Grid Technology for Maximizing Collaborative Decision Management and Support:

Advancing Effective Virtual Organizations

Nik Bessis
University of Bedfordshire, UK

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE
Hershey · New York

Director of Editorial Content:	Kristin Klinger
Senior Managing Editor:	Jamie Snavely
Managing Editor:	Jeff Ash
Assistant Managing Editor:	Carole Coulson
Typesetter:	Sean Woznicki
Cover Design:	Lisa Tosheff
Printed at:	Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com/reference

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

Library of Congress Cataloging-in-Publication Data

Grid technology for maximizing collaborative decision management and support : advancing effective virtual organizations / Nik Bessis, editor.
p. cm.
Includes bibliographical references and index.

Summary: "This book presents research on building network of excellence by effectively and efficiently managing ICT-related resources using Grid technology"--Provided by publisher.

ISBN 978-1-60566-364-7 (hbk.) -- ISBN 978-1-60566-365-4 (ebook)
1. Information technology--Management. 2. Computational grids (Computer systems) 3. Shared virtual environments. I. Bessis, Nik, 1967-
T58.64.G75 2009
658.4'038--dc22
2008042439

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter VI
Leveraging Simulation Practice in Industry through Use of Desktop Grid Middleware

Navonil Mustafee
University of Warwick, UK

Simon J. E. Taylor
Brunel University, UK

ABSTRACT

This chapter focuses on the collaborative use of computing resources to support decision making in industry. Through the use of middleware for desktop grid computing, the idle CPU cycles available on existing computing resources can be harvested and used for speeding-up the execution of applications that have "non-trivial" processing requirements. This chapter focuses on the desktop grid middleware BOINC and Condor, and discusses the integration of commercial simulation software together with free-to-download grid middleware so as to offer competitive advantage to organizations that opt for this technology. It is expected that the low-intervention integration approach presented in this chapter (meaning no changes to source code required) will appeal to both simulation practitioners (as simulations can be executed faster, which in turn would mean that more replications and optimization are possible in the same amount of time) and management (as it can potentially increase the return on investment on existing resources).

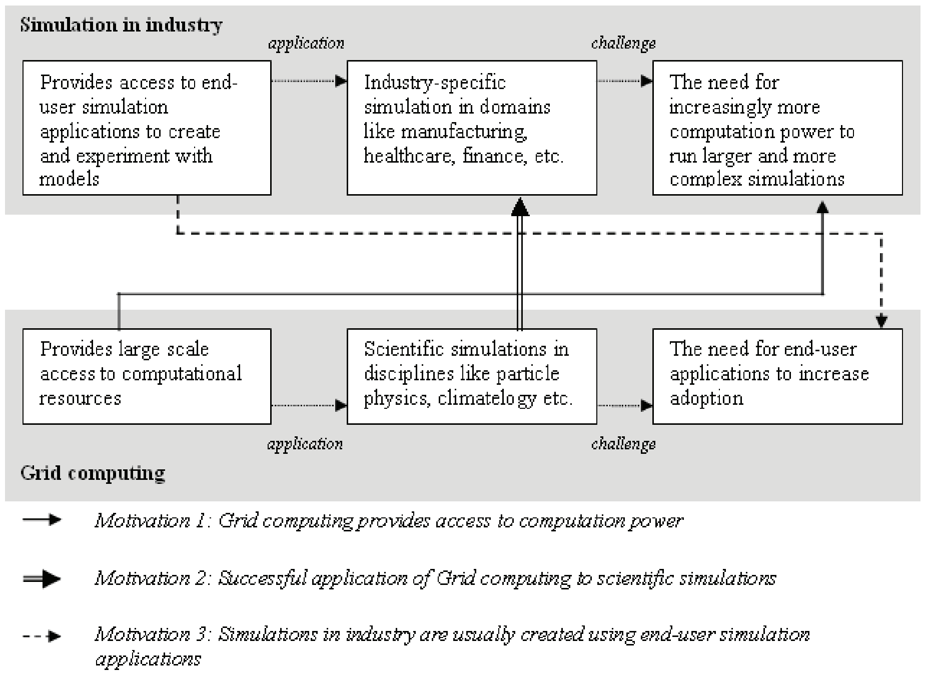
INTRODUCTION AND MOTIVATION

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). Simulation in industry can potentially benefit from this as computing power can be an issue in the time taken to get results from a simulation (Robinson, 2005a). This is further supported by the observation that the use of grid computing in scientific simulation has certainly proved beneficial in disciplines such as particle physics, climatology, astrophysics and medicine, among others. Thus, our first motivation is to inform simulation users in industry as to how the practice of simulation can benefit from grid computing.

Another motivation is the 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 stakeholder 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 adoption of grid computing technologies by employees in industry has so far been relatively modest. One important reason for this is, 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, off-the-shelf (COTS) Simulation Packages (CSPs)* used in industry to model simulations are an ideal candidate for such type of integration. This chapter, thus, focuses on leveraging the practice of CSP-based simulation in industry through use of grid computing. Figure 1 shows the motivations of this research.

The remainder of this chapter is organised as follows. The second section gives an overview of the practice of simulation in industry and the CSPs used to model such simulations. The following two sections are devoted to grid computing and

Figure 1. Chapter motivations


Simulation in industry
application
challenge
Provides access to end-user simulation applications to create and experiment with models
Industry-specific simulation in domains like manufacturing, healthcare, finance, etc.
The need for increasingly more computation power to run larger and more complex simulations
Provides large scale access to computational resources
Scientific simulations in disciplines like particle physics, climatelogy etc.
The need for end-user applications to increase adoption
application
challenge
Grid computing
Motivation 1: Grid computing provides access to computation power
Motivation 2: Successful application of Grid computing to scientific simulations
Motivation 3: Simulations in industry are usually created using end-user simulation applications

desktop grid computing respectively. The CSP-grid integration approaches are proposed in the subsequent section, followed by two CSP-grid integration case studies. The last section presents a summary of this research and brings the chapter to a close.

SIMULATION PRACTICE IN INDUSTRY

Defining Computer Simulation

A computer simulation uses the power of computers to conduct experiments with models that represent systems of interest (Pidd, 2004). Experimenting with the computer model enables us to know more about the system under scrutiny and to evaluate various strategies for the operation of the system (Shannon, 1998). Computer simulations are generally used for experimentation as they are cheaper than building (and discarding) real systems; they assist in the identification of problems in the underlying system and allow testing of different scenarios in an attempt to resolve them; allow faster than real-time experimentation; provide a means to depict the behaviour of systems under development; involve lower costs compared to experimenting with real systems; facilitate the replication of experiments; and provide a safe environment for studying dangerous situations like combat scenarios, natural disasters and evacuation strategies (Brooks et al., 2001; Pidd, 2004).

