INVESTIGATING GRID COMPUTING TECHNOLOGIES FOR USE WITH COMMERCIAL SIMULATION PACKAGES

Navonil Mustafee
The Health Care Research Group
Warwick Business School, University of Warwick
Coventry, CV4 7AL, UK
navonil.mustafee@wbs.ac.uk

Simon J E Taylor
Centre for Applied Simulation Modelling
School of Information Systems, Computing & Maths, Brunel University
Uxbridge, Middlesex, UB8 3PH, UK.
simon.taylor@brunel.ac.uk

ABSTRACT:
As simulation experimentation in industry become more computationally demanding, grid computing can be seen as a promising technology that has the potential to bind together the computational resources needed to quickly execute such simulations. To investigate how this might be possible, this paper reviews the grid technologies that can be used together with commercial-off-the-shelf simulation packages (CSPs) used in industry. The paper identifies two specific forms of grid computing (Public Resource Computing and Enterprise-wide Desktop Grid Computing) and the middleware associated with them (BOINC and Condor) as being suitable for grid-enabling existing CSPs. It further proposes three different CSP-grid integration approaches and identifies one of them to be the most appropriate. It is hoped that this research will encourage simulation practitioners to consider grid computing as a technologically viable means of executing CSP-based experiments faster.

Keywords: Simulation Modelling, Grid Computing, COTS Simulation Package, BOINC, Condor

1. INTRODUCTION AND MOTIVATIONS

Grid computing has the potential to provide users on-demand access to large amounts of computing power, just as power grids provide users with consistent, pervasive, dependable and transparent access to electricity, irrespective of its source (Baker et al., 2002). It has been identified that simulation modelling can potentially benefit from this as computing power can be an issue in the time taken to get results from a simulation (Robinson, 2005; Taylor and Robinson, 2006). Furthermore, development in simulation has been closely allied to the advances in the field of computing (Robinson, 2005) and it is expected that it will continue to rely on the latest advances in computing to support increasingly large and complex simulations (Pidd and Carvalho, 2006). Grid computing is a significant advancement in the field of distributed computing and it is possible that, like previous beneficial developments in computing adopted by simulation users, this technology may provide an opportunity to further improve the use of simulation in industry. This is supported by the observation that the use of grid computing in scientific simulation has certainly proved beneficial. For example, the role it plays to reduce the time taken to produce results (and therefore the opportunity to do more!) is certainly true in disciplines such as particle physics, climatology, astrophysics and medicine, among others. The question is can the same benefits be passed on to the use of simulation modelling as practiced in industry?

Another issue is the relatively low adoption rate of grid computing outside of academic and research domains. At present a major proportion of grid users comprise of researchers (physicists, biologists, climatologists, etc. who are the primary stakeholders of the applications running on the grid) and computer specialists with programming skills (the providers of IT support to the stakeholders). This is not unexpected as the majority of applications using grid computing are research applications. The widespread adoption of grid computing technologies by employees in industry has so far been relatively little. One important reason for this is that although the employees are experts in their own discipline they generally do not have the necessary technical skills that are required to work with present generation grid technologies. A possible means to increase adoption is to incorporate grid support in software applications that require non-trivial amounts of computation power and which are used by the end-users to perform their day-to-day jobs. The commercial simulation packages used in industry are an ideal candidate for such type of integration. Figure 1 summarises the motivations of this research (we shall return to this figure in the conclusions).

This paper therefore investigates the use of grid computing technologies to support simulation modelling by

