Grid-enabling FIRST: Speeding Up Simulation Applications Using WinGrid

Navonil Mustafee, Anders Alstad, Bjorn Larsen, Simon J E Taylor
Centre for Applied Simulation Modelling, Brunel University
Uxbridge, Middlesex, UB8 3PH, UK

John Ladbrook
Ford Motor Company,Dunton Engineering Centre
Laindon, Basildon, Essex, SS15 6EE, 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 a case study developed in conjunction with Ford Motor Company, the paper demonstrates how experimentation with the CSP Witness™ and FIRST can achieve a linear speedup when WinGrid™ is used to harness idle PC computing resources. This, combined with the lessons learned from the case study, has encouraged us to develop the web service extensions to WinGrid™. It is hoped that this would facilitate wider acceptance of WinGrid™ among enterprises having stringent security policies in place.

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 [1]. 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 [2].

Simulation modelling is a field that 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 [3] and NEESgrid [4]. 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 [5].

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 underutilized, 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 [6]. 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* [7].

Windows™-based desktop grid applications like DCGrid [8], GridMP [9] and Platform LSF [10] 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 (1) develop new grid

Proceedings of the Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'06)
0-7695-2697-7/06 $20.00 © 2006 IEEE

software to specifically deal with Windows™ issues and (2) 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 [6]. 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*™ [11] 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 *COTS Simulation Packages* (CSPs). CSPs are visual interactive modelling software widely used by simulation practitioners in the industry. Examples of CSPs include Arena™, AnyLogic™, Automod™, Promodel™, Simul8™ and Witness™. Users of these packages tend to be skilled in simulation modelling 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.

Taylor, et al. [12] identified that CSPs and the practice of simulation modelling can widely benefit from Grid computing. By means of a case study with the Ford Motor Company we investigate how a desktop grid implemented with our system WinGrid™ can increase the performance of simulation experimentation. Our approach differs to previous attempts to use distributed computing to speed up simulation experimentation [13,14,15,16] 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 relevant current approaches to Desktop Grids. The WinGrid™ architecture is described in Section 3. Section 4 discusses the Ford case study and how a Witness™-based application called FIRST was grid-enabled using WinGrid™. This is followed by experimentation and presentation of results in Section 5. Section 6 discusses the lessons learned from Ford and the web services extension to WinGrid™. 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 [17]. 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 [18]. Given the number of desktop computers across the world, this represents an enormous computing resource. The immediate implication of this is that software applications can potentially run substantially faster. 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 [19]. 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 [20]. Berkeley Open Infrastructure for Network Computing (BOINC) [21] 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 [22], understanding protein folding [23] and computing mersenne prime numbers [24]. 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 and is 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 like CONDOR [25], Platform LSF [10], DCGrid [8] and GridMP [9] 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 modelling 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.

In DCGrid for example, the DGVM is called the Entropia Virtual Machine (EVM) and it wraps interpreters like cmd.exe, perl and Java Virtual Machine to prevent unauthorized access to a computer [26]. Thus, it might be possible to include a CSP installation inside the EVM and offer it as part of an Entropia installation. In this case the master will need to send the data files associated with the simulation and a script file to trigger the CSP execution in the worker DGVM. The simulation results would be collected in a file, which would then be sent back to the master. The problem with this approach is that it would require major changes to the CSP (such as integrating CSP into a DGVM). For a vendor this would be prohibitively costly.

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).

Let us now consider our approach to supporting simulation with desktop grids by introducing our *WinGrid™*.

3. WinGrid™ Architecture

The WinGrid™ middleware supports EDGC and 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 [27]. 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™.

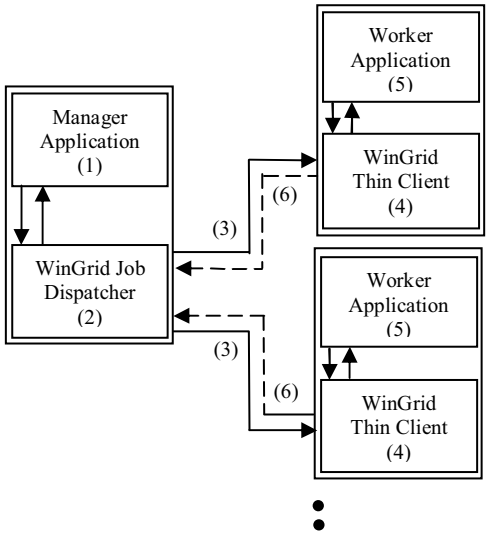


Figure 1: WinGrid architecture

Proceedings of the Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'06)
0-7695-2697-7/06 $20.00 © 2006 IEEE

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.). As seen in Fig. 1 above, 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.