Application of Simulation in Industry

Various simulation techniques are applied to a wide range of application domains for a variety of purposes. For example, *System Dynamics (SD)* is used in industry for strategy development and supply chain management; *Discrete Event Simulation (DES)* is used to estimate availability of weapons systems in the military, in the manufacturing industry DES is used for inventory management, scheduling and optimization; *Parallel and Distributed Simulation (PADS)* is used in the military for conducting large-scale simulation-based training; *Agent-Based Simulation (ABS)* has been used in defence to examine dynamic teaming and task allocation problems, in industry the possible applications of ABS include supply chain management, organizational design and process improvement (Eldabi et al., 2008). In healthcare, *Monte-Carlo Simulation* (MCS) has been used to evaluate the cost-effectiveness of competing technologies, SD has assisted in designing healthcare policies, ABS has been used to study problems such as the spread of epidemics, DES has been used to forecast the impact of changes in patient flow, to examine resource needs, to manage patient scheduling and admissions, etc.

This chapter focuses on DES and MCS performed for the purposes of (1) optimization of resources in the manufacturing industry and (2) risk analysis in the banking, insurance and finance sector. Simulations associated with both optimization and risk analysis can usually benefit from more computing power, since optimization generally involves conducting several sets of DES experiments with varying resource parameters and MCS involves executing thousands of Monte Carlo iterations. Although manufacturing and finance remain the focus of this chapter, the reader should note that the case study exemplars presented in this chapter could easily be generalized to other areas of use. The grid computing technologies that are subsequently proposed are ideally suited to SMEs who have a requirement for additional computing power to run optimization and credit risk simulations, but may not be willing to invest in additional computer hardware (like Beowulf clusters, computers with multi-core processors, etc.) or commercial “black-box” software (like Digipede Network, MathLab Parallel Computing Toolbox, etc.), but would instead be interested in increasing their Return on Investment (ROI) on existing hardware, software and technical resources. The proposed grid solutions are suitable for SMEs because they can be downloaded free of cost; they present an opportunity to harness idle CPU cycles from computing resources that already exist in an organization; they have the potential to increase the ROI on simulation software through their integration with the grid solutions; and finally, these grid solutions could increase the utilization of technical resources like IT staff and subscription to specialized online IT helpdesks, references and forums.

DES and MCS

DES and MCS are two simulation techniques that are widely used in industry. In DES the behaviour of a model, and hence the system state, changes at an instant of time (Brooks et al., 2001). DES is arguably the most frequently used classical Operational Research (OR) technique that is applied across a range of industries like manufacturing, travel, healthcare, among others (Hollocks, 2006).

MCS is yet another OR technique that is extensively used in application areas like finance and insurance (Herzog & Lord, 2002). MCS uses a sequence of random numbers according to probabilities assumed to be associated with a source of uncertainty, for example, stock prices, interest rates, exchange rates or commodity prices (Chance, 2004).

In the context of simulation practice in industry, although programming languages may be used to build simulations in certain circumstances, models are generally created using commercially available simulation packages (Robinson, 2005b). In this chapter the term *Commercial Off-The-Shelf (COTS) Simulation Packages (CSPs)* are used to refer to software used for modelling both DES and MCS. CSPs are described next.

COTS Simulation Packages

Visual interactive modelling systems usually refer to DES software that enable users to create models in a graphical environment through an interactive "click-and-drag" selection of pre-defined simulation objects (entry points, queues, workstations, resources, etc.) and linking them together to represent the underlying logical interactions between the entities they represent (Pidd, 2004). Examples of such software include DES packages like Witness (Lanner group) and Simul8 (Simul8 corporation). Similarly, MCS may be modelled in a visual environment using spreadsheet software like Excel (Microsoft), Lotus 1-2-3 (IBM, formerly Lotus Software); spreadsheet add-ins, for example @Risk (Palisade Corporation), Crystal Ball (Decisioneering); or through MC-specific simulation packages such as Analytica (Lumina Decision Systems) and Analytics (SunGard).

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 tools and features the most well known CSP vendors and their products (Swain, 2007). All the 45 CSPs (12 MCS CSPs and 33 DES CSPs) that have been identified from Swain's survey are supported in the Windows platform, 15.56% (approx.) are supported in UNIX and Linux platforms, and only 13.33% (approx.) are supported on Apple Macintosh Operating System (Mustafee, 2007). As will be discussed later in this chapter, platform support for CSPs is important when considering different grid technologies that can be potentially be used with existing CSPs. A discussion on grid computing is presented in the next section.

GRID COMPUTING

Defining Grid Computing

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 & Kesselman, 1998). This definition has since then been modified twice by the grid veterans over a period of nearly 5 years. 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 QoS". This definition of grid computing, referred to in this chapter as *cluster-based grid computing*, is generally geared towards dedicated high performance clusters and super computers running on UNIX and Linux flavour operating systems. However, as will be discussed in the subsequent section on desktop grid computing, 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. The following two sub-sections pertain only to *cluster-based grid computing.*

Grid Middleware

A grid middleware is a distributed computing software that integrates network-connected computing resources (computer clusters, data servers, standalone PCs, sensor networks, etc.), that may span multiple administrative domains, with the objective of making the combined resource pool available to user applications for number crunching, remote data access, remote application access, among others (Mustafee & Taylor, 2008). A grid middleware is what makes grid computing possible. Table 1 presents an overview of grid middleware that are commonly installed on distributed computing resources to create an underlying infrastructure for grid computing. The operating system support for each middleware is also highlighted.

Table 1. Examples of middleware for grid computing

Middleware	Description	Operating System support
Globus (GT 4)	Globus middleware is an open architecture and an open source set of services and software libraries which supports grids and grid applications (Foster et al., 2002).	UNIX, Linux and Windows. However, the WSRF- GRAM; the non-web services implementations for GRAM, MyProxy, etc. can only be run on UNIX and Linux platforms (Globus Alliance, 2005).
Condor	Condor is a job scheduling system that maximizes the utilization of collections of networked PCs through identification of idle resources and scheduling background user jobs on them (Litzkow et al., 1988).	UNIX, Linux and Windows platforms. However, several Condor execution environments like standard universe, PVM universe, etc. are not supported on Windows (Condor Version 6.9.1 Manual, 2007).
Virtual Data Toolkit (VDT)	It is a combined package of various grid middleware components, including Globus and Condor, and other utilities. The goal of VDT is to provide users with a middleware that is thoroughly tested, simple to install and maintain, and easy to use.	VDT (version 1.6.1) supports only Linux-based platforms like *Debian Linux*, *Fedora Core Linux*, *RedHat Enterprise Linux*, *Rocks Linux*, *Scientific Linux* and *SUSE Linux* (Virtual Data Toolkit, 2007).
GLite	gLite uses components developed from several other grid projects like Globus and Condor. gLite is primarily being developed for the LHC Computation Grid (LCG) and the EGEE grids (see section 3.3).	GLite-3 middleware is presently supported only on the *Scientific Linux* operating system (Burke et al., 2007).