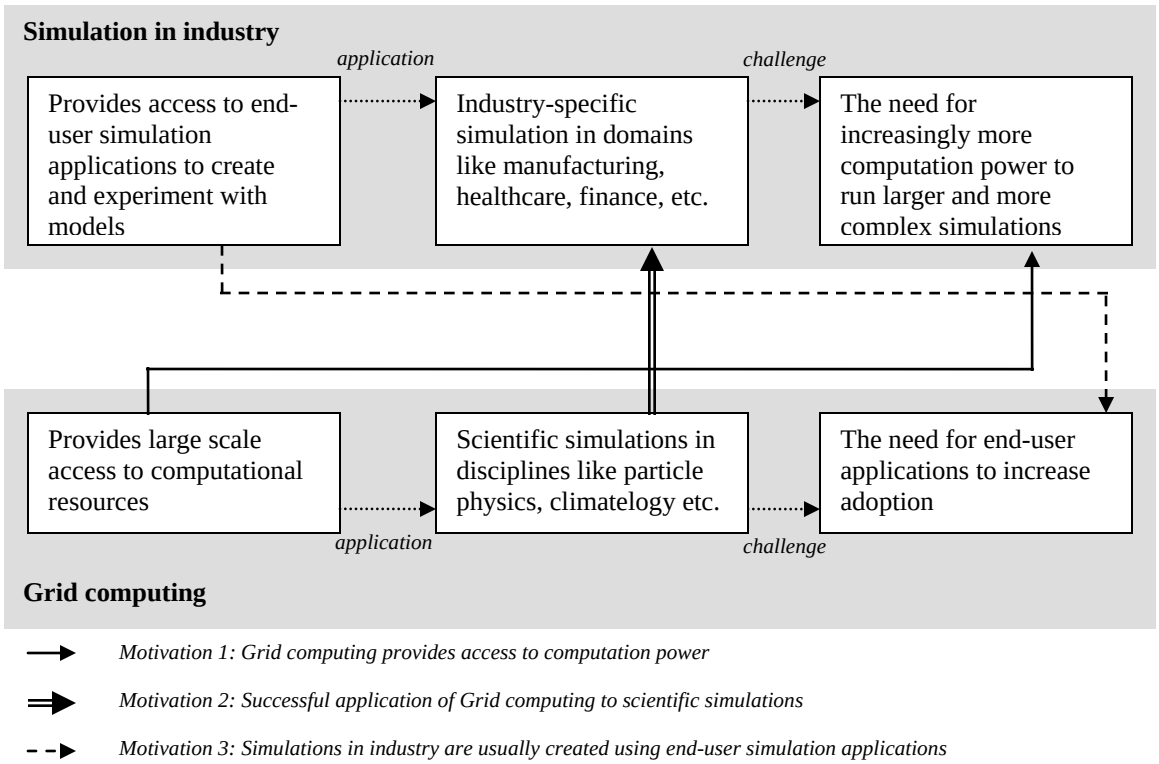

Figure 1: Research motivations

investigating how these technologies can be integrated with *Commercial Off-The-Shelf (COTS) Simulation Packages (CSPs)*. This paper is structured as follows. This section has described the motivations for this research. Section 2 presents an overview of CSPs and discusses operating system support for them. This is followed by a review of grid computing which highlights the opportunities and barriers to using this technology together with the CSPs (section 3). This leads to a discussion of grid technologies for integration with CSPs (section 4 and 5). Three different integration approaches to using CSPs together with desktop grid middleware are presented and one approach is identified as most the appropriate (section 6). Finally, section 7 concludes the paper with a discussion on the challenges facing the wider adopting of this technology for simulation modelling with CSPs.

2. COTS SIMULATION PACKAGES

In the context of simulation practice in industry, discrete event simulation is arguably the most frequently used classical OR technique that is applied across a range of industries like manufacturing, travel, finance and healthcare, among others (Hollocks, 2006). Commercially available discrete-event simulation packages like Simul8™, Witness™ and AnyLogic™ are generally used to model such simulations (Taylor et al., 2005). Monte Carlo simulation is yet another OR technique that is extensively used in application areas like finance and insurance (Herzog and Lord, 2002). Commercially available spreadsheet applications (Microsoft Excel™, Lotus 1-2-3™, etc.), spreadsheet add-ins (Crystal Ball™, @Risk™, etc.) and Monte Carlo simulation packages (Analytica™, Analytics™, etc.) are often used for modelling Monte Carlo simulations in industry (Swain, 2007). We use the term *Commercial Off-The-Shelf (COTS) Simulation Packages (CSPs)* to collectively refer to these.

Swain (2005) has made a comprehensive survey of commercially available simulation tools based on the information provided by vendors in response to a questionnaire requesting product information. This list presently consists of 56 CSPs and features the most well known CSP vendors and their products (Swain, 2007). Of these, a total 45 CSPs have been identified by Mustafee (2007) to be either discrete event or Monte Carlo simulation packages. The 45 CSPs are all supported in the Windows platform, 15.56% (approx.) are supported in UNIX and Linux platforms, and only 13.33% (approx.) are supported under the Apple Macintosh Operating System (Mustafee, 2007). As will be discussed later in this paper, platform support for CSPs is important when considering different grid technologies that can be potentially used with existing CSPs.

3. GRID COMPUTING: DEFINITION, OPPORTUNITIES AND BARRIERS

Grid computing (or Grids) was first defined by Ian Foster and Carl Kesselman in their book *"The Grid: The*

Blueprint for a New Computing Infrastructure" as a hardware and software infrastructure that provides access to high-end computational resources (Foster and Kesselman, 1998). It was further stated that this access should be dependable, consistent, pervasive and inexpensive. This definition of grid computing has since been modified twice by the grid veterans; once by Foster, Kesselman and Tuecke in their paper titled *"Anatomy of the Grid"* (Foster et al., 2001), and again by Foster and Kesselman with the publication of the second edition of their book *"The Grid: The Blueprint for a New Computing Infrastructure"* (Foster and Kesselman, 2004).

