resources"; Foster, Kesselman, and Tuecke (2001) refer to "large-scale resource sharing" and, finally, Foster and Kesselman (2004) highlight "delivery of nontrivial QoS (Quality of Service)". This focus on large scale computing makes grid computing an enabling technology for *e-Science* (Hey and Trefethen 2002). The software component that makes grid computing possible is commonly referred to as grid middleware. A discussion on grid middleware and e-Science is presented in sections 2.1 and 2.2 respectively.

2.1 Grid Middleware

A grid middleware is a distributed computing software that integrates network-connected computing resources (computer clusters, data servers, standalone PCs, sensor networks, etc.), that may span multiple administrative domains, with the objective of making the combined resource pool available to user applications for number crunching, remote data access, remote application access, among others (Mustafee and Taylor 2008). A grid middleware is what makes grid computing possible. With multiple organizations involved in joint research collaborations, issues pertaining to security (authentication and authorization), resource management, job monitoring, secure file transfers, etc. are of paramount importance. Thus, in addition to making available a seamless distributed computing infrastructure to cater to the computing needs of the grid user, the grid middleware usually provides mechanisms for security, job submission, job monitoring, resource management and file transfers, among others. Example of grid middleware and the operating systems they support is presented in Table 1 in the adjacent column.

The next three sections of this paper will discuss the use of grid computing for large scale collaborative research (grid computing for e-Science), use of grid computing for relatively small scale personal research (grid computing for specialized users) and the use of this technology in industry.

2.2 Grid Computing for e-Science

e-Science is large scale science that is increasingly being carried out through global collaborations, and which requires access to very large data sets and computing resources distributed across a wide geographical area (National e-Science Centre 2001). An example of a notable e-Science project is the *Large Hadron Collider (LHC)* project at the European Organization for Nuclear Research (CERN) in Geneva (Lamanna 2004).

Table 1: Examples of grid middleware (MW)

MW	Description	Operating System
Globus (GT 4)	It is an open architecture and an open source set of services and software libraries that support grids & its applications (Foster, Kesselman, and Tuecke 2002).	UNIX, Linux and Windows. However, some components can only be run on UNIX and Linux platforms (Globus Alliance 2008).
European Data Grid (EDG)	EDG MW extends Globus to offer services like resource brokering and replication mgmt. (Berlich, Kunze, and Schwarz 2005).	The EDG MW has only been tested on *RedHat Linux 7.3* (EDG WP6 Integration Team 2003).
Condor	Condor is a job scheduling system that maximizes the utilization of networked PCs through identification of resources & schedules background user jobs on them (Litzkow, Livny, and Mutka 1988).	UNIX, Linux and Windows platforms. However, several Condor execution environments are not supported on Windows (Condor Version 6.9.1 Manual 2007).

It is interesting to note that all the e-Science projects mentioned above use grid computing to execute computer simulations. This is not an exception but generally the norm (i.e., e-Science projects usually make use of vast amounts of computing power, made available through grid computing, to run simulation experiments). As grid computing presents immense opportunities for e-Science projects, consequently the majority of grid users comprise of researchers and computer specialists who are associated with such e-Science projects and have the technical knowledge to work with the present generation grids. This is because the creation of an application that can benefit from grid computing (faster execution speed, linking of geographically separated resources, interoperation of software, etc.) typically requires the installation of complex supporting software and an in-depth knowledge of how this complex supporting software works (Jaesun and Daeyeon 2003).

But what about those researchers and academics who may not be associated with e-Science projects but nonetheless have a demand for non-trivial amounts of computing power for their own research (e.g., running Monte Carlo simulations to calculate credit risk, running experiments that simulate spread of infectious diseases, etc.)? For this rather specialized subset of users, grid resources made available as *Production Grids* can be the answer. This is described next.

2.3 Grid Computing for Personal Research

Production grids can be defined as grid computing infrastructures that have transitioned from being "research and development" test beds to being fully-functional grid environments, offering users round-the-clock availability at sustained throughput levels (Mustafee and Taylor 2008). Production grids are usually supported by a team that is responsible for the day-to-day maintenance of the grid (including upgrading software), solving technical problems associated with the grid, helping users through help-desk support, creating user documents, conducting training courses for knowledge dissemination purposes, among others. Table 2 lists the two largest production grids in the EU/UK.

Table 2: Examples of production grids

Grid	Infrastructure
EGEE, Europe	The EGEE project involves over 90 partner institutions across Europe, Asia and the US and provides access to over 20,000 CPU and 5PB of storage.
NGS, UK	NGS provides access to over 2,000 CPUs & over 36 TB storage capacities. These resources are provided by the Universities of Manchester, Leeds, Oxford, among others.