Production Grids

Production grids can be defined as grid computing infrastructures that have transitioned from being "research and development" test beds to being fully-functional grid environments, offering users round-the-clock availability at sustained throughput levels. Production grids are usually supported by a team that is responsible for the day-to-day maintenance of the grid, solving technical problems associated with the grid, helping users through help-desk support, creating user documents, conducting training courses for knowledge dissemination purposes, among others. Table 2 lists some of the large production grids and the grid middleware running on them.

As can be seen from Table 2, most of these production grids have a resource base spanning multiple virtual organizations (VOs). These production grids are mainly being used for e-Science projects. There are very few examples of multiple VO-based grid computing being used in industry. However, it is also true that grid computing middleware like Globus is gradually being introduced within enterprises for processing enterprise-related applications. In this scheme the organizations seek to leverage their existing computing resources using grid middleware. Collaborations, if any, are limited to intra-organizational resource sharing and problem solving. Organizations that use grid computing middleware for their day-to-day operations or integrate these middleware within their own applications include SAP (Foster, 2005), GlobeXplorer (Gentzsch, 2004) and Planet Earth (Levine & Wirt, 2004).

It has to be said here that there is little agreement over what the term grid computing actually means and there is not one, all-accepted, defini-

Table 2. Examples of production grids

Grid Name	Purpose	Infrastructure	Grid MW	Reference
LCG (LHC Computing Grid), Europe	The purpose of LCG is to provide computation and storage resources for four LHC particle physics experiments at CERN, Geneva.	At present the LCG grid spans over 200 sites around the world and has access to more than 30,000 CPUs and 20 PB of data storage capacity.	LCG-2 /gLite	(Lamanna, 2004), and (Burke et al., 2007)
EGEE (Enabling Grids for E-sciencE) Grid, Europe	EGEE Grid infrastructure is ideal for any scientific research.	The EGEE project involves over 90 partner institutions across Europe, Asia and the United States and provides access to over 20,000 CPU and 5 Petabytes of storage.	LCG-2 /gLite	(EGEE, 2007)
NGS (National Grid Service), UK	Production use of computational and data grid resources in all branches of academic research.	NGS provides access to over 2,000 processors and over 36 TB storage capacities. These resources are provided by the Universities of Manchester, Leeds, Oxford and RAL, among others.	Globus	(Yang et al., 2005)
TeraGrid, USA	Research in genomics, earthquake studies, cosmology, climate and atmospheric simulations, biology, etc.	As of 2003, the TeraGrid infrastructure consists of the NCSA, SDSC, ANL, among others.	Globus	(Reed, 2003)

tion of grid computing. For example, Baker et al. (2002, p. 1437) mention that the "cooperative use of geographically distributed resources unified to act as a single powerful computer" is known by several names such as "meta-computing, scalable computing, global computing, Internet computing, and more recently peer-to-peer or Grid computing" and Luther et al. (2005) refer to enterprise desktop grid computing, public distributed computing and peer-to-peer computing as different names for Internet computing. However, as will be seen from the discussion presented in the next sub-section, grid computing, enterprise desktop grid computing and Internet / peer-to-peer / public resource computing generally have a different set of objectives that determine the design architecture of their underlying middleware technologies.

Different Forms of Grid Computing

The discussion on grid computing, until this point, has shown that grid infrastructures and middleware applications have traditionally been geared towards dedicated, centralized, high performance clusters and super-computers running on UNIX and Linux operating systems. This form of grid computing will henceforth be referred to as *cluster-based grid computing*. It 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. 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.

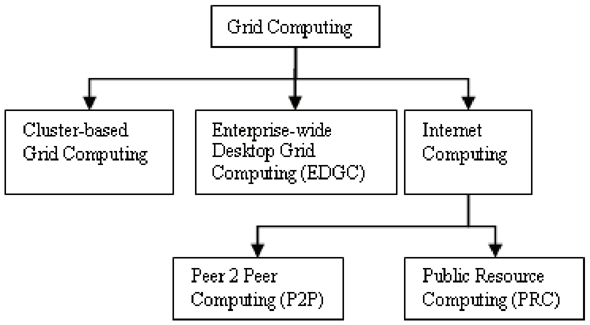
In this chapter the use of a desktop grid within the enterprise is termed as *Enterprise-wide Desktop Grid Computing (EDGC)*. Thus, EDGC refers to a grid infrastructure that is confined to an institutional boundary, where the spare processing capacities of an enterprise's desktop PCs are used to support 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, Entropia DCGrid, and Digipede Network 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.

Grid Middleware and CSPs

Discussions presented earlier in the chapter have highlighted that all CSPs are supported on the Windows platform, 15.56% on both UNIX and Linux operating systems and only 13.33% CSPs are supported on Macintosh. This shows the prevalence of Windows-based CSPs in industry. It is therefore arguable that for this research to be widely relevant to the practice of CSP-based simulation in industry, it should, first and foremost, focus on Windows-based grid computing solutions. Discussion on cluster-based UNIX and Linux grid solutions for CSP-based simulation modelling is thus outside the scope of this chapter.

Figure 2. Forms of grid computing


Grid Computing
Cluster-based Grid Computing
Enterprise-wide Desktop Grid Computing (EDGC)
Internet Computing
Peer 2 Peer Computing (P2P)
Public Resource Computing (PRC)

P2P computing is also 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. Thus, the next section focuses only on PRC and EDGC forms of grid computing.

DESKTOP GRID COMPUTING

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 e-Science research and for enterprise application processing. Both BOINC and Condor are cycle stealing systems that can run on non-dedicated Windows PCs.

The rationale for choosing BOINC as a representative form of PRC middleware is as follows:

- It is arguably the most popular PRC middleware. "BOINC is currently used by about 20 projects, to which over 600,000 volunteers and 1,000,000 computers supply 350 TeraFLOPS of processing power" (Anderson et al., 2006, p. 33).
- It is presently the only PRC middleware that allows users to create their own projects.
- It is available free of cost.

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

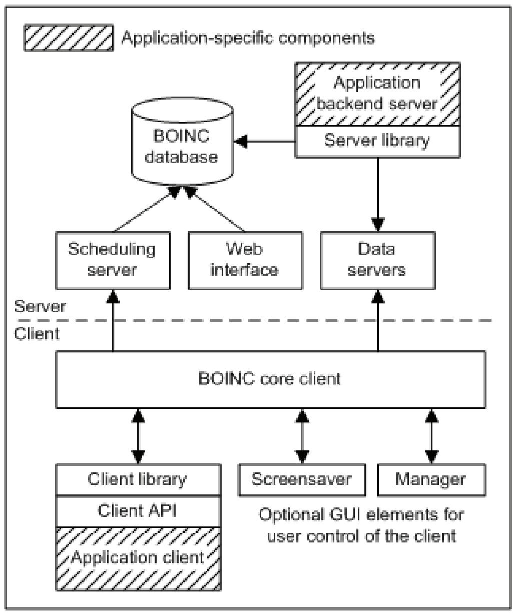
- It has the largest EDGC deployment base. More than 80,000 Condor hosts around the world make up approximately 160 production-level Condor pools (see <http://www.cs.wisc.edu/condor/map/> for updated Condor statistics).
- It is available free of cost.