The re-definition of the term "grid computing" twice over the period of nearly 6-7 years suggests that this is still an evolving field. However, all the three definitions are consistent in terms of their focus on large-scale computing. Thus, Foster and Kesselman (1998) mention "access to high-end computational resources", Foster et al. (2001) refer to "large-scale resource sharing" and, finally, Foster and Kesselman (2004) highlight "delivery of nontrivial Quality of Service". This focus on large scale computing makes grid computing an enabling technology for *eScience* (Hey and Trefethen, 2002). e-Science is large scale science that is increasingly being carried out through global collaborations, and which requires access to very large data sets and computing resources distributed across a wide geographical area (National e-Science Centre, 2001). Some of the e-Science projects using grid technology include CERN's Large Hadron Collider (LHC) project that is devoted to studying particle physics under conditions well beyond any other previous experiment (Lamanna, 2004); the Network for Earthquake Engineering Simulation (NEES) e-science project that inks earthquake researchers across the U.S. with leading-edge computing resources and research equipment like supercomputers, data storage, networks, visualization displays and sensors (Spencer et al., 2004) and the Earth System Grid (ESG) project where global climate models are used to simulate climate, and experiments are executed on an array of distributed supercomputers (Bernholdt et al., 2005).

Thus it is clear that grid computing presents immense opportunities for e-Science projects that require large scale collaborative use of computing resources. Consequently, the majority of grid users comprise of researchers and computer specialists who are associated with such e-Science projects and have the technical knowledge to work with the present generation grids. This is because the creation of an application that can benefit from grid computing (faster execution speed, linking of geographically separated resources, interoperation of software, etc.) typically requires the installation of complex supporting software and an in-depth knowledge of how this complex supporting software works (Jaesun and Daeyeon, 2003). This software is commonly referred to as grid middleware. A grid middleware is a distributed computing software that integrates network-connected computing resources (computer clusters, data servers, standalone PCs, sensor networks, etc.), that may span multiple administrative domains, with the objective of making the combined resource pool available to user applications for number crunching, remote data access, remote application access, among others. A grid middleware is what makes grid computing achievable.

Globus™, arguably the most recognized grid middleware, is an open source set of services and software libraries which supports grids and grid applications (Foster et al., 2002). Examples of other grid middleware include gLite (Berlich et al., 2005), VDT (Virtual Data Toolkit, 2007), European Data Grid (EDG) middleware (Berlich et al., 2005), OMII middleware (OMII, 2006), LCG-2 (Peris et al., 2005), etc. The middleware mentioned above are all geared towards dedicated, centralized, high performance clusters (such as Beowulf clusters (Beowulf.org, 2007)) and supercomputers running on UNIX and Linux flavour operating systems. Currently, the only exception appears to be Globus™, which allows certain components to be installed in Windows™ computers. These middleware are hence forth referred to as *cluster-based grid middleware*. The operating system support for grid middleware is important when considering the adoption of grid technologies by the end-users at their workplace.

It is common knowledge that most desktop computers run on the different variants of the Windows™ operating system. As such, most of the end-user applications are also supported under the Windows™ platform. Let us take the example of CSPs. The CSPs are typically standalone packages that run on a single desktop PC on the Windows operating system. The users of CSPs tend to be skilled in simulation modelling and not computer science (as many users of grid computing are). Grid support for CSPs must therefore take 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. However, the barrier here is that most of the grid middleware designed for large scale computing are either based on UNIX and Linux operating systems or provide only partial functionality on Windows™ based system.

4. SOLUTION: DESKTOP GRID COMPUTING

The discussion on grid computing, until this point, has shown that grid middleware and applications have traditionally been geared towards large scale projects that use cluster computers running on UNIX and Linux operating systems. *Cluster-based grid computing* 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. Middleware for cluster-based grid computing severely limits the ability to effectively utilize the vast majority of Windows-based resources that are common place in both enterprise and home environments, and therefore development of middleware for desktop-based grid computing is important with the growing industry interest in grids (Luther et al., 2005).

Desktop grid computing or desktop grids addresses the potential of harvesting the idle computing resources of desktop PCs for processing of parallel, multi-parameter applications which consist of a lot of instances of the same computation with its own input parameters (Choi et al., 2004). The desktop grid resources can be part of the same local area network (LAN) or can be geographically dispersed and connected via a global 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. In this paper the use of a desktop grid within the enterprise is termed as *Enterprise-wide Desktop Grid Computing (EDGC)*. Thus, EDGC refers to a grid infrastructure that is confined to an institutional boundary, where the spare processing capacities of an enterprise's desktop PCs are used to support the execution of the enterprise's applications (Chien et al., 2003). User participation in such a grid is not usually voluntary and is governed by enterprise policy. Applications like 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.

Like EDGC, Internet computing seeks to provide resource virtualization through the aggregation of idle CPU cycles of desktop PCs. But unlike EDGC, where the desktop resources are generally connected to the corporate LAN and used to process enterprise applications, Internet computing infrastructure consists of volunteer resources connected over the Internet and is used either for scientific computation or for the execution of applications from which the user can derive some benefit (for example, sharing music files). This research distinguishes between two forms of Internet computing - Public Resource Computing (PRC) and Peer-to-Peer Computing (P2P) - based on whether the underlying desktop grid infrastructure is used for solving scientific problems or for deriving some user benefit respectively. The different forms of grid computing are shown in figure 2. PRC and P2P computing are described next.