2.4 Grid Computing in Industry

The adoption of grid computing outside research projects has been limited. There are only a few examples in the literature of the use of grids in industry for inter-organizational collaborative work (i.e., access to shared resources for day to day operations of an organization) or collaborative research. Arguably, this is best illustrated by the fact that the majority of the papers related to "grid applications" that are listed on the website of Globus Alliance (Globus Alliance 2008), a well recognized community of organizations and individuals that are involved in the research and development of grid computing technologies, are about the use of grid computing in research projects. One exception to this is the Distributed Aircraft Maintenance Environment (DAME) project that has developed a distributed aircraft engine diagnosis environment as a proof of concept demonstration for Grid computing (Jackson et al. 2003).

However, it is also true that grid computing middleware like Globus is gradually being introduced within enterprises for processing enterprise-related applications. In this scheme, the organizations seek to leverage their existing computing resources using grid middleware. Collaborations, if any, are limited to intra-organizational resource sharing and problem solving. Some of the organizations that use grid computing middleware for their day-to-day operations or integrate these middleware with their own application is given in table 3 below.

Table 3: Examples of use of grid computing in industry

Company	Description
SAP R/3 *Application:* Internet Pricing and Configurator (IPC), Workforce Management (WFM) and Advanced Planner and Optimizer (APO) *Middleware:* Globus *Reference:* (Foster 2005)	IPC, WFM and APO applications are part of SAP's R/3 product line and are designed to support large numbers of requests generated by interactive clients using Web browsers or from batch processes. Each client request is dispatched to one of a number of worker processes. SAP has modified these applications to use Globus components to discover and reserve the resources used to host those worker processes, and to execute, monitor, and remove the worker processes on those resources.
GlobeXplorer *Application:* GlobeXplorer *Middleware:* Globus *Reference:* (Gentzsch 2004)	The data portrayed in the maps served by GlobeXplorer originate from multiple sources, e.g. population data, data on street networks, aerial images, satellite Imagery, etc. Globus provides the technology required to integrate data from such heterogeneous resource base.

As can be seen in the table above, the two applications use Globus middleware. This is to be expected since Globus is arguably the most recognized grid middleware and consists of open source set of services and software libraries which supports grids and grid applications (Foster et al. 2002). However, as can be gathered from Table 1 (section 2.1), not all Globus components can be installed on Windows computers. This is also true of other grid middleware like gLite, VDT, etc. The middleware mentioned in Table 1 are all geared towards dedicated, centralized, high performance clusters and supercomputers running on UNIX and Linux flavour operating systems.

It is important to adequately appreciate the difference between grid-based research activity, typified by grid computing for e-Science and for personal research, and profit-oriented commercial activity being undertaken in industry. It is usually the case that in industry the employees are experts in their own discipline but do not generally have the necessary technical skills that are required to work with present generation cluster-based grid technologies like Globus, VDT, gLite, etc. Cluster-based grid computing has a steeper learning curve, since a user generally benefits from learning grid installation procedures (which is normally very complex!) and the use of the middleware (mostly command line arguments), and may not appeal sufficiently enough to warrant widespread deployment in industry. Another reason for this is presented in Luther et al. (2005), namely, the middleware for cluster-based grid computing severely limits the ability to effectively utilize the vast majority of Windows-based resources that are common place in industry. Thus, Luther et al. (2005) suggest that development of middleware for

desktop-based grid computing is important with the growing industry interest in grids.

Development of Windows-based desktop grid middleware can potentially facilitate faster execution of many Windows-based applications (which would otherwise take a long time to execute on standalone machines) through effective utilization of resources that are made available by desktop grids. Examples of such applications are Commercial, Off-The-Shelf (COTS) Simulation Packages (CSPs), Audio/Video encoding programs and applications that render images. Windows-based desktop grid computing is discussed in the next section. This is followed by a brief description of WinGrid – a desktop grid middleware that was developed by the authors to facilitate integration with Windows-based applications. Section 5 highlights the use of simulation in industry and presents CSPs as exemplar Windows-based applications that can be grid-enabled and from which the industry stands to benefit. Section 6 presents a case study in which a CSP was grid-enabled to increase performance of Monte Carlo simulations manifolds. Section 7 draws the paper to a close.