PRC Middleware BOINC

The BOINC system (see Figure 3) 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.

The widespread availability of desktop PCs in organizations makes the deployment of an enterprise-wide BOINC infrastructure an 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. (©2007 IEEE. Used with permission.)

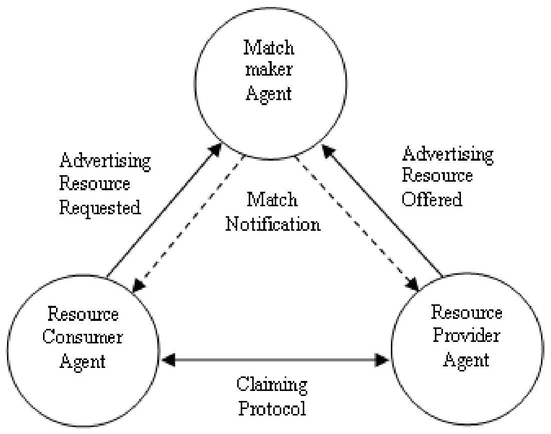


EDGC Middleware Condor

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. 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*

Figure 4. Condor resource management architecture - adapted from Basney and Livney (1999)



agent, 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).

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 an appropriate Condor universe. *Java universe* supports the execution of java programs and is appropriate for executing CSP-based simulations over Condor.

Three different approaches to integrating CSPs with grid computing middleware are discussed next.

CSP-GRID INTEGRATION APPROACHES

Three possible approaches for using desktop grids with CSPs are the *CSP-middleware integration approach*, the *CSP-runtime installation approach* and the *CSP-preinstalled approach* (Mustafee & Taylor, 2008).

CSP-Grid Middleware Integration Approach

One possible approach to 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. 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). This approach is therefore not considered feasible.

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 requiring 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).

CSP-Preinstalled Approach

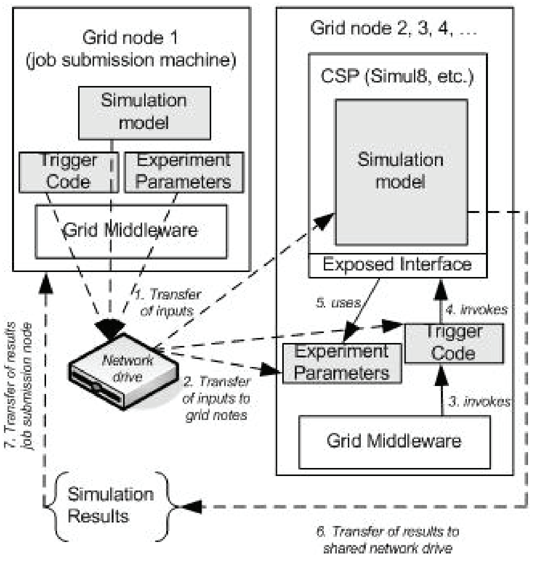
The third CSP-grid integration approach is to install the CSP in the desktop grid resource, just like any other application is installed on a PC. The drawback with this approach is that the sandbox security mechanism implemented by most enterprise desktop grids may have to be forfeited. However, as simulations 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-supported grid middleware like BOINC and Condor can be used, and (3) CSPs that are installed on the user PCs can be utilized for running simulation experiments from other users.

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,

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



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 into 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.

The next section of this chapter uses Microsoft Excel together with BOINC and Condor middleware to execute Monte Carlo simulations over the grid. Although Excel has been used here as an example, the CSP-preinstalled approach can generally be used with other DES and MCS CSPs that expose package functionality.

INTERFACING BOINC AND CONDOR WITH CSPs

Interfacing BOINC with Excel

This section is structured as follows. Following an overview, the next sub-section describes the Excel-based MCS application (Range Accrual Swap [RAS]) that is used as an example. This is followed by a technical discussion on how the RAS application is grid-enabled.

Overview

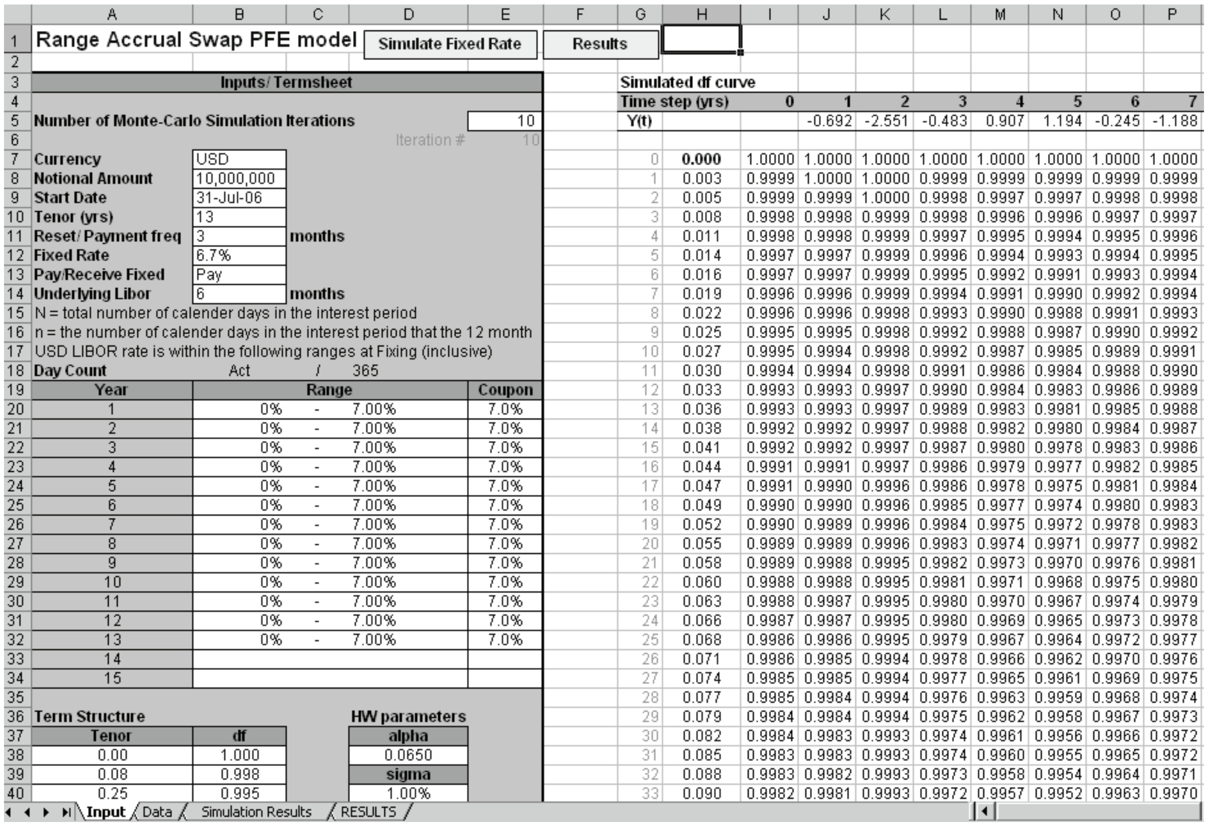
BOINC middleware is primarily used for scientific computing using millions of volunteer PCs. However, it should also be possible to use the PRC middleware within an organization for the processing of enterprise applications. Using the Excel-based RAS application, this research now investigates how BOINC can be used in a desktop grid environment to provide task farming service to the CSPs. Arguably, this is the first attempt to use a PRC middleware in an enterprise environment. There are no existing examples of enterprise application processing using BOINC in literature.