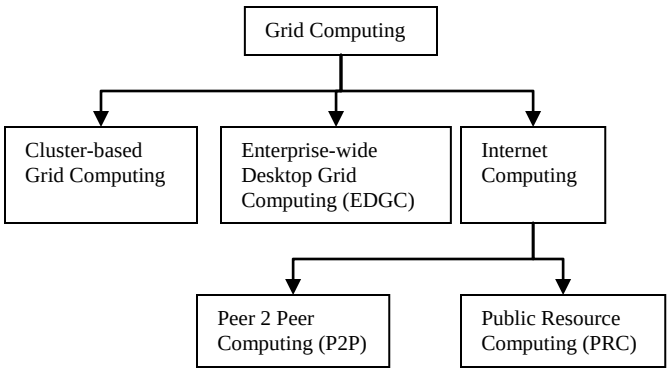

Figure 2: Forms of grid computing (Mustafee, 2007)

PRC refers to the utilization of millions of desktop computers primarily to do scientific research (Anderson, 2004). The participants of PRC projects are volunteers who contribute their PCs to science-oriented projects like SETI@home (Anderson et al., 2002) and Climateprediction.net (Christensen et al., 2005). Berkeley Open Infrastructure for Network Computing (BOINC) (BOINC, 2007b) is arguably the most widely used PRC middleware that enables the project participants to download work units from BOINC servers, process them and upload the results back to the servers. The majority of the PRC middleware is supported on Windows. This is not unexpected as PRC projects depend on volunteer computing resources, and the bulk of these resources presently run on the Windows operating system. The participants of a PRC project are unable to use the underlying desktop grid infrastructure, of which they themselves are part of, to perform their own computations.

P2P computing refers to a non-centralized infrastructure for file sharing over the Internet. P2P networks are created with the resources of the volunteer users (peers) who derive benefit from such networks as it allows them to download files that are shared by other peers. As P2P computing is voluntary, the middleware for such systems should ideally have mechanisms to organize the ad-hoc and dynamic peers in such a way that they can co-operate to provide file sharing services to the P2P community; for example, the P2P middleware should have mechanisms to quickly and efficiently locate files that are distributed among peers (Saroiu et al., 2002). Some of the popular P2P file sharing systems are Gnutella (Sun et al., 2006), KaZaA (Good and Krekelberg, 2003) and in the past, Napster (Giesler and Pohlmann, 2003). They are all supported under the Windows operating system.

5. DESKTOP GRID MIDDLEWARE

This section of this paper presents an overview of two different middleware which has relevance to CSP-based simulation, namely, PRC middleware BOINC and EDGC middleware Condor. P2P computing is not investigated

further because it generally supports only file sharing and as such P2P networks cannot be used to execute programs (like CSPs) on the peer resources. From this point on, the terms "desktop grid computing", "desktop grids", "grid computing" and "grids" will be used synonymously to refer to only PRC and EDGC, unless explicitly stated.

5.1 PRC MIDDLEWARE BOINC

The BOINC system (figure 3, adapted from (Anderson, 2006) and (Perez, 2005)) contains several server-side components, which may execute on separate machines if required. Most of the server side components can only be installed over a UNIX or Linux flavour operating system. The database holds all the metadata associated with the project and lifecycle information for each work unit. A client's command channel operates via the scheduling server, using an XML-based protocol. Results are transferred using HTTP via the data servers. In addition to work units and results, other files may be transferred between server and client, including application executables and any other interim data the application may require during the operation. The database also has a web-based front-end that is used for displaying project information specific to volunteers, for example, how many computers have been contributed by the user, the number of work units processed, etc. On the client side, the *BOINC core client* manages interaction with the server, while optional components (like screensaver and manager) provide graphical control and display elements for the benefit of the user. The core client can be installed in the Windows™ operating system. The BOINC client API provides the interface between the user-created *application client* and the BOINC core client. The API is a set of C++ functions and the application client is compiled with it. All communication between the BOINC core client and the BOINC project servers take place through HTTP on port 80. The BOINC core client can therefore operate behind firewalls and proxies.

Although BOINC was originally designed to support PRC, lately there has been a realization that the same software can be reconfigured to support desktop grid computing (BOINC, 2007a). The widespread availability of desktop PCs in organizations makes the deployment of such an enterprise-wide BOINC infrastructure an even more attractive option. Thus, it may be possible to implement and deploy BOINC-based projects for use exclusively within an enterprise, such that it is geared up to support the execution of the enterprises' applications. The participants of such an enterprise-wide BOINC setup can be the employees of the organization who contribute their work PCs. The participation in such projects may not be voluntary and can be governed by the policy of the organization. The computations being performed by the BOINC clients will be in line with the needs of the enterprise, and unlike PRC where volunteers are encouraged to contribute their resources, only employees and other trusted sources will be allowed to participate in the enterprise-wide BOINC projects. BOINC features that are necessary in the PRC context but may not be required in an enterprise grid (for e.g., user rewards system, anti-cheating measures, mechanisms to deal with client failure or extended network non-connectivity, etc.) can be disabled.