3 DESKTOP GRID COMPUTING

Desktop-based grid computing (DGC) or *desktop grids* addresses the potential of harvesting the idle computing resources of desktop PCs (Choi et al. 2004). These resources can be part of the same local area network (LAN) or can be geographically dispersed and connected via a wide area network such as the Internet. Studies have shown that desktop PCs can be under utilized by as much as 75% of the time (Mutka 1992). This coupled with the widespread availability of desktop computers and the fact that the power of network, storage and computing resources is projected to double every 9, 12, and 18 months respectively (Casanova 2002), represents an enormous computing resource. The immediate implication of this is, software applications having non-trivial processing requirements can potentially run substantially faster using commonly available computing resources. In enterprises, this also means that the Return on Investment (RoI) of enterprise computing resources can also be potentially increased. We use the term *Enterprise Desktop Grid Computing (EDGC)* to refer to a desktop grid infrastructure that is confined to an institutional boundary, where the spare processing capacity of an enterprise's desktop PCs are used to support the execution of the enterprise's applications. User participation in such a grid is not usually voluntary and is governed by enterprise policy. Examples of EDGC middleware include Condor (Litzkow, Livny, and Mutka 1988), Platform LSF (Zhou 1992), Entropia DCGrid (Kondo, Chien, and Casanova 2004).

This section now discusses the ideal EDGC implementation for executing Windows-based applications in industry. In doing so, it takes into consideration the implementation and deployment aspects of the middleware. This discussion is informed by literature, by author's interactions with simulation experts and IT staff, and the author's own experience with implementing different grid-based solutions.

This discussion is structured under five specific categories. Four of these categories directly map to the implementation aspects of the middleware (over which a user usually has no control). These four categories refer to the *operating system* for which the middleware has been implemented, the number of ports that are opened by the middleware for *communication*, the *job scheduling mechanism* that is implemented and the *task farming* support that is provided by the middleware. The fifth category, namely, *application support*, is specific to the application that is being written to be executed over the grid and over which the user has some control. The programming language being used to implement the application is the important consideration here.

***Operating system category*:** EDGC middleware that can be installed on Windows PCs may be more appropriate for use with Windows-based applications.

***Communication category*:** In the confines of an organisation, security is a prime concern. It is therefore expected that the EDGC middleware that will open the least number of channels for communication (ports) has a greater chance of acceptance by the network administrators.

Job scheduling mechanism category: We may have the "pull" or the "push" job scheduling mechanism. If the EDGC middleware implements the "push" mechanism then it periodically polls the grid nodes to find out the load levels and decide on whether new jobs are to be assigned to the node; on the other hand, a middleware that implements the "pull" mechanism empowers the grid nodes to decide the best time to start a job and thereafter request a new job (Berlich, Kunze, and Schwarz 2005). This discussion now considers the efficiency of "pull", "push" and "broker-based" scheduling mechanisms in the enterprise environment. Garonne, Tsaregorodtsev, and Caron (2005) have conducted performance studies related to the efficiency of "pull" and "push" approaches in the context of scheduling tasks on multiple local schedulers that are shared among many users. The results have shown that, in terms of performance for High Throughput Computing (HTC), the centralized "push" approach is better than the decentralized "pull" approach under ideal conditions (e.g., no network or hardware failures, no disk space shortage, no service failure, etc.). Thus, the authors consider a "push" based scheduling mechanism to be a highly desirable characteristic in an EDGC middleware.

Task farming support category: In a task-parallel task farming application one master process is responsible for directing and coordinating the execution of multiple worker process and assimilation of the results; whereas in a job-parallel task farming application one application (or user) submits many jobs using standard middleware-

specific job submission mechanisms to submit a batch of jobs, which may be different instances of the same job or single instances of different jobs or both. For conducting simulation experiments, task-parallel applications will generally be better suited since one master process will be in control of the overall experimentation process. Thus, the simulation practitioner will usually be able to load the experiment parameters into the task-parallel application, which will in turn interact with the underlying grid middleware to schedule the experiments over different grid nodes, receive simulation results asynchronously from nodes, and finally collate the results and present them to the simulation user.

Application support category: Java is widely used to program enterprise applications in industry. It is generally accepted that the two important reasons contributing to its popularity and widespread use are, Java applications can be run on any operating system that has Java Runtime Environment (JRE) installed and Java is open source and available for free. Thus, in the application support category, it is arguable that a EDGC middleware that will be able to execute Java-based programs will be suitable for executing Windows-based applications in industry.

From the above discussion we gather that the ideal EDGC middleware for integrating Windows-based applications would be the one that is supported on Windows, which uses only one communication channel, implements the "push" job scheduling mechanism, supports task-parallel task farming applications and would support Java-based user applications. The authors implemented WinGrid (Mustafee and Taylor 2006, Mustafee et al. 2006, Mustafee 2007) with the view that it provides a reference implementation of a middleware that incorporates all the five ideal middleware characteristics.