Range Accrual Swap (RAS) Application

The application that is used to implement task farming using BOINC is a Microsoft Excel-based spreadsheet application used for financial modelling by a leading European financial institution. The financial model calculates the risk of a *Range Accrual Swap* at various points in time until the maturity of the transactions. Range Accrual Swap is a type of financial derivative instrument in which certain fixed cash flows are exchanged for an uncertain stream of cash flows based on the movement of interest rates in the future. A screenshot of the RAS application is shown in Figure 6.

The successful and accurate calculation of risk using the RAS application requires a large number of MCS and takes a significant amount of time. Each simulation run (iteration) is independent of previous runs and is characterized by the generation of random values for various defined variables and by solving equations containing these variables. The conventional approach of using only one instance of Excel is not feasible in situations where the business desires a quick turnaround (answer). One solution to this is to distribute the processing of the MCS model over a grid and utilize the spare processing power of the grid nodes and the Excel software installed on them. This grid-facilitated execution of the RAS model has the potential of speeding up the simulation of the financial models manifold, depending on the number of grid nodes available and whether they are dedicated or non-dedicated resources.

Figure 6. Range Accrual Swap (RAS) application (created by the credit risk division of a leading European investment bank)



Range Accrual Swap PFE model | Simulate Fixed Rate | Results

Inputs/ Termsheet

Number of Monte-Carlo Simulation Iterations	10
Iteration #	10
Currency	USD
Notional Amount	10,000,000
Start Date	31-Jul-06
Tenor (yrs)	13
Reset/ Payment freq	3 months
Fixed Rate	6.7%
Pay/Receive Fixed	Pay
Underlying Libor	6 months

N = total number of calender days in the interest period
n = the number of calender days in the interest period that the 12 month
USD LIBOR rate is within the following ranges at Fixing (inclusive)

Day Count Act / 365

Year	Range			Coupon
1	0%	-	7.00%	7.0%
2	0%	-	7.00%	7.0%
3	0%	-	7.00%	7.0%
4	0%	-	7.00%	7.0%
5	0%	-	7.00%	7.0%
6	0%	-	7.00%	7.0%
7	0%	-	7.00%	7.0%
8	0%	-	7.00%	7.0%
9	0%	-	7.00%	7.0%
10	0%	-	7.00%	7.0%
11	0%	-	7.00%	7.0%
12	0%	-	7.00%	7.0%
13	0%	-	7.00%	7.0%
14				
15				

Term Structure

Tenor	df
0.00	1.000
0.08	0.998
0.25	0.995

HW parameters

alpha
0.0650
sigma
1.00%

Simulated df curve

Time step (yrs)		0	1	2	3	4	5	6	7
Y(t)			-0.692	-2.551	-0.483	0.907	1.194	-0.245	-1.188
0	**0.000**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.003	0.9999	1.0000	1.0000	0.9999	0.9999	0.9999	0.9999	0.9999
2	0.005	0.9999	0.9999	1.0000	0.9998	0.9997	0.9997	0.9998	0.9998
3	0.008	0.9998	0.9998	0.9999	0.9998	0.9996	0.9996	0.9997	0.9997
4	0.011	0.9998	0.9998	0.9999	0.9997	0.9995	0.9994	0.9995	0.9996
5	0.014	0.9997	0.9997	0.9999	0.9996	0.9994	0.9993	0.9994	0.9995
6	0.016	0.9997	0.9997	0.9999	0.9995	0.9992	0.9991	0.9993	0.9994
7	0.019	0.9996	0.9996	0.9999	0.9994	0.9991	0.9990	0.9992	0.9994
8	0.022	0.9996	0.9996	0.9998	0.9993	0.9990	0.9988	0.9991	0.9993
9	0.025	0.9995	0.9995	0.9998	0.9992	0.9988	0.9987	0.9990	0.9992
10	0.027	0.9995	0.9994	0.9998	0.9992	0.9987	0.9985	0.9989	0.9991
11	0.030	0.9994	0.9994	0.9998	0.9991	0.9986	0.9984	0.9988	0.9990
12	0.033	0.9993	0.9993	0.9997	0.9990	0.9984	0.9983	0.9986	0.9989
13	0.036	0.9993	0.9993	0.9997	0.9989	0.9983	0.9981	0.9985	0.9988
14	0.038	0.9992	0.9992	0.9997	0.9988	0.9982	0.9980	0.9984	0.9987
15	0.041	0.9992	0.9992	0.9997	0.9987	0.9980	0.9978	0.9983	0.9986
16	0.044	0.9991	0.9991	0.9997	0.9986	0.9979	0.9977	0.9982	0.9985
17	0.047	0.9991	0.9990	0.9996	0.9986	0.9978	0.9975	0.9981	0.9984
18	0.049	0.9990	0.9990	0.9996	0.9985	0.9977	0.9974	0.9980	0.9983
19	0.052	0.9990	0.9989	0.9996	0.9984	0.9975	0.9972	0.9978	0.9983
20	0.055	0.9989	0.9989	0.9996	0.9983	0.9974	0.9971	0.9977	0.9982
21	0.058	0.9989	0.9988	0.9995	0.9982	0.9973	0.9970	0.9976	0.9981
22	0.060	0.9988	0.9988	0.9995	0.9981	0.9971	0.9968	0.9975	0.9980
23	0.063	0.9988	0.9987	0.9995	0.9980	0.9970	0.9967	0.9974	0.9979
24	0.066	0.9987	0.9987	0.9995	0.9980	0.9969	0.9965	0.9973	0.9978
25	0.068	0.9986	0.9986	0.9995	0.9979	0.9967	0.9964	0.9972	0.9977
26	0.071	0.9986	0.9985	0.9994	0.9978	0.9966	0.9962	0.9970	0.9976
27	0.074	0.9985	0.9985	0.9994	0.9977	0.9965	0.9961	0.9969	0.9975
28	0.077	0.9985	0.9984	0.9994	0.9976	0.9963	0.9959	0.9968	0.9974
29	0.079	0.9984	0.9984	0.9994	0.9975	0.9962	0.9958	0.9967	0.9973
30	0.082	0.9984	0.9983	0.9993	0.9974	0.9961	0.9956	0.9966	0.9972
31	0.085	0.9983	0.9983	0.9993	0.9974	0.9960	0.9955	0.9965	0.9972
32	0.088	0.9983	0.9982	0.9993	0.9973	0.9958	0.9954	0.9964	0.9971
33	0.090	0.9982	0.9981	0.9993	0.9972	0.9957	0.9952	0.9963	0.9970