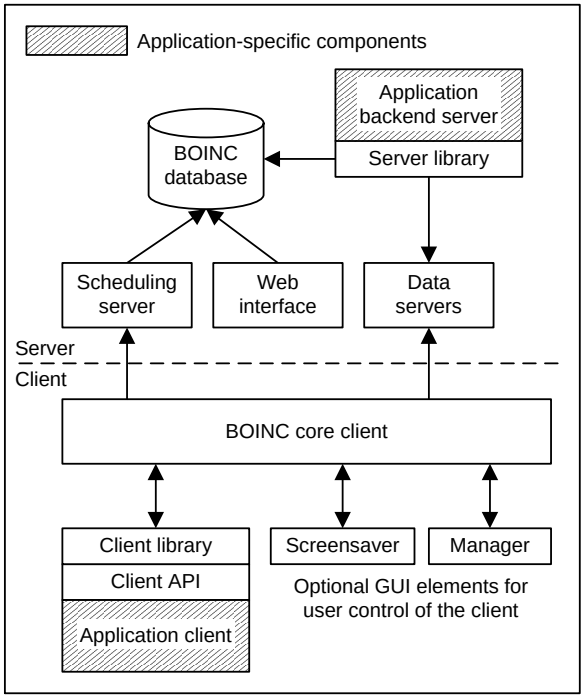

Figure 3: The BOINC system

5.2 EDGC MIDDLEWARE CONDOR

The Condor project was born in the University of Wisconsin-Madison in 1988. Condor is an opportunistic job scheduling system that is designed to maximize the utilization of workstations through identification of idle resources and scheduling background jobs on them (Litzkow et al., 1988). A collection of such workstations is referred to as a Condor pool. When Condor was first introduced in 1988 it was unique because it was arguably the only production system that allowed every user to contribute as much or as little of their resources, and offered an alternative to the dominant centralized processing model of the day (Thain et al., 2004). Two fundamental concepts of Condor middleware, which are also important in our discussions on CSPs, are (a) Condor matchmaking and (b) Condor universe. These are described next.

(a) Condor architecture defines resource providers and resource consumers. The resource providers make their resources available to Condor for the processing of jobs

that originate from the resource consumers. Condor allows both resource consumers and providers to advertise these requirements, conditions and preferences by providing a language called *classified advertisements (ClassAds)* (Thain et al., 2004). The ClassAds are scanned by a Condor *matchmaker agent* (an agent is a Condor software component), running on only one computer in a Condor Pool, to find a match between the requirements advertised by the *resource consumer agents* and the resources advertised by the *resource provider agents.* Once a match has been found by the matchmaker agent, it notifies both the resource consumer and the resource provider agents. Upon receiving this notification, the resource consumer agent claims the resource advertised by the resource provider agent through a claiming protocol. The job is executed by the resource provider agent and the results of the computation are returned back to the resource consumer agent. The matchmaking process is illustrated in figure 4. The figure has been adapted from Basney and Livney (1999).

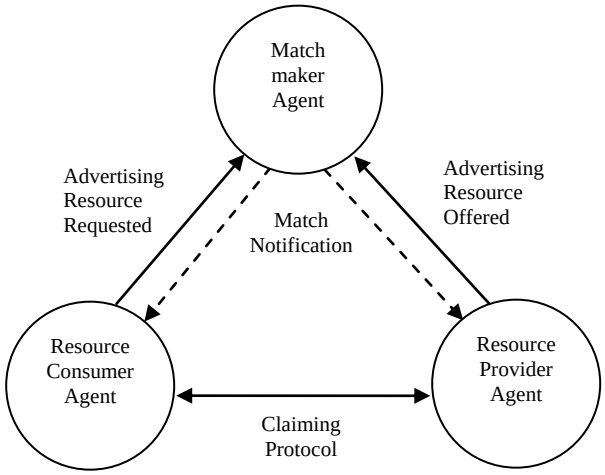

Figure 4: Condor resource management architecture

Thus, in order to execute CSP-based simulations using Condor, PCs acting as resource provider agents will have to be installed with CSPs (Simul8™, Excel™, etc.) and will need to advertise this using ClassAds mechanism. The resource consumer agents will also be required to advertise their requirement (for example, 10 PCs required) with the condition that the resource providers will have the appropriate CSPs installed on them.

(b) *Condor universe* is an execution environment for jobs that are submitted by the users. Depending upon the type of job to be executed and its requirements, the user needs to select from among the following Condor universes (Condor Version 6.9.1 Manual, 2007b): standard universe, vanilla universe, Java universe, PVM universe, parallel universe, grid universe, scheduler universe, local universe. Of these, Java universe, which supports the execution of java programs using the Java Virtual Machine (JVM) execution environment, is the most appropriate for executing CSP-based simulations over Condor (Mustafee, 2007).

6. CSP-GRID INTEGRATION APPROACHES

For desktop grids to support CSP-based simulation, it should take into account that the CSP vendors and the grid middleware developers may be unwilling to make any source code changes to their software. Thus, any technological solution proposed should be able to integrate "unmodified" grid middleware with "unmodified" CSPs. Three possible approaches for using desktop grids with CSPs are discussed next. These are referred to as the *CSP-middleware integration approach*, the *CSP-runtime installation approach* and the *CSP-preinstalled approach*.