4 WINGRID: THE DESKTOP GRID FOR WINDOWS

WinGrid is an EDGC middleware that is targeted at the Windows operating system. WinGrid incorporates the five ideal middleware characteristics that were identified in the last section and were considered important for executing Windows-based applications in industry. Thus, WinGrid is supported on Windows, it uses only one communication channel, it implements the "push" job scheduling mechanism, it supports task-parallel task farming applications and would support Java-based user applications.

The WinGrid middleware is based on the master-worker distributed computing architecture. WinGrid implements this "push" approach (master pushes the job to the workers) by starting a server process for each worker. The server process enables the worker to listen continuously for incoming tasks from the master. The presence of multiple servers transparently incorporates a degree of fault-tolerance to the WinGrid architecture as it means that processing over WinGrid continues even if one or more workers fail (computer hangs, PC re-boots etc).

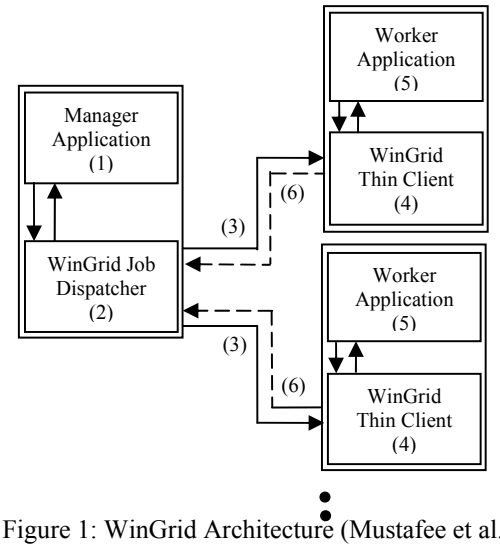

Figure 1: WinGrid Architecture (Mustafee et al. 2006)

WinGrid consists of four different parts: the *manager application* (MA), the *WinGrid Job Dispatcher* (WJD), the *worker application* (WA) and the *WinGrid Thin Client* (WTC). The MA runs on the manager computer (the application user's computer) and is software written specifically for the management of the application running over the desktop grid. The MA interacts with the WJD also running on the master computer and passes work to, and receives results from, the WJD. The WAs and WTCs run on each worker computer. The WJD sends and receives work to and from the WTCs. The WTCs in turn send and receive work to and from their WA. The WAs are unmodified application software (like CSPs) connected via a COM interface with the WTCs. The WTC is also responsible for advertising and monitoring local resources, accepting new jobs from the master process and returning back the results, and provides an interface through which the desktop user can set his preferences (when guest jobs are to be run, applications to share etc.). As seen in Figure 1, the user submits a job through the MA (1), which in turn interacts with the WJD process (2) in the manager computer to send work (3) to the WinGrid workers and their WTCs (4). The WTC pass this work to their WA for processing (5) and returns the result to the WJD (6). The results of all the sub-jobs are communicated back to the MA which then collates the results and presents it to the user.

Having described the architecture of WinGrid, the next section highlights one exemplar Windows-based application in industry that has the potential to effectively utilize the computing resources made available through it. This application is the *Commercial, Off-The-Shelf Simulation Package* (or *CSPs* for short).

5 CSP: EXEMPLAR APPLICATION FOR INTEGRATION WITH DGC

A possible means of increasing adoption of desktop grid computing (DGC) in industry is to incorporate grid support in Windows-based software applications that (1) require non-trivial amounts of computation power and (2) that are used by the end-users to perform their day-to-day jobs. As has been discussed earlier, the application area of simulation is one such area that demands extensive use of computation cycles. CSPs are generally used to build and execute simulations in industry. Also, they are extensively used by simulation practitioners since the CSPs incorporate many user-friendly features like visual interface, animation, inbuilt programming language, etc. By virtue of these two characteristics CSPs are considered as an exemplar application for integration with DGC middleware.

In this paper the term CSP is used to refer to software used for modelling both Discrete Event Simulation (DES) and Monte Carlo Simulation (MCS). Examples of such software include commercially available DES packages like Witness (Lanner group), Simul8 (Simul8 corporation), and AnyLogic (XJ technologies). Similarly, MCS may be modelled in a visual environment using spreadsheet software like Excel (Microsoft), Lotus 1-2-3 (IBM, formerly Lotus Software); spreadsheet add-ins, for example @Risk (Palisade Corporation; or through MCS-specific simulation packages such as Analytica (Lumina Decision Systems) and Analytics (SunGard).