Input / Data / Simulation Results / RESULTS

Grid-Enabling RAS Application

A BOINC-based project requires application specific implementation on both the client side and the server side. The client side implementation usually consists of writing a C++ application client that uses BOINC client library and APIs to integrate with the BOINC core client. The core client is downloaded from the BOINC website, installed on individual PCs and is attached to a BOINC project. Once successfully attached the core client downloads the project specific application client and work units for processing. The core client, which is in effect the manager of a compute resource, makes available CPU cycles to the attached project based on the user's preferences. These preferences can be set using either the menu provided by the core client (Figure 7) or through a web interface (Figure 8). The latter offers the user more flexibility in specifying CPU, memory, disk and network usage. The core client can support multiple BOINC-based projects, but

Figure 7. Setting user preference using menu provided by BOINC core client

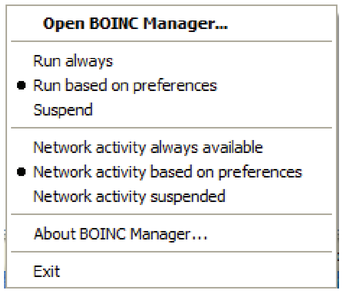
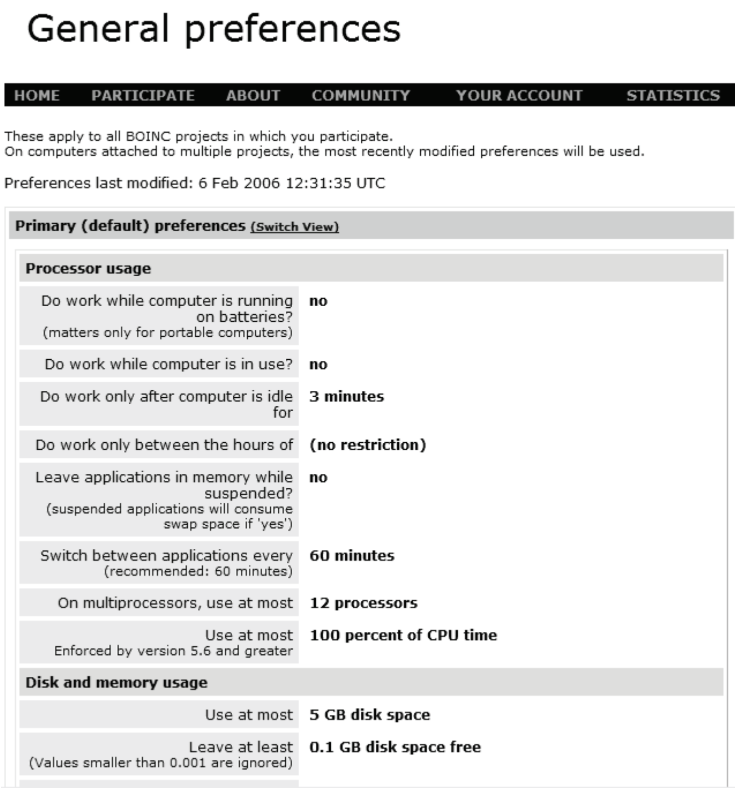

Open BOINC Manager...
Run always
● Run based on preferences
Suspend
Network activity always available
● Network activity based on preferences
Network activity suspended
About BOINC Manager...
Exit

Figure 8. Setting user preference using web interface



General preferences

HOME PARTICIPATE ABOUT COMMUNITY YOUR ACCOUNT STATISTICS

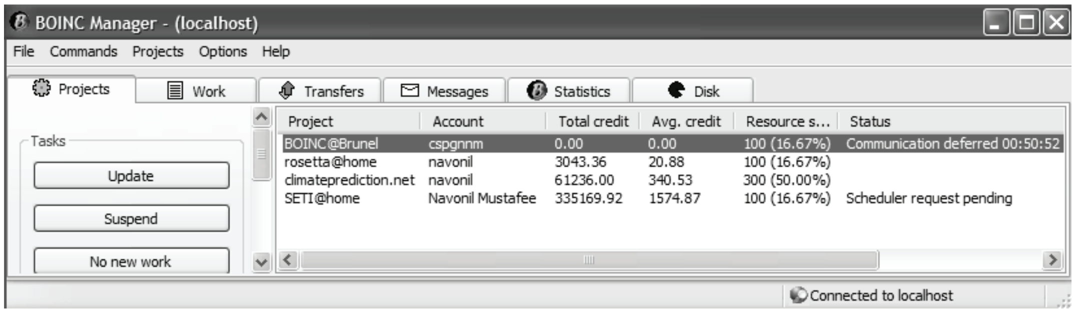
These apply to all BOINC projects in which you participate.
On computers attached to multiple projects, the most recently modified preferences will be used.

Preferences last modified: 6 Feb 2006 12:31:35 UTC

Primary (default) preferences (Switch View)

Processor usage	
Do work while computer is running on batteries? (matters only for portable computers)	**no**
Do work while computer is in use?	**no**
Do work only after computer is idle for	**3 minutes**
Do work only between the hours of	**(no restriction)**
Leave applications in memory while suspended? (suspended applications will consume swap space if 'yes')	**no**
Switch between applications every (recommended: 60 minutes)	**60 minutes**
On multiprocessors, use at most	**12 processors**
Use at most Enforced by version 5.6 and greater	**100 percent of CPU time**
Disk and memory usage	
Use at most	**5 GB disk space**
Leave at least (Values smaller than 0.001 are ignored)	**0.1 GB disk space free**

Figure 9. BOINC core client attached to multiple projects


BOINC Manager - (localhost)
File Commands Projects Options Help
Projects Work Transfers Messages Statistics Disk
Tasks
Update
Suspend
No new work
Connected to localhost

