

Speeding Up Simulation Applications Using WinGrid

Navonil Mustafee
Operational Research and Management Sciences Group
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 Mathematics
Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

Abstract. *The vision of grid computing is to make computational power, storage capacity, data and applications available to users as readily as electricity and other utilities. Grid infrastructures and applications have traditionally been geared towards dedicated, centralized, high performance clusters running on UNIX “flavour” operating systems (commonly referred to as cluster-based grid computing). This can be contrasted with desktop-based grid computing which refers to the aggregation of non-dedicated, de-centralized, commodity PCs connected through a network and running (mostly) the Microsoft Windows operating system. Large scale adoption of such Windows-based grid infrastructure may be facilitated via grid-enabling existing Windows applications. This paper presents the WinGrid approach to grid enabling existing Windows-based Commercial-Off-The-Shelf (COTS) simulation packages (CSPs). Through the use of two case studies developed in conjunction with a major automotive company and a leading investment bank respectively, the contribution of this paper is the demonstration of how experimentation with the CSP Witness (Lanner Group) and the CSP Analytics (SunGard Corporation) can achieve speedup when using WinGrid middleware on both dedicated and non-dedicated grid nodes. It is hoped that this research would facilitate wider acceptance of desktop grid computing among enterprises interested in a low-intervention technological solution to speeding up their existing simulations.*

Keywords. Grid computing, desktop grids, grid middleware, commercial-off-the-shelf simulation packages, discrete-event simulation, Monte Carlo simulation.

1. Introduction

Grids are sharing environments implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of distributed communities and sharing of resources like computers, storage space, sensors, software applications and data between them (Foster and Iamnitchi, 2003). These distributed communities, frequently referred to as virtual organizations, or virtual enterprises, comprise of a group of individuals and/or institutions engaged in some joint work who share resources based on strict sharing policies that define what is shared, who is allowed to share and the conditions under which such sharing occurs (Foster et al., 2002).

Computer simulation has the potential to benefit from sharing access to computing resources, storage capacities and research equipments provided by grid computing. Examples of large scale grid-based simulation projects include the Earth Grid System (Bernholdt et al., 2005) and NEESgrid (Spencer et al., 2004). The creation of such applications typically requires the

installation of complex supporting software (like Globus) and an in-depth knowledge of how this complex supporting software works (Jaesun et al., 2003).

The exponential growth of global computer ownership, local networks and Internet connectivity, coupled with the fact that desktop PCs in corporate and home environments are heavily under utilized, has given rise to enterprise/desktop grid computing, public resource computing and peer-to-peer (P2P) computing – all of which are different forms of Internet computing (Luther et al., 2005). Internet computing seeks to provide resource virtualization through aggregation of idle CPU cycles of the PCs connected over the Internet and the Local Area Network (LAN). When this form of computing is confined to an enterprise and the purpose of resource virtualization is to support the execution of enterprises' applications then we use the term *enterprise desktop grids* (Chien et al., 2003).

Desktop grid applications such as Platform LSF (Zhou, 1992), Entropia DCGrid (Kondo et al., 2004), United Devices GridMP (United Devices, 2007) and Digipede Network (Digipede Technologies, 2006) are increasingly being deployed within enterprises to tap into their PC-based networks and maximize return on investment (ROI) on computing resources. In order to increase the enterprise-wide adoption of Windows-based grid technologies, it is also imperative to develop new grid software to specifically deal with Windows issues and grid-enable existing Windows applications to encourage adoption. With regards to the former, for example, a .NET-based grid computing framework called Alchemi has been developed that provides the runtime machinery and programming environment required to construct Windows-based desktop grids and develop grid applications (Luther et al., 2005). As for the latter, it requires development of a grid-enabling solution that requires little or no change to existing Windows applications. Our system *WinGrid* (Mustafee and Taylor, 2006; Mustafee et al., 2006; Mustafee, 2007) aims to deliver such a low intervention technological solution to grid-enable existing Windows applications.

In this paper we discuss how WinGrid can benefit users of *Commercial-off-the-shelf Simulation Packages* (CSPs). CSPs are modelling support environments software widely used by simulation practitioners in the industry. Examples of CSPs include Arena™ (Rockwell Automation), AnyLogic™ (XJ Technologies), Automod™ (Brooks Software), Promodel™ (ProModel Corporation), Simul8™ (Simul8 corporation), Witness (Lanner Group) and Analytics (SunGard Corporation). Users of these packages tend to be skilled in simulation and not computer science (as many users of Grid computing are). Vendors of CSPs change the functionality of their CSPs on an incremental basis. Major possible changes to their packages are often prohibitively costly and do not have a guaranteed ROI.

The practice of CSP-based simulation can widely benefit from Grid computing (Taylor et al., 2004; Mustafee, 2007). By means of two case studies, done in conjunction with a major automobile manufacturing company (the Ford Motor Company) and a leading European investment bank respectively, we investigate how a desktop grid implemented with our system WinGrid can increase the performance of both discrete event and Monte Carlo simulation experimentation. Our approach differs to previous attempts to use distributed computing to speed up simulation experimentation (Anagnostopoulos et al., 2003; Biles and Kleijnen, 2003; Paris et al., 2001; Yücesan et al., 2001) by using a desktop grid specifically aimed at Windows applications and by transparently, in as much as possible, grid-enabling simulation within an enterprise context (i.e. by changing the existing simulation application as little as possible to encourage adoption of this technology).