6 CASE STUDY: GRID ENABLING MONTE CARLO SIMULATION (MCS)

This case study was conducted at a large European investment bank and investigates the performance gains that could potentially be derived through integration of CSPs with WingGrid (in both dedicated and non-dedicated modes). The investment bank uses CSP Analytics for Monte Carlo-based credit risk simulations of counterparty transactions.

Credit risk simulations are usually used to calculate the credit exposure over a period of time. CSP Analytics is the calculation engine for the Credient credit risk system that provides algorithms to calculate time-dependent profiles of credit exposure using MCSs (Credient Analytics 2007). Analytics consists of Analytics Server COM Object (essentially a COM interface to Analytics) and can be invoked by external systems. Analytics Desktop application is installed on multiple workstations within the credit risk division of the investment bank. It is currently used to support five different financial products, namely, currency swaps, default swaps, forward rate agreements, interest rate swaps (IRS) and risky bond forwards (RBF).

6.1 IRS-RBP Simulation Application

The investment bank uses the IRS-RBF application to simulate five different financial products. This application comprises of different Excel spreadsheets, VBA modules and CSP Analytics. Analytics is invoked by the VBA modules (present in the Excel spreadsheets) through the Analytics Server COM Object. The application takes its name from two different products, namely, Interest Rate Swaps and Risky Bond Forwards, which it simulates.

Simulations of the financial products are a two-stage process. In the first stage, risk profiles are generated by invoking Analytics through Excel. The parameters passed-on include different currency codes like GBP, INR and USD. Analytics outputs the results of the simulation in the form of text files. The first stage is subsequently referred to as the ***generate profiles stage***.

In the second stage, referred to as the ***create table stage***, PFE and EPE tables are generated by Excel. These tables are based on the values present in the text files that are created in the generate profiles stage. PFE or *Potential Future Expose* is the maximum amount of counterparty exposure (i.e., the maximum outstanding obligation if counterparties were to default) that is expected to occur on a future date with a high degree of statistical confidence; EPE or *Expected Positive Exposure* is the average counterparty exposure in a certain interval, e.g., a month or a year (Canabarro and Duffie 2003).

Stage one and stage two processing of the IRS-RBF application involves three distinct operations that have to be "manually-executed". These operations are (1) generate profiles, (2) create EPE tables, and (3) create PFE tables. The EPE/PFE create table operations can only start after successful execution of the generate profile operation. The time taken to execute both these phases for the IRS-RBF application is shown in Table 4. The total number of currencies simulated by the application is also indicated. The data for this table has been provided by the credit risk analysts who have developed the IRS-RBF application.

Table 4: Execution time for different products using the original IRS-RBF application

Products	Generate Profiles	Create Tables	Currencies
Interest Rate Swaps (IRS)	1 hour 15 minutes	12 hours	23
Risky Bond Forwards (RBF)	4 hours 30 minutes	1 hour 20 minutes	13

The numbers of currencies that are simulated by these products are 23 and 13 respectively. Ideally, the bank would expect to run the IRS and RBF simulations with 37 currencies. This would further increase the execution time. The authors therefore saw the potential of using WinGrid to speed up the simulation of the IRS-RBP application at the investment bank.

6.2 Grid-enabling IRS-RBF Simulation

For the IRS-RBF application to utilize the resources made available through WinGrid, it has to be integrated to the WTC and the WJD (please refer to Figure 1). Integration of the Excel-based IRS-RBF application with WTC is achieved using Excel's COM interface. A custom built *IRS-RBF adapter* has been developed which encapsulates the COM function calls required by WTC to interact with the IRS-RBF application. In the WinGrid architecture, the IRS-RBF application is the Worker Application (WA).

In this case study the WinGrid Master Application (MA) that controls the IRS and RBF simulation execution is called the *WJD Application Specific Parameter (ASP) Tool for IRS-RBF application*. It is an Excel-based tool that consists of specific parameters that are required for processing the IRS-RBF application; for example, the name of the output directory, the name of the product to simulate (IRS or RBF), the operation to perform (create table, create profiles or both), the filename to simulate, etc. WJD APS tool also consists of two other worksheets, namely *"RBF"* and *"IRS"*. These worksheets contain data specific to the RBF and the IRS simulations respectively. Each worksheet has a list of currencies. Each currency is a separate unit of computation (job). The interaction between the MA and WJD is by means of an *Excel Adapter*. This adapter contains specific COM calls required by WJD to access the MA.

6.3 Results

Identical IRS-RBF experiments for this case study were conducted on, (1) one dedicated WinGrid node (running both WJD and WTC), (2) 4 non-dedicated WinGrid nodes connected through the investment bank's corporate LAN, and (3) 8 non-dedicated WinGrid nodes connected with the corporate LAN. The grid-enabled IRS-RBF application was used for running experiments over the different test beds. The reasons for not using the original IRS-RBF application for execution over one dedicated, standalone PC was that the original IRS-RBF application was modified to a large extent by the author to enable faster execution of the grid-version of the application.