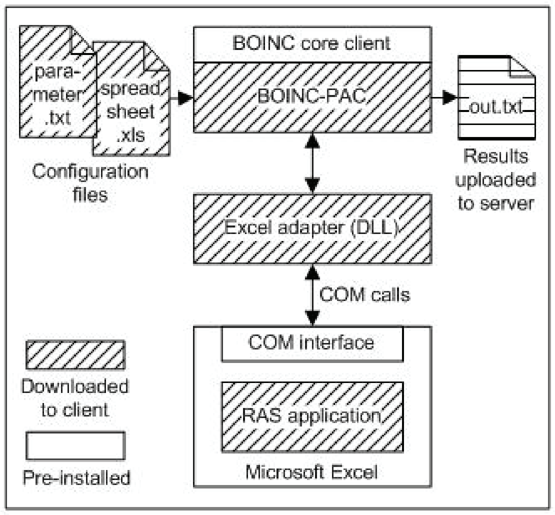
Project	Account	Total credit	Avg. credit	Resource s...	Status
BOINC@Brunel	cspgnnm	0.00	0.00	100 (16.67%)	Communication deferred 00:50:52
rosetta@home	navonil	3043.36	20.88	100 (16.67%)	
climateprediction.net	navonil	61236.00	340.53	300 (50.00%)	
SETI@home	Navonil Mustafee	335169.92	1574.87	100 (16.67%)	Scheduler request pending

at any one time only one project can be executed. This is illustrated in Figure 9 where four different BOINC projects, viz, BOINC@Brunel, Rosetta@Home, ClimatePrediction.net and SETI@home, are attached but only one project (SETI@home) is communicating with the BOINC server side scheduler.

In this chapter the software that has been developed to integrate BOINC with Excel is referred to as *BOINC Proxy Application Client* or *BOINC-PAC* for short. It assumes that Microsoft Excel is installed on all the BOINC client computers.

BOINC-PAC is implemented in Visual C++. The VC++ code invokes CSP-specific operations (through interfaces exposed by the CSPs) defined by a Visual Basic DLL adapter. BOINC-PAC uses the BOINC client library and APIs to interface with the BOINC core client. It interacts with the

Figure 10. Execution of RAS application using BOINC



Excel adapter to execute operations on the RAS Excel-based spreadsheet. The Excel adapter, in turn, uses the COM interface of Excel to perform basic operations like opening and closing the simulation file, setting the number of iterations, executing the simulation and writing the results of the simulation to a text file (*out.txt*). The text file is subsequently uploaded to the BOINC server. The number of Monte Carlo iterations to be performed by the RAS application is not hard-coded and is read by BOINC-PAC from a parameter file (*parameter.txt* in Figure 10). The interaction of the different program components is shown in Figure 10. Once the BOINC-PAC is downloaded by the core client onto a PC it triggers the execution of the RAS MCS by utilizing the Excel software installed on the local resource.

The discussion that follows mainly concerns the BOINC server side implementation for the RAS application. When the BOINC core client first attaches itself with the RAS project it downloads the BOINC-PAC from the BOINC server. This application consists of a VC++ executable and a client initialization file called init_data.xml. Subsequently, the core client downloads the project workunits. In BOINC one unit of computation is represented as a workunit. These workunits are created using the BOINC *create_work* command and then placed in the download directory of the BOINC server. The arguments supplied to the *create_work* command include, among others, (1) the workunit template filename, (2) the result template filename and (3) the *command_line* parameter. The template files are XML files that describe the workunit (*work_unit_template.xml*) and its corresponding results (*result_template.xml*). The workunits are created by running a program that invokes the *create_work* command in a loop to generate the required number of workunits. The arguments to the *create_work* command are described next:

- The "workunit template file" lists the input files that are packed together as a workunit. In the RAS BOINC project the input files are the RAS Excel-based spreadsheet, the Excel adapter, and the parameter file. The workunit template file also mentions the quorum (XML tag <*min_quorum*>) and the maximum total results (XML tag <*max_total_results*>). However, since BOINC is being used in an enterprise grid environment that assumes some form of centralized control over the computing resources, the value for both <*min_quorum*> and <*max_total_results*> are set to one. In other words, it is expected that all the results that are returned are valid and therefore the same workunit will not be sent to more than one BOINC node.
- The "result template file" lists the files that will be uploaded to the BOINC server after the results have been computed by the BOINC-PAC. In the RAS application, the file that is uploaded from each BOINC client is called out.txt. As has been said earlier, this file contains the results of the RAS simulation.
- The optional *command_line* argument in the *create_work* command is used to pass

a position value to the BOINC-PAC application. This position value represents an experiment number and BOINC-PAC reads the parameter file *parameter.txt* to extract the value at this position. This value, in our case, represents the number of Monte Carlo iterations that have to be performed on a simulation experiment being run on the client. The use of the *command_line* argument is specific to the BOINC-PAC application being developed.

200 MCS experiments (each with 300 iterations) were conducted in this study. A Java program was used to iteratively create these 200 work units by invoking *create_work* with *command_line* argument. These workunits were downloaded by different BOINC nodes and the RAS application executed using the locally installed MCS CSP Excel. The results of the simulation were then automatically uploaded to the BOINC project server.

Interfacing Condor with CSPs

In this section EDGC middleware Condor is used to execute two different Excel-based Monte Carlo simulations simultaneously on different grid nodes. An overview of the case study is presented. The two applications being grid-enabled with the objective of executing them concurrently using Condor middleware are – the Asian Option application and the Range Accrual Swap application. The last section then discusses the technology used to grid-enable these applications.

Overview

Having the capability to run two or more simulation applications concurrently has the potential to execute different CSP models, which may belong to different simulation users, simultaneously over the grid. Furthermore, these models may be created and executed using different CSPs. However, in this hypothetical case study, models created using the same MCS CSP (Microsoft Excel) are used. The first model is called the Asian Option application which has been created by Professor Eduardo Saliby (Federal University of Rio de Janeiro, Brazil; visiting professor at Brunel University, UK). The second model is the RAS application that has been previously used in the BOINC case study. The RAS model has been created by the credit risk division of a major investment bank.