Although this multiple-server based approach works well (as demonstrated in section 5), the requirement of starting one server process per worker is sometimes seen as a security-threat by organizations (as we learnt in our Ford case study). The alternative to this can be to implement a single-server based "pull" approach (the workers pull job from the master) as it requires starting only one server process for the master. In this case the server listens continually for incoming task requests from the workers. However, the presence of one server implies that the system becomes dependent on one computer. If this master computer crashes then no processing can take place. The "pull" architecture, implemented through web services extension to WinGrid™, will be discussed in section 6.

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 at Ford as it involves the manufacture and assembly of a wide variety of components into several possible engine types based on orders from the customer [28]. 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 Essex. 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, Ford felt the requirement for an application which would make it easier and quicker for people to use simulation [29]. As a response to this the FIRST application was developed by Ford with assistance from the Lanner Group, the suppliers of Witness™.

4.1 The Fast Interactive Replacement Simulation Tool (FIRST)

Fast Interactive Replacement Simulation Tool (FIRST) is a Ford 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 with 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 Ford.

4.2 Possibility of speeding up experimentation using FIRST

The complexity of an engine manufacturing line at Ford 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 Ford has multiple Witness™ licences which can be accessed from many 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 Ford 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 [30]. 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). 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

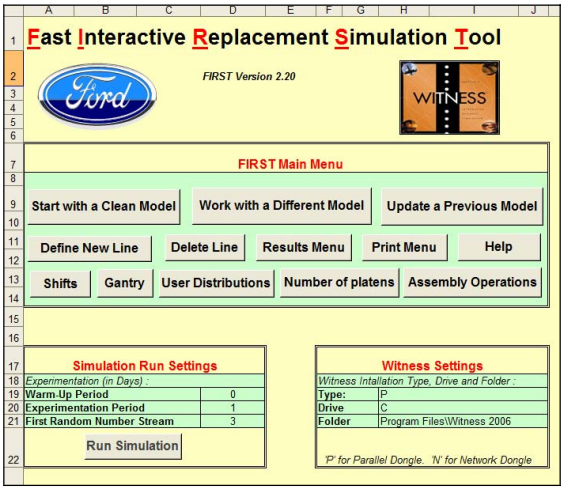


Figure 2: FIRST application main menu

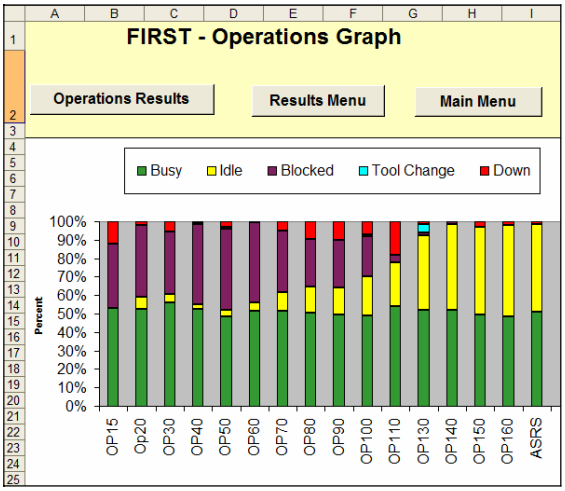
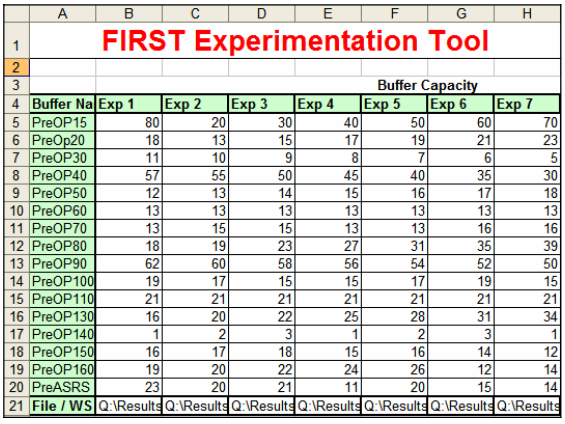


Figure 3: Graph generated by FIRST using data returned by Witness™



FIRST Experimentation Tool

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

Figure 4: FIRST experimentation tool showing a list of experiments

Excel Adapter. This adapter contains specific COM calls required by WJD to access MA. A screenshot of the FIRST experiment tool is shown in figure 4. 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 [31] for communication between Excel Adapter, WinGrid™ and the First Adapter. Fig. 5 shows the integration architecture of WinGrid™ and FIRST.