6.1 CSP-GRID MW. INTEGRATION APPROACH

One possible way of using desktop grid middleware together with CSPs is to "bundle" the latter along with the former. When a desktop grid middleware is installed on a PC, the CSP is also installed on it. In an enterprise-wide desktop grid the jobs from other users (guest processes) may run alongside the programs being executed by the resource owner (host processes). However, the guest processes are usually run in a "sandbox" that is implemented by the middleware. This provides a logically separate and secure execution environment for both the host and guest processes. In Entropia DCGrid for example, the sandbox mechanism is called the *Entropia Virtual Machine (EVM)* and it wraps interpreters like cmd.exe, Perl and Java Virtual Machine (JVM) to prevent unauthorized access to a computer (Calder et al., 2005). Thus, it might be possible to include a CSP installation inside the EVM and offer it as part of an Entropia installation. The problem with this approach is that it will require changes to the enterprise desktop grid middleware as a CSP will have to be integrated with it. Furthermore, an enterprise desktop grid is a general purpose distributed computing environment that allows the execution of various user applications (not limited to simulation alone). Although the integration of interpreters like JVM can be justified because of the wide prevalence of Java applications, it is arguably more difficult to explain the inclusion of a CSP (but which CSP? there are at least 45 of them), unless a customized desktop grid middleware distribution is created for meeting simulation requirements of a specific organization. This approach is not considered feasible for reasons outlined earlier (section 6).

6.2 CSP-RUNTIME INSTALLATION APPROACH

The second approach involves the installation of a CSP package at runtime, i.e. just before the simulation experiment is conducted. In this case the CSP itself is transferred to the desktop grid nodes, along with the data files associated with the simulation and the trigger code

(executable code which starts the CSP-based simulation on a grid node). This approach may not be feasible for a number of reasons. (1) the size of CSPs frequently exceed 100s of MBs and it may not be feasible to transfer such large amounts of data to multiple clients over the network, (2) the CSP will first need to be installed on the desktop grid node before the simulation can start, (3) such an installation is normally an interactive process and requires human intervention, (4) an installation normally requires administrative privileges on the client computers, (5) transferring CSPs may lead to a violation of the software licence agreement that may be in place between the CSP vendor and the organization (if the number of desktop grid nodes executing simulations exceed the number of licences purchased).

6.3 CSP-PREINSTALLED APPROACH

The third CSP-grid integration approach is to install the CSP in the desktop grid resource, just like any other application is installed on a PC. The drawback with this approach is that the sandbox security mechanism implemented by most enterprise desktop grids may have to be forfeited. However, as simulations are created by trusted employees running trusted software within the bounds of a fire-walled 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).

Of the three CSP-grid integration approaches discussed in this section, the CSP-preinstalled approach is considered the most appropriate because (1) it does not require any modification to the CSPs – thus, CSPs that expose package functionality can be grid-enabled, (2) it does not require any modification to the grid middleware – thus, existing Windows™-based grid middleware like BOINC and Condor can be used, and (3) CSPs that are usually installed on the PCs of the simulation practitioners can be utilized for running simulation experiments from other users in the background.

The procedure to execute CSP-based simulation experiments over desktop grids following the CSP-preinstalled approach is as follows (see figure 5):

1. The simulation user writes an executable "trigger" code in C++, Java, Visual Basic (VB), etc. that accesses the CSP functionality through exposed interfaces. The trigger code should generally invoke the CSP, load the model file, transfer experiment parameters into the model, execute the model, etc. Mustafee (2007) provides a list of CSPs that expose package functionality using well-defined interfaces.
2. The simulation user makes available the data files associated with the simulation (simulation model files, experiment parameter files, etc.) and the executable file containing the trigger code to the desktop grid nodes where the experiment will be executed. Two possible ways of accomplishing this are (1) by providing a shared grid access to a network drive, or (2) by transferring the required files using the desktop grid middleware.
3. The desktop grid middleware invokes the executable trigger code on a remote desktop node. The simulation starts and results are saved in a file. The user retrieves the results by (1) accessing them from the shared network drive, or (2) the result files are transferred back to the user through the grid middleware.

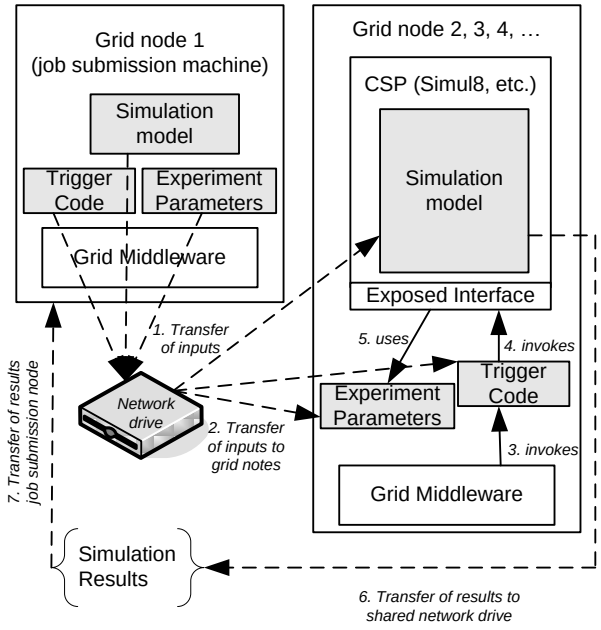

Figure 5: Executing CSP-based simulation over grid resources using CSP-preinstalled approach

The reader is referred to Mustafee (2007) for case studies associated with using CSPs together with BOINC and Condor and the CSP-grid integration technology that is used for this purpose.