The paper is structured as follows. In section 2 we review the current approaches to desktop grids. The WinGrid architecture is described in Section 3. Section 4 discusses the case study that was conducted together with Ford and how a Witness-based application called FIRST was grid-enabled using WinGrid. This is followed by the investment bank case study in which WinGrid was used to speed up Monte Carlo credit risk simulations (section 5). The results of both the case studies are presented in section 6. Section 7 draws the paper to a close.

2. Desktop Grids

While much of Grid computing is focussed on meeting the needs of large virtual organizations, *Desktop Grid Computing* 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 ROI of enterprise computing resources can also be potentially increased.

Two principal types of desktop grids have emerged. These are *Public Resource Computing* and *Enterprise Desktop Grid Computing*. Both these are based on variants of the master/workers distributed computing architecture (Chakravarti et al., 2004). In such a model a user launches an application on a master computer that is responsible for allotting work generated by the application to the available worker computers for processing. The individual results are returned by the workers to the master for compilation by the application and presentation to the user.

2.1 Public Resource Computing

Public-resource computing (PRC) refers to the utilization of desktop grids comprising millions of desktop computers primarily to do scientific research (Anderson, 2004). The Berkeley Open Infrastructure for Network Computing (BOINC) (University of California, 2006) is the most widely used desktop grid application that supports scientific projects with diverse objectives such as searching for evidence of extraterrestrial intelligence, studying climate change, improvement in the design of particle accelerators, finding cures for human diseases and searching for gravitational waves from space. Non-BOINC based projects use their own software to facilitate research with similar objectives, for example, finding a cure to cancer (Parabon computation, 2006), understanding protein folding (Pande, 2006) and computing mersenne prime numbers (Woltman, 2006). The participants of PRC projects are volunteers who register with one or more such projects and install the required desktop grid software. This software then contacts the central project servers and downloads work units for processing (in case of BOINC it also downloads project specific executable code as BOINC is a general purpose PRC client). The time it takes to complete the execution of a work unit and return back the result depends, among other things, on the machine hardware, the amount of time a PC is left running and user preferences. The volunteers are themselves unable to use the underlying desktop grid infrastructure, of which they themselves are part of, to perform their own computations.

2.2 Enterprise Desktop Grid Computing

We use the term *Enterprise Desktop Grid Computing* (EDGC) to refer to a 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. Applications such as CONDOR (Litzkow et al., 1988), Platform LSF (Zhou, 1992), Entropia DCGrid (Kondo et al., 2004), United Devices GridMP (United Devices, 2007) and Digipede Network (Digipede Technologies, 2006) are all examples of EDGC. Unlike the PRC model these applications usually allow users to submit jobs for processing.

2.3 Desktop Grids and CSPs

How can a desktop grid support the needs of CSP experimentation? To recap, our aim is to create a system that takes into account that these packages are Windows-based, their users are specialists in simulation and not computing and any technological solution must be developed with little or no change to the CSP.

Building on PRC and EDGC, one possibility is to “bundle” the CSP along with each desktop grid worker. Thus, whenever a desktop grid worker is started the CSP is also loaded. In an enterprise desktop grid the worker usually runs in a “sandbox”. We call this sandbox the Desktop Grid Virtual Machine (DGVM) and this provides logically separate, secure execution environment for both the host and guest processes.

An alternative solution would be to install the CSP in the worker nodes as a normal application and then have the master communicate directly with that application. The drawback with this is that the sandbox security mechanism which is present in most EDGC approaches would have to be forfeited. However, as simulations are created by *trusted* employees running *trusted* software within the bounds of a firewalled network, security in this open access scheme could be argued as being irrelevant (i.e. if it were an issue then it is an issue with the wider security system and not the desktop grid).

In the next section we consider the suitability of the existing approaches to desktop grids, namely PRC and EDGC, with regard to their use with CSPs in an enterprise environment. For this we select BOINC and Condor as representative forms of PRC and EDGC middleware respectively. As has been mentioned earlier, BOINC is an open source PRC middleware that allows users to create new BOINC-based projects to cater to their computational needs. Condor is an EDGC middleware that is used for both high throughput computing and for enterprise application processing. Both BOINC and Condor are cycle stealing systems (i.e., a system that harnesses the unused CPU cycles of idle PCs to process other jobs in the background) that can run on non-dedicated Windows PCs.

The rationale of selecting BOINC as a representative form of PRC middleware is as follows.

- It is presently the most popular PRC middleware.
- It is presently the only PRC middleware that allows users to create their own projects.
- It is free.

The rationale of selecting Condor as a representative form of EDGC middleware is as follows:

- It has the largest EDGC deployment base. More than 80,000 Condor hosts around the world make up approximately 160 production-level Condor pools (see <http://www.cs.wisc.edu/condor/map/> for updated Condor statistics).
- It is free (but contributions to the CONDOR project are encouraged). Other EDGC middleware like Entropia DCGrid, United Devices GridMP and Digipede Network are commercial solutions.

2.4 Selection Criteria

This section discusses the selection criteria for grid middleware for CSP-based simulation in industry, and tests the suitability of BOINC and Condor in this regard. In doing so, it takes into consideration the implementation and deployment aspects of the middleware. This discussion is informed by literature, by the authors' interactions with simulation experts and IT staff, and the authors' own experiences 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) and are considered important when deciding upon the suitability of the middleware for deployment for CSP-based simulations in industry. 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: A grid middleware that can be installed on Windows PCs may be more appropriate for use with CSPs (almost all the CSPs only run on Windows).