5. Results

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.

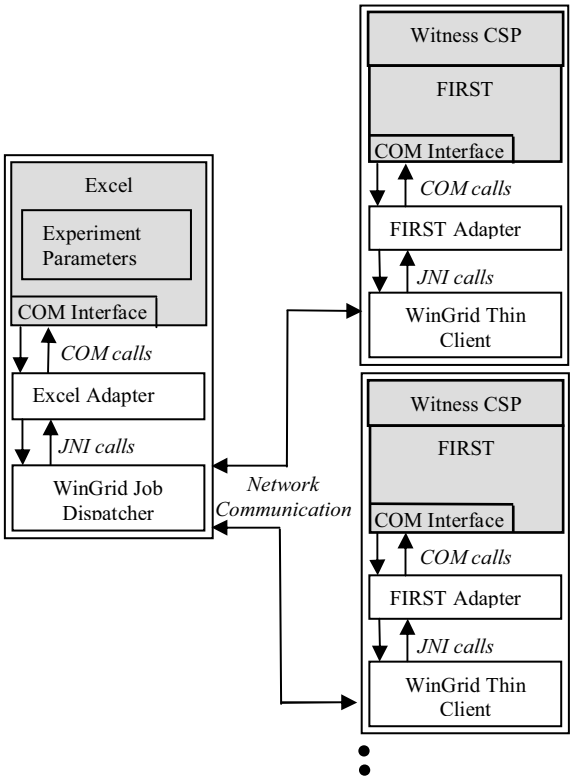


Figure 5: Architecture of WinGrid and First

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 (see figure 4) 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 below.

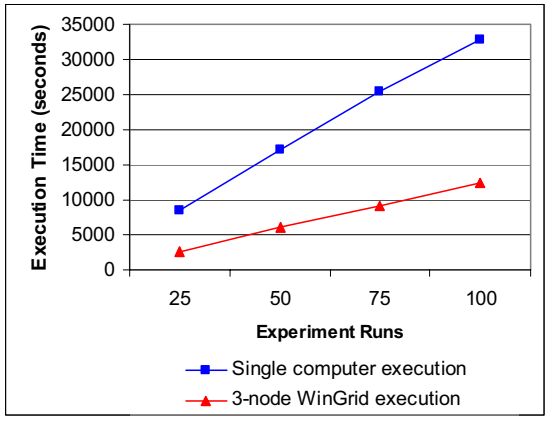


Figure 6: Time taken to build and simulate Witness™ models using FIRST application

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. Web services extension to WinGrid™

The results of the experiments demonstrated the potential of grid-enabled FIRST application to speed up the process of simulation experimentation within Ford. The logical next step was to deploy WinGrid™ and grid-enabled FIRST on computers at Dunton, demonstrate the application to the engineers of the group and ask for feedback to further develop the WinGrid-FIRST

integration so that it could be effectively used by engineers. Doing this required the approval of IT support at Ford as they are responsible for maintaining existing hardware, installing software and securing computer systems within the organization. Several issues were identified through negotiations with the IT support with regards to WinGrid™ deployment at Ford. First of all, Ford did not allow any kind of server software to be installed on the office computers. Another requirement from Ford was the use of web services for communication between PCs. Web services enable application interaction using standard Internet protocols and open standards. In other words, a web service is accessible on the same terms as any other resource on the Internet. Since desktops at Ford have Internet access (with current security policy), web services deployed on any web server should also be accessible.

Both these constraints ruled out the deployment of the current WinGrid™ implementation at Ford. WinGrid™ did not fulfil the first criteria because WTC has inbuilt server functionality and was meant to be installed on individual office PCs. The use of sockets for communication between the WinGrid™ nodes meant that it failed to satisfy the second requirement too.