7. DISCUSSIONS AND CONCLUSIONS

Through a review of literature this paper has identified two forms of grid computing that can be used to grid-enable existing CSPs. These are Public-Resource Computing (PRC) in an enterprise context and Enterprise Desktop Grid Computing (EDGC). The use of PRC and EDGC forms of grid computing for CSP-based simulation in industry can not only speed up simulation experimentation, but it can also maximize the utilization of hardware and software resources (PCs, network infrastructure, CSPs) within an organization. The latter is achieved through making use of

under utilized desktop computers and the software installed on them.

This paper has then discussed two specific grid computing middleware, namely PRC middleware BOINC and EDGC middleware Condor. Both these middleware are available for download free of charge, include installation manuals and user guides, and are supported by user forums and training programs (for example, *Condor Week* is an annual training program conducted by the University of Wisconsin, Madison). This presents an opportunity for the simulation user to experiment with these middleware with an objective to run simulation experiments faster.

This research has shown that it is technologically feasible for grid computing to make available computational resources for running CSP-based experiments (figure 1: motivation one) and thus industry can potentially benefit from it (figure 1: motivation 2). It has also been shown that end-user tools like CSPs could be successfully integrated with grid middleware using low intervention solutions (figure 1: motivation 3).

However, the CSP-grid integration solution proposed by Mustafee (2007) requires some knowledge of Java and Visual Basic programming. Furthermore, the end-users will also need to know the middleware-specific mechanisms to create jobs (in the context of CSP-based simulation, a job can be thought of as one simulation experiment that is to be executed over a grid resource), submit jobs, retrieve results, etc. Some of this knowledge could be acquired through self-study and imparted through training. However, for the wider adoption of grid technology for CSP-based simulation, it may be necessary to develop higher-level tools that would hide the complexity of the CSP-grid integration technology and middleware specific mechanisms, and provide end-users with easy to use graphical interfaces through which they could possibly integrate CSPs with grid middleware.

REFERENCES

Anderson, D. P. (2004). BOINC: a system for public-resource computing and storage. In Proceedings of the 5th International Workshop on Grid Computing, pp.4-10. IEEE Computer Society, Washington, DC, USA.

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M. and Werthimer, D. (2002). SETI@home: an experiment in public-resource computing. Communications of the ACM, 45(11): 56-61.

Anderson, D. P., Christensen, C. and Allen, B. (2006). Designing a runtime system for volunteer computing. In Proceedings of the 2006 International Conference on High Performance Computing, Networking, Storage, and Analysis (Supercomputing, 2006). Article No. 126. ACM Press, New York, NY, USA.

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

Basney, J. and Livny, M. (1999). Deploying a high throughput computing cluster. In Buyya, R. (ed.), High Performance Cluster Computing, Volume 1 (chapter 5). NJ, USA: Prentice Hall PTR.

Beowulf.org. (2007). What makes a cluster a Beowulf? Website http://www.beowulf.org/overview/index.html. Last accessed 9th February 2007.

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

Bernholdt, D., Bharathi, S., Brown, D., et al. (2005). The earth system grid: supporting the next generation of climate modelling research. Proceedings of the IEEE, 93(3): 485-495.

BOINC. (2007a). Desktop grid computing with BOINC. Website http://boinc.berkeley.edu/dg.php. Last accessed 17th February 2007.

BOINC. (2007b). Overview of BOINC. Website http://boinc.berkeley.edu/intro.php. Last accessed 17th February 2007.

Calder, B., Chien, A., Wang, J. and Yang, D. (2005). The entropia virtual machine for desktop grids. In Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, pp.186-196. ACM Press, New York, NY, USA.

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

Chien, 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, pp. 366-371. IEEE Computer Society, Washington, DC, USA.

Christensen, C., Aina, T. and Stainforth, D. (2005). The challenge of volunteer computing with lengthy climate model simulations. In Proceedings of the First International Conference on e-Science and Grid Computing (E-SCIENCE '05), pp.8-15. IEEE Computer Society, Washington, DC, USA.

Condor Version 6.9.1 Manual. (2007b). Road-map for running jobs, Condor 6.9.2 manual. Website http://www.cs.wisc.edu/condor/manual/v6.9/2_4Road_map_Running.html. Last accessed 27th February 2007.

Digipede Technologies. (2006). The digipede network. http://www.digipede.net/products/digipede-network.html. Last accessed 13th February 2007.