Communication category: In the confines of an organisation, security is a prime concern. It is therefore expected that the grid 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, push or broker job scheduling mechanisms. If the 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 allows the grid nodes to decide the best time to start a job and thereafter request a new job (Berlich et al., 2005). In a "broker-based" approach to job scheduling a software process (for example, the matchmaking agent in Condor) is responsible for matching jobs with available resources. After a match has been found, the resource consumer may "push" the job to the resource provider (Robinson and DeWitt, 2007). Let us now consider the efficiency of "pull", "push" and "broker-based" scheduling mechanisms in the enterprise environment. Garonne et al. (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, 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.). Similarly, a “broker-based” scheduling approach will generally be less efficient than the “push” based approach because the former introduces one more layer of communication between the nodes requesting resources and the nodes providing those resources.

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, single instances of different jobs or both. For conducting CSP-based 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 grid middleware that will be able to execute Java-based programs will be suitable for CSP-based simulation in the industry. Table 1 shows BOINC and Condor specific information regarding each of the five categories discussed above.

In the remaining part of this section, we show that neither BOINC nor Condor can be considered as the ideal middleware implementation for supporting CSP-based simulation in industry under all the five categories. For example, under the operating system category Condor was found suitable for deployment; under the communication category BOINC, which uses the standard HTTP port for all its communication, was considered suitable since it does not require opening up extra ports; in the job scheduling mechanism category neither of the middleware were considered ideal since they did not implement the “push” model of job scheduling; in the task farming support category, however, both BOINC and Condor were considered appropriate since both the middleware supported task-parallel task farming applications; finally, in the application support category, the use of Condor with Java execution environment was considered appropriate. Table 2 below summarizes this information.

The table shows that neither Condor nor BOINC has an ideal middleware implementation for running CSP-based simulation in industry. The ideal middleware would be the one which 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) specifically for the purpose of executing CSP-based simulation over desktop grid. WinGrid is discussed in the next section.

3. WinGrid Architecture

WinGrid is an EDGC middleware that is targeted at the Windows operating system. The primary motivation for implementing WinGrid was to provide an ideal middleware implementation for supporting CSP-based simulations in industry. As such, WinGrid incorporates the five ideal middleware criteria that were identified in the last section and were considered important for grid-based simulations 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 support Java-based user applications.

WinGrid is based on the master-worker distributed computing architecture. This architecture (also referred to as task farming architecture) consists of one master entity and multiple workers entities, wherein the master entity decomposes the problem into small tasks, distributes these tasks among a farm of worker processes and gathers the partial results to produce the final result of the computation; and the worker entities receive message from the master with the next task, process the task and send back the result to the master (Heymann et al., 2000). 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). We now discuss the different components of WinGrid.

As shown in Figure 1, 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 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.). A 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. The reader is referred to Mustafee (2007) for an in-depth discussion on WinGrid, CSP-WinGrid integration technology and WinGrid interfaces.

4. Case Study: Grid enabling FIRST

The Ford Motor Company makes use of computer simulation to design new engine manufacturing facilities and for process improvement in routine day-to-day operations.. The production of an engine is a complex operation as it involves the manufacture and assembly of a wide variety of components into several possible engine types based on orders from the customer (Taylor et al., 2008). Using simulation in this process helps to experiment with different machine configurations, buffer capacities, changeover schemes (switching

production from one engine type to another), shift patterns, machine downtime, etc., and contributes to ensuring a smooth work-flow in the engine production line.

Ford uses the CSP Witness at the Dunton Engineering Center in the UK. Wider adoption of simulation has been hindered due to the lack of expertise required in using Witness. Like any other CSP such knowledge is normally acquired over a period of time. In order to encourage faster adoption of simulation, the company felt the requirement for an application which would make it easier and quicker for people to use simulation (Ladbrook and Januszczak, 2001). As a response to this the FIRST application was developed by the company with assistance from the Lanner Group suppliers of Witness.

4.1 The Fast Interactive Replacement Simulation Tool (FIRST)

Fast Interactive Replacement Simulation Tool (FIRST) is a proprietary tool that builds a Witness model of an engine manufacturing line based on data input through Microsoft Excel. The Excel-based application consists of more than 30 worksheets, 10 VBA modules and many Excel macros. It uses Visual Basic for Application (VBA) to interface between Excel and the Witness CSP, and dramatically cuts down the time it takes to build and run a Witness simulation model by automating much of the process of model building.

To build a manufacturing line in Witness through FIRST, the application has to be provided inputs like the number of machines, corresponding buffer sizes, time and frequency of tool change, changeovers, shift patterns, user defined distributions, warm-up period, experimentation period etc. Once all the data has been entered and the “Run Simulation” button clicked (see Figure 2), the model is remotely built in Witness and the simulation starts. Results of the simulation are returned back to FIRST and are displayed using various Excel-based mechanisms like tables, graphs (see Figure 3), conditional formatting, etc. FIRST is under continued development and new features are added to suit the requirements of the modellers at the automobile manufacturing company.