It was realized that for deployment of WinGrid™ to be possible at Ford, the existing "push" based architecture had to be substantially changed and requirements imposed by Ford incorporated into the system. The modified architecture is based on web services and is called web services extension to WinGrid™, or WinGrid-WS™ in short. The architecture of WinGrid-WS™ is presented below.

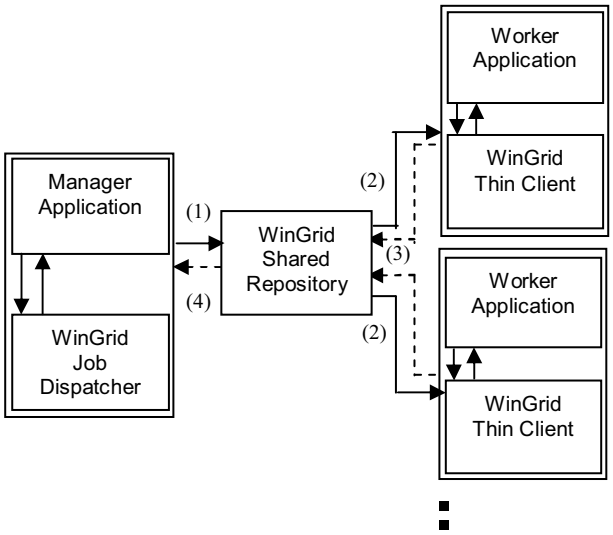


Figure 7: Architecture of web services extension to WinGrid™ (WinGrid-WS)

The constraint that office computers cannot have server software installed means that the WTCs can no longer accept incoming connections directly from the WJD. To overcome this, the *WinGrid Shared Repository* (WSR) has been added. WSR is server software and needs to be installed on only one secure server (which can be administered directly by the IT staff at Ford). Instead of sending jobs directly to a WTC, the WJD will now pass jobs to the WSR (1). To get jobs the WTCs will send requests to the WSR on a regular basis (2). When a WTC has finished with a job, it will send the results to the WSR (3) and request a new job. To get the results, the WJD will also send requests to the WSR on a regular basis (4). Thus, this approach implements the "pull" model which has been discussed earlier in section 3.

The WinGrid-WS architecture has met the approval of the Ford IT support and it is expected to be deployed at Dunton in the near future.

7. Conclusions

This paper has introduced WinGrid™, general purpose desktop grid computing software specifically designed for Microsoft Windows™. We have discussed the use of WinGrid™ in support of FIRST, a specialist simulation application using the CSP Witness™. A Grid-enabled version of FIRST, the FIRST Experimentation Tool, has been produced with minimum technological intervention (indeed the tool was needed to automate experimentation regardless of distributed implementation). We have presented the performance results of the grid-enabled version of FIRST and have discussed the problems of deploying a desktop grid in an enterprise environment and a potential solution. The speed up that this promises over a small Grid at Ford, and the ease with which grid enabling has been accomplished, will give users of FIRST a competitive advantage as results will be delivered significantly faster with minimum technological intervention (i.e. by completely re-implementing FIRST). 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. More information on this project and other sister projects can be found at www.gridalliance.org.uk.

Acknowledgements

The authors would like to thank Tony Waller, Witness™ Product Manager, for granting the use of Witness™ licenses in support of this project.

References

[1] Foster I, Iamnitchi A. 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); 2003 February 21-22; Berkeley, CA, USA; 2003.p.118–128.

[2] Foster I, Kesselman J, Nick J, Tuecke S. The physiology of the grid: An open grid services architecture for distributed systems integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

[3] Bernholdt D, Bharathi S, Brown D, Chanchio K, Chen M, Chervenak A, Cinquini L, Drach B, Foster I, Fox P. The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. Proceedings of the IEEE 2005; 93(3):485-495.

[4] Spencer B, Finholt T, Foster I, Kesselman C, Beldica C, Futrelle J, Gullapalli S, Hubbard P, Liming L, Marcusiu D. Neesgrid: A distributed collaboratory for advanced earthquake engineering experiment and simulation. In: Proceedings of the 13th World Conference on Earthquake Engineering; 2004 August 1-6; Vancouver, BC, Canada; 2004.paper No. 1674.

[5] Jaesun H, Daeyeon P. A lightweight personal grid using a supernode network. In: Proceedings of the 3rd International Conference on Peer-to-Peer Computing; 2003 Sept 1-3; 2003.p. 168-175.