The experiments were conducted over a period of two days during normal working hours of the investment bank. The 4-node and the 8-node WinGrid experiments were run using production machines that were also being used by the analysts to do their jobs. The one node experiments were conducted using a PC that was not being used. The computers had a 2.13GHz Pentium II or a 2.99GHz Intel Pentium IV processor with 512MB / 2GB RAM, and were running on Microsoft XP Professional. The dedicated WinGrid node used for performing the standalone experiments had a 2.99GHz HTT Intel Pentium IV processor with 512MB RAM.

The results of the IRS and RBF simulations are presented in Figure 2. These results are based on two separate runs for each workload. The execution of all the four workloads, pertaining to either IRS or RBF simulation, was fastest using the 8 non-dedicated WinGrid nodes. The slowest execution was recorded by the standalone, dedicated WinGrid node.

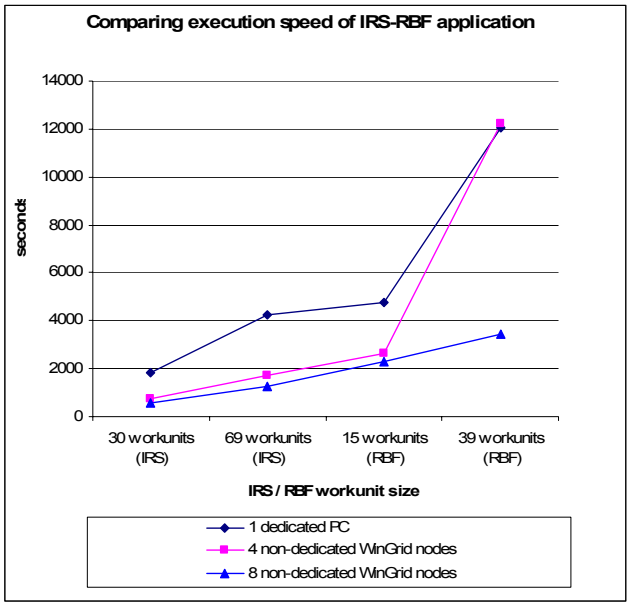

Figure 2: Time taken to execute the IRS-RBF application using different workloads

6.4 Discussion

For workloads *[30 workunits (IRS)]*, *[69 workunits (IRS)]* and *[15 workunits (RBF)]* the time taken to execute the IRS-RBF simulations using the 4 node WinGrid test bed was comparable to its 8 node counterpart. One reason for this may be that, with 8 nodes the number of Excel files created in Phase 2 (create EPE table) and Phase 3 (create PFE table) of the workflow are double the number of Excel files created when running the simulation using 4 nodes. Thus, the sequential MA operation in phases 4 and 5 (collate data from the EPE and PFE tables) would generally take more time in the case of the former. An additional reason could be the specific usage pattern of the PCs during the experiments. It is therefore possible that the majority of the PCs in the 8 node set-up had their WTC clients manually or automatically shut down because the analysts were using the computers for their own work. The WTC program can be shut down manually through WinGrid's graphical user interface. This can also happen automatically as the WTC program is designed to continuously monitor CPU and the memory usage on a PC, and if the resource usage crosses the pre-determined CPU/RAM threshold levels then the user jobs are immediately stopped. Similarly, jobs are started automatically again when the CPU and memory usage decreases as a result of a resource not being used. Thus, the time taken to execute the simulations on non-dedicated WinGrid nodes is very much related to the usage pattern of

the underlying desktop PCs. Arguably, this is best shown by the results of workload *[30 workunits (RBF)]* in relation to its execution over 4 non-dedicated WinGrid nodes, where the time taken to complete the simulation is comparable to that of its standalone counterpart.

7 CONCLUSION

The research presented in this paper has been motivated by the advances being made in the field of grid computing and the realization that simulation in industry could potentially benefit through the use of grid computing technologies. This research recognises that end-user adoption of grids could be facilitated by focusing on software tools that are commonly used by employees at their workplace. In the context of simulation in industry, the end-users are the simulation practitioners and the tools that are generally used to model simulations are the Commercial, Off-The-Shelf (COTS) Simulation Packages (CSPs). Thus, this research has investigated how grid computing can further the field of CSP-based simulation practice and, thereby, offer some benefits to simulation end-users.