4.2 Speeding up experimentation using FIRST

The complexity of an engine manufacturing line means that a number of experiment scenarios may have to be run before an ideal solution can be identified. Each run would require setting experiment values using FIRST and then executing the model to determine the outcome. This commences with the process of parsing the various Excel worksheets (defined within the application) and executing appropriate Witness commands with arguments based on the extracted values. This, in turn, progressively builds the Witness model, and when the model is complete, Witness starts simulating it. The time taken to generate the model using FIRST is dependent upon the amount of data to be parsed. For example, in case of large models comprising multiple manufacturing lines it may take as long as 10-15 minutes to modify the model (re-parameterise for experimentation) and up to 60 minutes to run it. If 10 different scenarios were to be experimented using FIRST then the execution time is approximately 11 to 12 hours to finish all the experiments using one computer. Keeping in mind the fact that the company has multiple Witness licences which can be accessed from any computers, it would be reasonable to assume that the time taken to build and conduct multiple simulation experiments can be significantly reduced by utilizing all the available computing resources. One way to achieve this is through pooling unused resources by means of a desktop grid infrastructure and interfacing the FIRST application with it. This case study with the

automobile manufacturing company looks at how WinGrid was used to speed up experimentation using FIRST.

4.3 Grid-enabling FIRST using WinGrid

In order to grid-enable FIRST we integrated it with the WTC using the Component Object Model (COM). COM is a Microsoft technology that allows different software components to communicate with each other by means of interfaces (Gray et al., 1998). Since FIRST is an Excel-based application we have access to its COM interface. A custom built *FIRST adapter* has been developed which encapsulates the COM function calls required by WTC to interact with the FIRST application. In the WinGrid architecture FIRST is the WA.

For the purpose of experimenting with multiple simulation scenarios, we have created an Excel spreadsheet based controller called *FIRST Experiment tool* which lists all the experiment parameters (as an integrated add-on to FIRST) (Figure 4). The First Experiment tool is the MA and it interacts with the WJD to send different parameters for experimentation to different FIRST applications through their corresponding WTCs. Once a FIRST application has completed simulating a model, it sends back to the MA the result it received from Witness. This communication is done through the corresponding WTCs and the WJD. For each result received by the FIRST application tool a new worksheet is created and the values stored. The worksheets are named according to the experiment numbers. 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 MA. A screenshot of the FIRST experiment tool is shown below. The example shows experimentation with the various buffer sizes of the machines.

Since WinGrid is written in Java (a non-COM compliant language), we have used Java Native Interface technology (Sun Microsystems, 2003) for communication between Excel Adapter, WinGrid and the First Adapter. Figure 5 shows the integration architecture of WinGrid and FIRST.

5. Case Study: Grid enabling IRS-IBF simulation

The investment bank uses CSP Analytics for Monte Carlo-based credit risk simulations of counterparty transactions. The transactions between the investment bank and the counterparties may involve agreements to exchange different sequences of payments over a period of time. Credit risk is the potential that the counterparty will fail to meet its obligations in accordance with the agreed terms (Basel Committee on Banking Supervision, 1999).

In assessing credit risk from a single counterparty, an institution (in our case the investment bank) must consider three issues (Contingency Analysis, 2003):

- (1) Default probability: What is the likelihood that the counterparty will default on its obligation either over the life of the obligation or over some specified time period?
- (2) Credit exposure: In the event of a default, how large will the outstanding obligation be when the default occurs?
- (3) Recovery rate: In the event of a default, what fraction of the exposure may be recovered through bankruptcy proceedings or some other form of settlement?

Credit risk simulations are usually used to calculate the credit exposure over a period of time. Analytics is the calculation engine for the Credent credit risk system that provides algorithms to calculate time-dependent profiles of credit exposure using MCSs (Credent Analytics, 2007). Analytics consists of three separate applications, namely, Analytics Desktop, Market Data Manager (MDM) and Analytics Server COM Object. The Analytics Desktop application is a standalone application that uses a calculation engine to construct and analyse financial portfolios. It links to the Market Data Manager to derive both current and historical market data which serve as inputs to these calculations. Analytics Server COM Object is essentially a COM interface to the Analytics Desktop 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).

5.1 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 IRS-RBF application takes its name from two different products, namely, Interest Rate Swaps (IRS) and Risky Bond Forwards (RBF), which it simulates. The name has been given by the author to represent the collective components that logically make up this application.

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 Exposure* 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 3. The total number of currencies used for simulating these products is also indicated. The data for this table has been provided by the credit risk analysts who have developed the IRS-RBF application.

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 means that the execution time will be further increased. It has been discussed in the

automobile company case study (section 4) that WinGrid could be used to reduce execution time of simulation experiments over dedicated nodes. WinGrid could, therefore, arguably be used for the investment bank case study to speed up the IRS-RBF application. However, unlike the automobile manufacturing company, where the simulation department had access to dedicated resources over which to run their simulations, the computers being used by the credit risk division of the investment bank are non-dedicated resources. These resources are the desktop PCs that are used by the credit risk analysts at their work place. Thus, for this case study WinGrid would have to be executed over non-dedicated PCs.

5.2 Grid-enabling the 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. 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). Further discussion on WinGrid architecture can be found in section 3.

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* (figure 6). 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, whether the WJD process had crashed during an earlier run, etc. All this information is present in the worksheet called "*General*". The 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. The WinGrid and IRS-RBF application integration architecture is similar to the one presented in Figure 5.

6. Results

The results from the automobile manufacturing company case study and the investment bank case study are now presented.