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

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

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

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

Giesler, M. and Pohlmann, M. (2003). The anthropology of file sharing: consuming Napster as a gift. Advances in Consumer Research, volume 30, pp 273-279. Available online www.acrwebsite.org/volumes/display.asp?id=8790. Last accessed 4th March 2007.

Good, N. S. and Krekelberg, A. (2003). Usability and privacy: a study of kazaa P2P filesharing. In Proceedings of the SIGCHI conference on human factors in computing systems, pp. 137-144. ACM Press, New York, NY, USA.

Herzog, T. N. and Lord, G. (2002). Applications of monte carlo methods to finance and insurance. Winstead, Conn: ACTEX Publications. Available online http://books.google.com. Last accessed 11th March 2007.

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

Hollocks, B. W. (2006). Forty years of discrete-event simulation - a personal reflection. Journal of the Operational Research Society, 57(12): 1383-1399.

Jaesun H and Daeyeon P. 92003). A lightweight personal grid using a supernode network. In Proceedings of the 3rd International Conference on Peer-to-Peer Computing, pp. 168-175.

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

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

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

Luther, A., Buyya, R., Ranjan, R. and Venugopal, S. (2005). Alchemi: a .NET-based enterprise grid computing system. In Proceedings of the 6th International Conference on Internet Computing (ICOMP'05), pp. 269-278. CSREA Press, USA. Online http://gridbus.csse.unimelb.edu.au/. Last accessed 4th April 2007.

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

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

National e-Science Centre. (2001). Defining e-Science. http://www.nesc.ac.uk/nesc/define.html. Last accessed 12th February 2007.

OMII. (2006). User guide for the OMII middleware, release 3.2.0. Accessible online http://www.omii.ac.uk/docs/3.2.0/user_guide/omii_user_guide.htm. Last accessed 15th March 2007.

Perez, J. A. L. (2005). BOINC architecture and basic principles, CERN presentation. Website https://twiki.cern.ch/twiki/pub/LHCAtHome/LinksAndDocs/boincciemat06.pdf. Last accessed 17th February 2007.

Peris, A. D., Lorenzo, P. M., Donno, F., Sciaba, A., Campana, S. and Santinelli, R. (2005). LCG-2 user guide, manuals series. Document identifier: CERN-LCG-GDEIS-454439. Online https://edms.cern.ch/file/454439//LCG-2-UserGuide.pdf. Last accessed 15th March 2007.

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

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

Saroiu, S., Gummadi, P. K. and Gribble, S. D. (2002). A measurement study of peer-to-peer file sharing systems. In Proceedings of the Multimedia Computing and Networking. Available online http://www.cs.toronto.edu/~stefan/publications/mmcn/2002/mmcn.html. Last accessed 18th March 2007.

Spencer, B., Finholt, T. A., Foster, I., et al. (2004). NEESgrid: A distributed collaboratory for advanced earthquake engineering experiment and simulation. In Proceedings of the 13th World Conference on Earthquake Engineering, paper No. 1674. Available online http://www.globus.org. Last accessed 4th April 2007.

Sun, Q., Daswani, N. and Garcia-Molina, H. (2006). Maximizing remote work in flooding based peer-to-peer systems. Computer Networks, 50(10): 1583-1598.

Swain J. J. (2005). Gaming reality: biennial survey of discrete-event simulation software tools. OR/MS Today (December 2005). Institute for Operations Research and the Management Sciences (INFORMS), USA. Available online http://www.lionhrtpub.com/orms/orms-12-05/frsurvey.html. Last accessed 4th April 2007.

Swain J. J. (2007). INFORMS simulation software survey. OR/MS Today. Institute for Operations Research and the Management Sciences (INFORMS), USA. Available http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation.html. Last accessed 4th April 2007.

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

Taylor, S. J. E, Turner, S. J., Mustafee, N., Ahlander, H. and Ayani, R. (2005). COTS distributed simulation: a comparison of CMB and HLA interoperability approaches to type I interoperability reference model problems. Simulation, 81(1): 33–43.

Thain, D., Tannenbaum, T. and Livny, M. (2004). Distributed computing in practice: the Condor experience. Concurrency and Computation: Practice and Experience, 17(2–4): 323–356.

United Devices. (2007). Grid MP: The technology for enterprise application virtualization. Web http://www.ud.com/products/gridmp.php. Last accessed 18th March 2007.

Virtual Data Toolkit. (2007). What is in VDT 1.6.1? Website http://vdt.cs.wisc.edu/releases/1.6.1/contents.html. Last accessed 16th March 2007.

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

AUTHOR BIOGRAPHIES

NAVONIL MUSTAFEE is a research fellow in the Health Care Research Group at Warwick Business School. His research interests are in parallel and distributed simulation, health care simulation and grid computing. He completed his PhD in Information Systems and Computing and his MSc in Distributed Information Systems from Brunel University in 2007 and 2003 respectively. He is a member of the drafting group of the COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) under the Simulation Interoperability Standards Organization. His email address is navonil.mustafee@wbs.ac.uk.

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