[6] Luther A, Buyya R,Ranjan R,Venugopal S. Alchemi: A. NET-Based Enterprise Grid Computing System. In: Proceedings of the 6th International Conference on Internet Computing (ICOMP'05); 2005 June; Las Vegas, USA; 2005.p.27-30.

[7] Chien A. A, Calder B, Elbert S, Bhatia K. Entropia: architecture and performance of an enterprise desktop grid system. Journal of Parallel and Distributed Computing 2003; 63(5):597-610.

[8]Entropia Inc; 2006.http://www.entropia.com [02/01/2006].

[9] United Devices Inc; 2006. http://www.ud. com/ [22/05/2006].

[10] Platform Computing; 2006. http://www.platform.com/ [22/05/2006].

[11] Mustafee N, Taylor S.J.E. Using a desktop grid to support simulation modelling. In: Proceedings of the 28th Information Technology Interfaces Conference (ITI2006); 2006 June 19-22; Dubrovnik, Croatia; (accepted).

[12] Taylor, S.J.E., Pullen, J.M.. Popescu, G.V, Turner, S.J. Panel on Distributed Simulation and the Grid. In: Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications; 2004 October 21 - 23; Budapest, Hungary; 2004.p.144-149.

[13] Anagnostopoulos D, Nikolaidou M. 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; 2003 October 23 - 25; Delft, The Netherlands; 2003.p.138-146.

[14] Biles W.E, Kleijnen J.P.C. Statistical Methodology for WEB-Based Simulation. In: Proceedings of the 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications; 2003 October 23 - 25; Delft, The Netherlands; 2003.p.147-149.

[15] Yücesan E, Luo Y.C, Chen C.H, Lee I.. Distributed Web-Based Simulation Experiments for Optimization. Simulation Practice and Theory 2001. 9(1):73-90.

[16] Paris J.L, Pierreval H. A Distributed Evolutionary Simulation Optimization Approach for Configuration of Multiproduct Kanban Systems. International Journal of Computer Integrated Manufacturing 2001. 14 (5): 421-430.

[17] Choi S, Baik M, Hwang C, Gil J, Yu H. 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; 2004 August; 2004.p.366-371.

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

[19] Chakravarti A.J, Baumgartner G, Lauria M. Application-specific scheduling for the organic grid. In: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing; 2004 November; 2004.p.146-155.

[20] Anderson D.P. BOINC: a system for public-resource computing and storage. In: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing; 2004 November; 2004.p.4-10.

[21] University of California. Berkeley Open Infrastructure for Network Computing; 2006. http://boinc.berkeley.edu/ [22/05/2006] .

[22] Parabon computation Inc.Compute against cancer; 2006. www.computeagainstcancer.org/ [22/05/2006].

[23] Pande V . Stanford University. Folding@Home; 2006. http://folding. stanford.edu/ [22/05/2006].

[24]Woltman G. GIMPS; 2006. http://www.mersenne.org [22/05/2006].

[25] Litzkow M, Livny M, Mutka M. Condor - A Hunter of Idle Workstations. In: Proceedings of the 8th International Conference of Distributed Computing Systems; 1988 June; 1988.p.104-111.

[26] Calder B, Chien A.A, Wang J, Yang D. The entropia virtual machine for desktop grids. In: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments; 2005 June 11-12; Chicago, IL, USA ; 2005.p.186-196.

[27] Heymann E, Senar M.A, Luque E, Livny M. 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; 2000.p. 214–227.

[28] Taylor S.J.E, Bohli L, Wang X, Turner S.J, Ladbrook J. Investigating Distributed Simulation at the Ford Motor Company. In: Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications; 2005 October 10-12; Montreal, Quebec, Canada; 2005.p.139-147.

[29] Ladbrook J, Janusszczak A. (2001). Ford's Power Train Operations – Changing the Simulation Environment. In: Proceedings of the 33rd Winter Simulation Conference; 2001 December 09 - 12; Arlington, Virginia; 2001.p.863-869.

[30] Gray D.N, Hotchkiss J, LaForge S, Shalit A, Weinberg T. Modern languages and Microsoft's component object model. Communications ACM 1998; 41(5):55-65.

[31] Sun Microsystems Ltd. Java Native Interface (2003). http:/java.sun.com/j2se/1.4.2/ docs/guide/jni [22/05/2006].