6.1 Results: Automobile manufacturing company case study

In order to evaluate the performance of FIRST over WinGrid, a 4-node experimental test bed was set up consisting of PCs with PIII 648 MHz processors and 256MB RAM, connected through an isolated 100Mbps switch. Three of these nodes were configured as WinGrid workers and were installed with WTC, Witness, the FIRST application and FIRST adapter. The fourth PC served as the WinGrid master and had the WJD, FIRST Experimentation Tool and Excel adapter installed on it.

In our example FIRST application, preset values automatically built a Witness model consisting of one main and one supplementary assembly line. The data present in FIRST provided, among other details, the number of machines in each assembly line and their

corresponding buffer sizes. To test our approach, it was decided to conduct multiple experiments with FIRST over WinGrid by varying the size of the buffer, such that each experiment was conducted using a different set of buffer parameters and was run to a preset simulation time. The FIRST experimentation tool defined the buffer capacities of each machine in the main assembly line for all the experiments that were to be conducted. The performance was measured in terms of the time taken to execute 25, 50, 75 and 100 runs of the experiment respectively. So as to demonstrate the potential of achieving speedup when using FIRST over WinGrid, the same experiments were repeated using a standalone version of FIRST. An Excel spreadsheet similar to FIRST Experimentation Tool was used to automate the running of the standalone version. The results obtained by the 4-node WinGrid version and the standalone version of FIRST are shown in figure 7 below.

The results show that the 4-node WinGrid version of FIRST completes execution of all the experiments approximately three times faster when compared to the standalone execution. This is to be expected since three WTCs are processing jobs sent by the master computer and are dedicated to this task.

6.2 Results: Investment bank case study

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 were as follows.

- The original IRS-RBF application was modified to a large extent by the authors to enable faster execution of the grid-version of the application.
- To run the IRS and RBF simulations using the original application meant that three different operations (create profiles, create EPE tables and create PFE tables) had to be manually invoked by the user. The execution of the grid-version of this application, on the other hand, was fully automated.

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 configurations of the machines used for the experiments are shown in table 4 below.

The dedicated WinGrid node used for performing the standalone experiments had a 2.99GHz HTT Intel Pentium IV processor with 512MB RAM. The 4 non-dedicated WinGrid nodes used for the experiments comprised of different subsets of the machines at different times. The results of the IRS and RBF simulations are presented in figure 8. 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.

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. Conclusions

This paper has introduced WinGrid, a desktop grid computing middleware specifically designed for executing CSP-based simulations on Windows computers. We have discussed the use of WinGrid in support of FIRST, a specialist simulation application using the CSP Witness, and the IRS-RBF simulation, a credit risk simulation application using the CSP Analytics. A Grid-enabled version of FIRST and the IRS-RBF simulation has been developed with minimum technological intervention. We have presented the performance results of the grid-enabled version of FIRST and the IRS-RBF simulation. The speed-up that this promises over the small desktop grids at the automobile company and the investment bank, and the ease with which grid enabling has been accomplished, will give users of FIRST and the IRS-RBF simulation a competitive advantage as results will be delivered significantly faster with minimum technological intervention (i.e. by completely re-implementing FIRST / IRS-RBF simulation). It is hoped that this paper will focus attention on the benefit that small desktop grids can give to simulation in industry as a whole.

Acknowledgements

The authors would like to thank Tony Waller, product manager at Lanner, for granting the use of Witness licenses in support of our research. For help with the case studies, thanks are due to John Ladbroke from the Ford Motor Company; Jonathan Berryman, Rahul Talwalkar and Robert Watson from the counterparty risk management group at the investment bank; and Anders Alstad and Bjørn Larsen from Brunel University.

References

- Anagnostopoulos D. and Nikolaidou M. (2003) Executing a Minimum Number of Replications to Support the Reliability of FRTS Predictions. In Proceedings of the 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications, October 23 – 25 2003, Delft, The Netherlands, 138-146.
- Anderson D.P. (2004) BOINC: a system for public-resource computing and storage. In Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, November 2004, 4-10.

- Basel Committee on Banking Supervision (1999). Principles for the management of credit risk. Available online <http://www.bis.org/publ/bcbs54.pdf>. [9/5/2008]
- Beckles B., Se-Chang S. and Kewley J. (2005) Current methods for negotiating firewalls for the Condor system. In Proceedings of the 4th UK e-Science All Hands Meeting. Available online <http://www.cs.wisc.edu/condor/doc/CondorandFirewalls.pdf>. [9/5/2008]
- 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, 21-27. Australian Computer Society, Darlinghurst, Australia.
- Bernholdt D., Bharathi S., Brown D., Chanchio K., Chen M., Chervenak A., Cinquini L., Drach B., Foster I. and Fox P. (2005). The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. In Proceedings of the IEEE, 93(3), 485-495.
- Biles W.E and Kleijnen J.P.C. (2003) Statistical Methodology for WEB-Based Simulation. In Proceedings of the 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications, October 23 – 25 2003, Delft, The Netherlands, 147-149.
- 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. Available http://www.stanford.edu/~duffie/Chapter_09.pdf. [9/5/2008]