This paper has introduced WinGrid, a desktop grid computing middleware specifically designed for executing Windows-based applications on Windows platform. The use of WinGrid to support execution of IRS-RBF Monte Carlo simulations based on CSP Analytics has also been discussed. The performance results of the grid-enabled version of the IRS-RBF simulation have been presented. The speed-up that this promises over the small desktop grids at the investment bank, and the ease with which grid enabling has been accomplished, will give users of the IRS-RBF simulation a competitive advantage as results will be delivered significantly faster with minimum technological intervention It is hoped that this paper will focus attention on the benefit that small desktop grids can give to simulation modelling and to industry as a whole.

ACKNOWLEDGEMENTS

The authors would like to thank Jonathan Berryman, Rahul Talwalkar and Robert Watson from the investment bank (counterparty risk management group) for their help with the case study.

REFERENCES

Baker, M., Buyya, R. and Laforenza, D. 2002. Grids and grid technologies for wide-area distributed computing. *Software - Practice and Experience*, 32(15): 1437-1466.

Berlich, R., Kunze, M. and Schwarz, K. 2005. Grid computing in Europe: from research to deployment. In Proceedings of the 2005 Australasian Workshop on Grid Computing and e-Research, pp. 21-27. Australian Computer Society, Darlinghurst, Australia.

Canabarro, E. and Duffie, D. 2003. *Measuring and marking counterparty risk*. In Tilman, L.M. (ed.), Asset/Liability Management of Financial Institutions (chapter 9). London, UK: Euromoney books. Online <http://www.stanford.edu/~duffie/Chapter_09.pdf> Last accessed 15th June 2008.

Casanova, H. 2002. Distributed computing research issues in grid computing. *ACM SIGACT News*, 33(3): 50-70. ACM Press, New York, NY, USA.

Choi, S., Baik, M., Hwang, C., Gil, J. and Yu, H. 2004. Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment. In *Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications*, pp. 366-371. IEEE Computer Society, Washington, DC, USA.

Condor Version 6.9.1 Manual. 2007. Platform-specific information on Windows, Condor 6.9.2 manual. Website <www.cs.wisc.edu/condor/manual/v6.9/6_2Microsoft_Windows.html>. Last accessed 15th June 2008.

Corbato, F. J. and Vyssotsky, V. A. 1965. Introduction and overview of the Multics system. In *Proceedings of the AFIPS Fall Joint Computer Conference*, pp. 185–196. IEEE Educational Activities Department, Piscataway, NJ, USA.

Credient Analytics. 2007. Credit risk management system - Credient Analytics 2.3 user guide. SunGard Corporation <http://www3.sungard.com/financial/>.

EDG WP6 Integration Team. 2003. Data grid installation guide. Document identifier: DataGrid-06-TED-0105-2-0. Available <http://marianne.in2p3.fr/datagrid/documentation/EDG-Installation-Guide-2.0.pdf>. Last accessed 15th June 2008.

Foster, I. 2005. A globus primer (draft version). Available <www.globus.org/toolkit/docs/4.0/key/>. Last accessed 15th June 2008.

Foster, I. and Kesselman, C. 1998. *The grid: blueprint for a new computing infrastructure.* San Francisco, CA: Morgan Kaufmann.

Foster, I. and Kesselman, C. 2004. Concepts and architecture. In Foster, I. and Kesselman, C. (eds.), The Grid: Blueprint for a New Computing Infrastructure (2nd Edition), chapter 4. San Francisco, CA: Morgan Kaufmann.

Foster, I., Kesselman, C. and Tuecke, S. 2001. The anatomy of the grid: enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3): 200-222.

Foster, I., Kesselman, C., Nick, J. M. and Tuecke, S. 2002. Grid services for distributed system integration. *IEEE Computer,* 35(6): 37-46.

Garonne, V., Tsaregorodtsev, A. and Caron, E. 2005. A study of meta-scheduling architectures for high throughput computing: pull versus push. In *Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPDC'05),* pp. 226-233. IEEE Computer Society, Washington, DC, USA.

Gentzsch, W. 2004. Enterprise resource management: applications in research and industry. In Foster, I. and Kesselman, C. (eds.), The Grid: Blueprint for a New Computing Infrastructure (2nd Edition), chapter 12. San Francisco, CA: Morgan Kaufmann.

Globus Alliance. 2008. Research papers from globus alliance members. Website <http://www.globus.org/alliance/publications/papers.php#Applications>. Last accessed 15th June 2008.

Hey, T. and Trefethen A. E. 2002. The UK e-science core programme and the grid. *Future Generation Computer Systems,* 18(8): 1017-1031.