- Casanova H. (2002) Distributed computing research issues in grid computing. *ACM SIGACT News*, 33(3), 50-70.
- Chakravarti A.J., Baumgartner G. and Lauria M. (2004) Application-specific scheduling for the organic grid. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*; November 2004, 146-155.
- Chien A. A., Calder B., Elbert S. and Bhatia K. (2003). Entropia: architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing*, 63(5), 597-610.
- 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*, August 2004, 366-371.
- Contingency Analysis. (2003). Credit risk glossary. Website http://www.riskglossary.com/link/credit_risk.htm, [9/5/2008]
- Credent Analytics. (2007). Credit risk management system - Credent Analytics 2.3 user guide. SunGard Corporation <http://www3.sungard.com/financial/>, [9/5/2008]
- Digipede Technologies. (2006) The digipede network. Website <http://www.digipede.net/products/digipede-network.html>. [9/5/2008]
- Foster I. and Iamnitchi A. (2003) On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*; February 21-22, 2003; Berkeley, CA, USA; 118–128.
- Foster I., Kesselman J., Nick J. and Tuecke S. (2002). The physiology of the grid: An open grid services architecture for distributed systems integration. *Open Grid Service Infrastructure WG*, Global Grid Forum, June 2002.
- 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.
- Gray D.N., Hotchkiss J., LaForge S., Shalit A. and Weinberg T. (1988) Modern languages and Microsoft's component object model, *Communications of the ACM*, 41(5), 55-65.
- Heymann E., Senar M.A., Luque E. and Livny M. (2000) Adaptive scheduling for master-worker applications on the computational grid. *Grid Computing - GRID 2000*, Lecture Notes in Computer Science; R. Buyya and M. Baker, Eds.; Springer Berlin / Heidelberg, 214–227.
- 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*, Sept 1-3 2003, 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.
- Ladbrook J. and Januszczak A. (2001). Ford's Power Train Operations – Changing the Simulation Environment. In *Proceedings of the 33rd Winter Simulation Conference*, December 9 – 12 2001, Arlington, Virginia, 863-869.
- 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*, June 1988, 104-111.
- 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)*, June 2005, Las Vegas, USA, 27-30.

- 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. Using a desktop grid to support simulation modelling. In Proceedings of the 28th Information Technology Interfaces Conference (ITI2006), June 19-22 2006, Dubrovnik, Croatia, 557-562.
- 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), Stiffler, V.L. and Dobric, V. H. (eds.), pp. 557-562. IEEE Computer Society, Washington, DC, USA.
- Mustafee, N., Alstad, A., Larsen, B., Taylor, S. J. E., 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), Alba, E., Turner, S. J., Roberts, D. and Taylor, S. J. E. (eds.), pp. 157-164. IEEE Computer Society, Washington, DC, USA.
- Mutka M.W. (1992) Estimating capacity for sharing in a privately owned workstation environment. IEEE Transactions on Software Engineering, 18(4), 319-328.
- Pande V. (2008) Folding@Home, <http://folding.stanford.edu/> [9/5/2008].
- Parabon computation Inc. (2008) Compute against cancer, www.computeagainstcancer.org/ [9/5/2008].
- Paris J.L. and Pierreval H. (2001) A Distributed Evolutionary Simulation Optimization Approach for Configuration of Multiproduct Kanban Systems. International Journal of Computer Integrated Manufacturing, 14 (5), 421-430.
- Robinson E. and DeWitt D. J. (2007). Turning cluster management into data management: a system overview. In Proceedings of 3rd Biennial Conference on Innovative Data Systems Research (CIDR). Available online <http://www-db.cs.wisc.edu/cidr/cidr2007/papers/cidr07p14.pdf>. [9/5/2008]
- Spencer B., Finholt T., Foster I., Kesselman C., Beldica C., Futrelle J., Gullapalli S., Hubbard P., Liming L. and Marcusiu D. (2004) Neesgrid: A distributed collaboratory for advanced earthquake engineering experiment and simulation. In Proceedings of the 13th World Conference on Earthquake Engineering, August 1-6 2004, Vancouver, BC, Canada, Paper No. 1674.
- Sun Microsystems Ltd. (2003) Java Native Interface. <http://java.sun.com/j2se/1.4.2/docs/guide/jni> [9/5/2008].
- Taylor S.J.E., Bohli L., Wang X., Turner S.J. and Ladbrook J. (2005) Investigating Distributed Simulation at the Ford Motor Company. In Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications, October 10-12 2005, Montreal, Quebec, Canada, 139-147.
- Taylor, S.J.E., Pullen, J.M., Popescu, G.V. and Turner, S.J. (2004) Panel on Distributed Simulation and the Grid. In Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications, October 21 – 23 2004, Budapest, Hungary, 144-149.
- United Devices (2007) Grid MP: The technology for enterprise application virtualization. Website <http://www.ud.com/products/gridmp.php>. [9/5/2008]
- University of California (2008) Berkeley Open Infrastructure for Network Computing, <http://boinc.berkeley.edu/> [9/5/2008].
- Woltman G. (2008) GIMPS, <http://www.mersenne.org> [9/5/2008].
- Yücesan E., Luo Y.C., Chen C.H. and Lee I.. (2001) Distributed Web-Based Simulation Experiments for Optimization. Simulation Practice and Theory, 9(1), 73-90.

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.

Tables and Figures

Table 1: Implementation and deployment aspects of BOINC and Condor (adapted from Mustafee, 2007)

Category	BOINC	Condor
Operating system	UNIX / Linux to host BOINC server. The clients can be Windows based.	Supported on Windows. Some components are only supported in Unix / Linux, but for CSP-specific services Windows installation is adequate.
Communication	Uses port 80	Uses multiple, bi-directional, static and dynamic ports (Beckles et al, 2005).
Job scheduling mechanism	“pull” based model of the master-worker architecture	Implements “broker-based” job scheduling mechanism (a Condor matchmaker agent acts as a broker and tries to find a match between the resource consumer agents and the resource provider agents).
Task farming support	Supports task-parallel applications	Supports job-parallel applications. Condor Master-Worker (Condor MW) supports both job-parallel and task-parallel applications.
Application support	Supports applications written using C++. User applications have to be compiled with the BOINC client C++ APIs.	Different Condor universes support user applications written in C, C++ and Java. For creating job-parallel and task-parallel applications, the user applications have to be compiled with the C++ Condor MW library.

Table 2: Ideal middleware implementation for CSP-based simulation

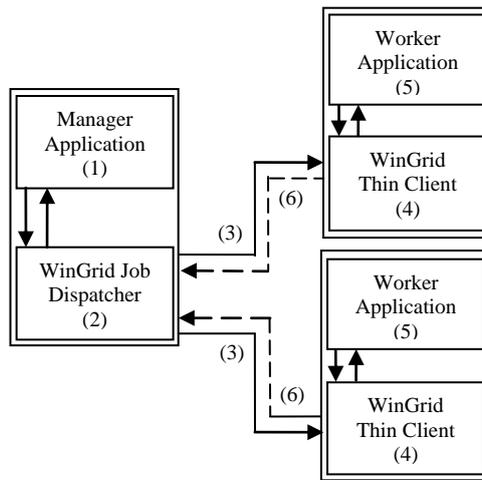
Category	Criteria	Middleware that implements this feature
Operating system	Middleware is supported on Windows operating system	Condor
Communication	Middleware opens only one communication port	BOINC
Job scheduling mechanism	Middleware implements “push” job scheduling mechanism	None
Task farming support	Middleware supports task-parallel task farming applications	BOINC and Condor
Application support	Middleware supports Java-based user applications	Condor with Java execution environment

Table 3: 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

Table 4: Configuration of WinGrid nodes

PC no.	CPU	RAM	Operating System
PC1	2.99GHz Intel Pentium IV Processor (hyper-threaded)	512MB	Microsoft XP Professional
PC2	2.99GHz Intel Pentium IV Processor (hyper-threaded)	512MB	Microsoft XP Professional
PC3	2.79GHz Intel Pentium IV Processor (hyper-threaded)	512MB	Microsoft XP Professional
PC4	2.13GHz Intel Pentium II Processor (hyper-threaded)	2GB	Microsoft XP Professional
PC5	2.13GHz Intel Pentium II Processor (hyper-threaded)	2GB	Microsoft XP Professional
PC6	2.13GHz Intel Pentium II Processor (hyper-threaded)	2GB	Microsoft XP Professional
PC7	2.13GHz Intel Pentium II Processor (hyper-threaded)	2GB	Microsoft XP Professional
PC8	2.13GHz Intel Pentium II Processor (hyper-threaded)	2GB	Microsoft XP Professional



•

Figure 1: WinGrid Architecture

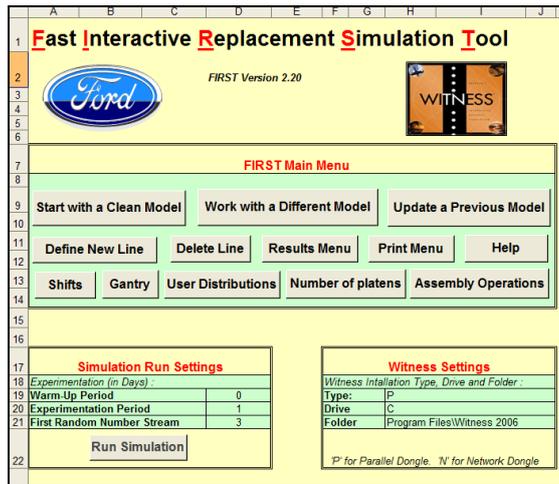


Figure 2: FIRST application main menu

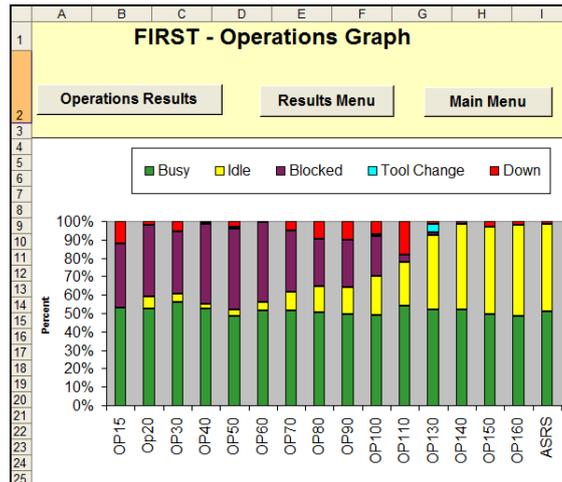


Figure 3: Graph generated by FIRST using data returned by Witness

	A	B	C	D	E	F	G	H
1	FIRST Experimentation Tool							
2								
3	Buffer Capacity							
4	Buffer Na	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6	Exp 7
5	PreOP15	80	20	30	40	50	60	70
6	PreOp20	18	13	15	17	19	21	23
7	PreOP30	11	10	9	8	7	6	5
8	PreOP40	57	55	50	45	40	35	30
9	PreOP50	12	13	14	15	16	17	18
10	PreOP60	13	13	13	13	13	13	13
11	PreOP70	13	15	15	13	13	16	16
12	PreOP80	18	19	23	27	31	35	39
13	PreOP90	62	60	58	56	54	52	50
14	PreOP100	19	17	15	15	17	19	15
15	PreOP110	21	21	21	21	21	21	21
16	PreOP130	16	20	22	25	28	31	34
17	PreOP140	1	2	3	1	2	3	1
18	PreOP150	16	17	18	15	16	14	12
19	PreOP160	19	20	22	24	26	12	14
20	PreASRS	23	20	21	11	20	15	14
21	File / WS	Q:\Results						