Jackson, T., Austin, J., Fletcher, M. and Jessop, M. 2003. Delivering a grid enabled distributed aircraft maintenance environment (DAME). In Proceedings of the 2003 UK e-Science All Hands Meeting, pp. 420-427. Available online <http://www.nesc.ac.uk/events/ahm2003/AHMCD/>. Last accessed 15th June 2008.

Jaesun, H. and Daeyeon, P. 2003. A lightweight personal grid using a supernode network. In: Proceedings of the 3rd *International Conference on Peer-to-Peer Computing,* pp. 168-175.

Kondo, D., Chien, A. and Casanova, H. 2004. Resource management for rapid application turnaround on enterprise desktop grids. In *Proceedings of the 2004 Conference on Supercomputing (SC'04)*, paper 17. IEEE Computer Society, Washington, DC, USA.

Lamanna, M. 2004.The LHC computing grid project at CERN. *Nuclear Instruments and Methods in Physics Research (Section A: Accelerators, Spectrometers, Detectors and Associated Equipment),* 534(1-2): 1-6.

Litzkow, M., Livny, M. and Mutka, M. 1988. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems,* pp.104-111. IEEE Computer Society, Washington, DC, USA.

Luther, A., Buyya, R., Ranjan, R. and Venugopal, S. 2005. Alchemi: a .NET-based enterprise grid computing system. In *Proceedings of the 6th International Conference on Internet Computing (ICOMP'05)*, pp. 269-278. CSREA Press, USA.

Mustafee, N. 2007. A grid computing framework for commercial simulation packages. PhD thesis. School of Information Systems, Computing and Mathematics, Brunel University, UK.

Mustafee, N. and Taylor, S.J.E. 2006. Using a Desktop Grid to Support Simulation Modelling. In *Proceedings of the 28th Information Technology Interfaces Conference (ITI 2006)*, Dubrovnik, Croatia. June 19-22, 2006. pp. 557-562.

Mustafee, N. and Taylor, S. J. E. 2008. Investigating grid computing technologies for use with commercial simulation packages. In *Proceedings of the 2008 Operational Research Society Simulation Workshop (SW08),* Worcestershire, UK. April 1-2, 2008. pp. 297-307.

Mustafee, N., Alstad, A., Larsen, B., Taylor, S.J.E. and Ladbrook, J. 2006. Grid-enabling FIRST: Speeding Up Simulation Applications Using WinGrid. In *Proceedings of the 10th International Symposium on Distributed Simulation and Real Time Applications (DSRT 2006)*, Malaga, Spain. October 2-6, 2006. pp. 157-164.

Mutka, M. W. 1992. Estimating capacity for sharing in a privately owned workstation environment. *IEEE Transactions on Software Engineering,* 18(4): 319-328.

National e-Science Centre. 2001. Defining e-Science. Website <http://www.nesc.ac.uk/nesc/define.html> Last accessed 15th June 2008.

Pidd, M. and Carvalho, M. A. 2006. Simulation software: not the same yesterday, today or forever. *Journal of Simulation*, 1(1): 7-20.

Robinson, S. 2005. Discrete-event simulation: from the pioneers to the present, what next? *Journal of the Operational Research Society*, 56 (6): 619-629.

Smarr, L. and Catlett, C. E. 1992. Metacomputing. *Communications of the ACM*, 35(6): 44-52.

Taylor, S. J. E. and Robinson, S. 2006. So where to next? A survey of the future for discrete-event simulation. *Journal of Simulation*, 1(1): 1-6.

Zhou, S. 1992. LSF: Load sharing in large-scale heterogeneous distributed systems. In *Proceedings of the 1992 Workshop on Cluster Computing*. Supercomputing Computations Research Institute, Florida State University, Florida, USA.

AUTHOR BIOGRAPHIES

NAVONIL MUSTAFEE is a research fellow in Warwick Business School. His research interests are in parallel and distributed simulation, grid computing and health care simulation. He completed his PhD in Information Systems and Computing Brunel University in 2007. He is a member of the drafting group of the COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) under the Simulation Interoperability Standards Organization. <navonil.mustafee@wbs.ac.uk>.

SIMON J. E. TAYLOR is the co-founding Editor-in-Chief of the UK Operational Research Society's (ORS) *Journal of Simulation* and the Simulation Workshop series. He has served as the Chair of the ORS Simulation Study Group between 1996 to 2006 and was appointed Chair of ACM's Special Interest Group on Simulation (SIGSIM) in 2005. He is also the Founder and Chair of the COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) under the Simulation Interoperability Standards Organization. He is a Senior Lecturer in the Centre for Applied Simulation Modelling in the School of Information Systems, Computing and Mathematics at Brunel. <simon.taylor@brunel.ac.uk>.