Figure 4: FIRST Experimentation Tool showing a list of experiments

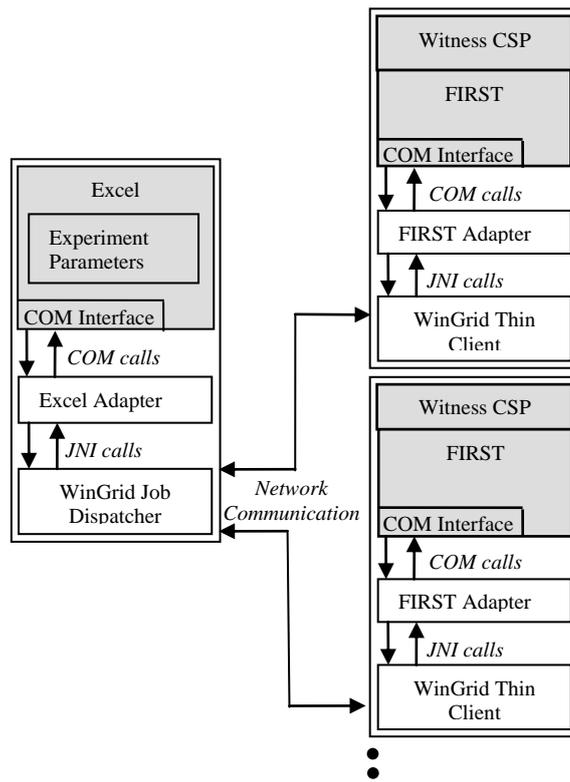


Figure 5: Architecture of WinGrid and FIRST

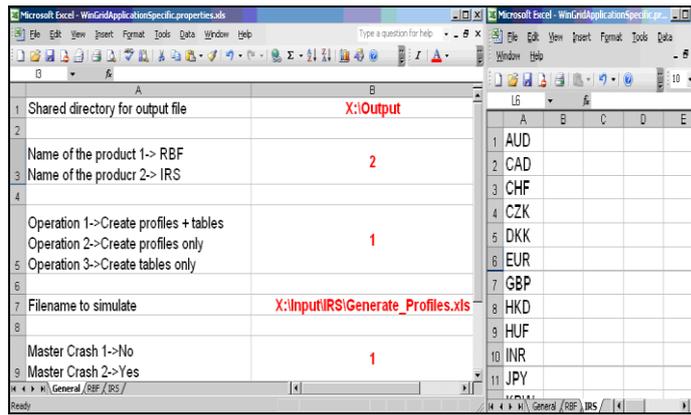


Figure 6: WJD Application Specific Parameter (APS) tool for IRS-RBF application

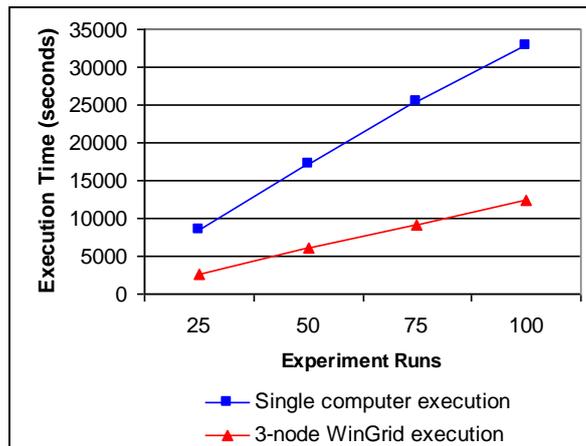


Figure 7: Time taken to build and simulate Witness using models using the FIRST application

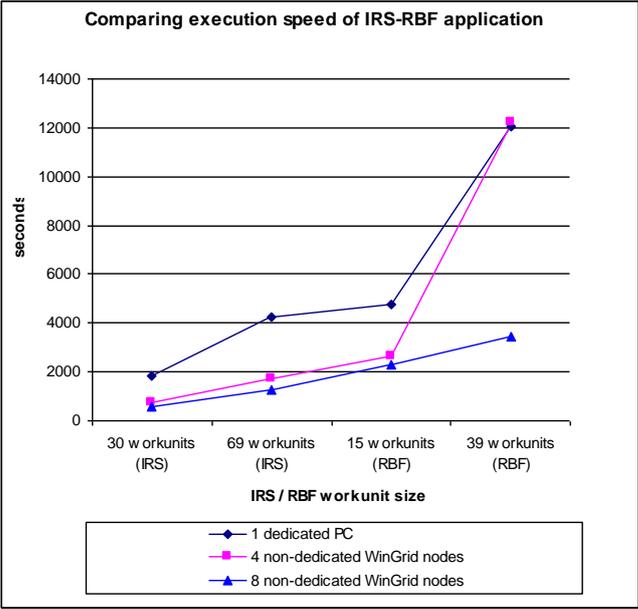


Figure 8: Time taken to execute the IRS-RBF application using different workloads