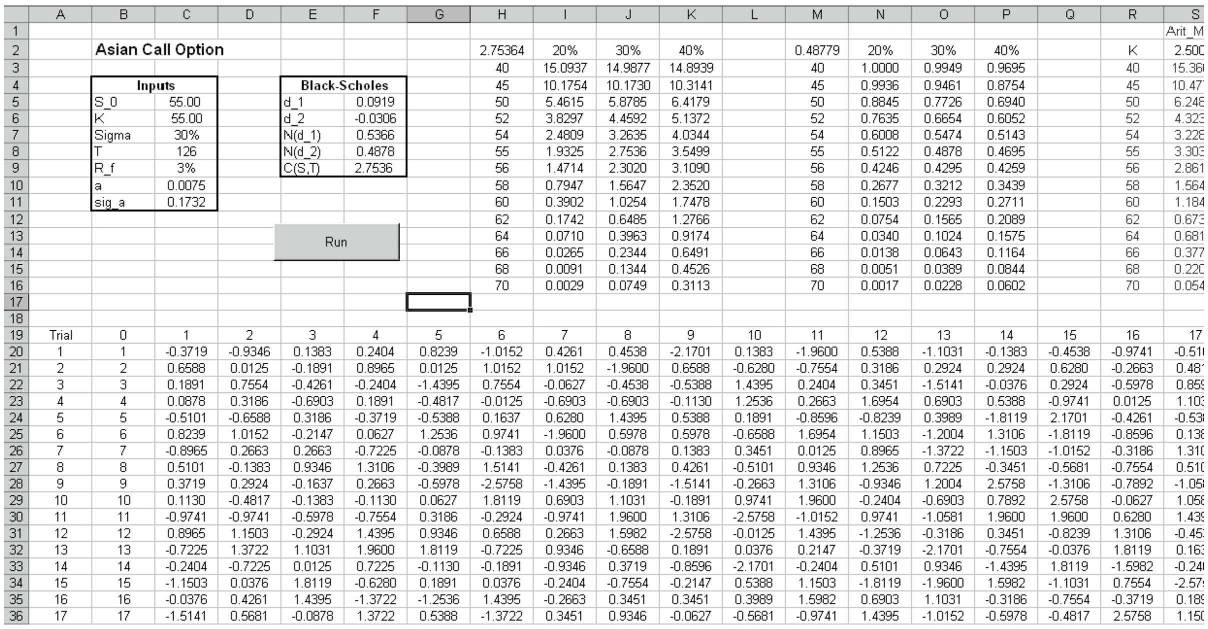
Asian Options (AO) Application

The Asian Options Application uses Descriptive Sampling, which can be seen as a variance reduction technique, to calculate options whose payoffs are path-dependent on the underlying asset prices during the life of the option (Marins et al., 2004). The AO application estimates the value of the Asian options by simulating the model a number of times and then calculating the average of the results of the individual iterations. On a single PC, executing multiple iterations of the AO application takes a significant amount of time. CSP-specific task farming service has the potential to reduce the time taken to process the AO application by distributing its processing over multiple grid nodes. An average of the results returned from each node can then be calculated to determine the value of the options. Figure 11 shows the Microsoft Excel-based AO application.

Range Accrual Swap (RAS) Application

The RAS application has already been described in the preceding pages. The application is the same but the technologies used for interfacing RAS with BOINC and RAS with Condor are different. The integration of RAS with BOINC has been discussed earlier. The section that follows describes how both RAS and AO are used with the Condor Java universe execution environment.

Figure 11. Asian Options (AO) application (created by Professor Eduardo Saliby, Federal University of Rio de Janeiro, Brazil)



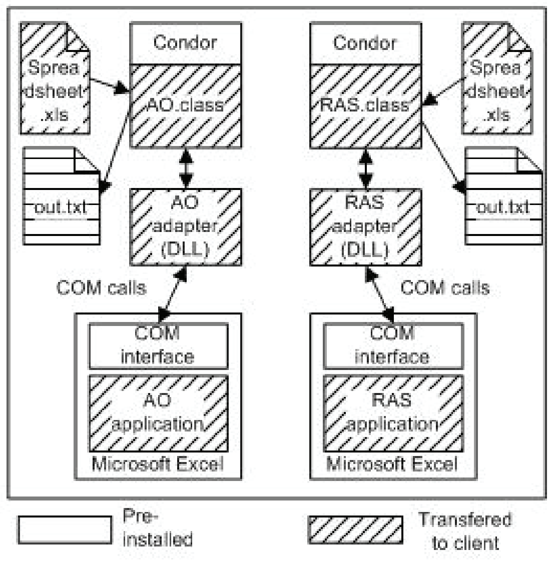
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1																			Arit_M
2		Asian Call Option						2.75364	20%	30%	40%		0.48779	20%	30%	40%		K	2.50
3								40	15.0937	14.9877	14.8939		40	1.0000	0.9949	0.9695		40	15.36
4		**Inputs**			**Black-Scholes**			45	10.1754	10.1730	10.3141		45	0.9936	0.9461	0.8754		45	10.47
5		S_0	55.00		d_1	0.0919		50	5.4615	5.8785	6.4179		50	0.8845	0.7726	0.6940		50	6.24
6		K	55.00		d_2	-0.0306		52	3.8297	4.4592	5.1372		52	0.7635	0.6654	0.6052		52	4.32
7		Sigma	30%		N(d_1)	0.5366		54	2.4809	3.2635	4.0344		54	0.6008	0.5474	0.5143		54	3.22
8		T	126		N(d_2)	0.4878		55	1.9325	2.7536	3.5499		55	0.5122	0.4878	0.4695		55	3.30
9		R_f	3%		C(S,T)	2.7536		56	1.4714	2.3020	3.1090		56	0.4246	0.4295	0.4259		56	2.861
10		a	0.0075					58	0.7947	1.5647	2.3520		58	0.2677	0.3212	0.3439		58	1.564
11		sig_a	0.1732					60	0.3902	1.0254	1.7478		60	0.1503	0.2293	0.2711		60	1.184
12								62	0.1742	0.6485	1.2766		62	0.0754	0.1565	0.2089		62	0.673
13					Run			64	0.0710	0.3963	0.9174		64	0.0340	0.1024	0.1575		64	0.681
14								66	0.0265	0.2344	0.6491		66	0.0138	0.0643	0.1164		66	0.377
15								68	0.0091	0.1344	0.4526		68	0.0051	0.0389	0.0844		68	0.220
16								70	0.0029	0.0749	0.3113		70	0.0017	0.0228	0.0602		70	0.054
17																			
18																			
19	Trial	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
20	1	1	-0.3719	-0.9346	0.1383	0.2404	0.8239	-1.0152	0.4261	0.4538	-2.1701	0.1383	-1.9600	0.5388	-1.1031	-0.1383	-0.4538	-0.9741	-0.51
21	2	2	0.6588	0.0125	-0.1891	0.8965	0.0125	1.0152	1.0152	-1.9600	0.6588	-0.6280	-0.7554	0.3186	0.2924	0.2924	0.6280	-0.2663	0.48
22	3	3	0.1891	0.7554	-0.4261	-0.2404	-1.4395	0.7554	-0.0627	-0.4538	-0.5388	1.4395	0.2404	0.3451	-1.5141	-0.0376	0.2924	-0.5978	0.85
23	4	4	0.0878	0.3186	-0.6903	0.1891	-0.4817	-0.0125	-0.6903	-0.6903	-0.1130	1.2536	0.2663	1.6954	0.6903	0.5388	-0.9741	0.0125	1.10
24	5	5	-0.5101	-0.6588	0.3186	-0.3719	-0.5388	0.1637	0.6280	1.4395	0.5388	0.1891	-0.8596	-0.8239	0.3989	-1.8119	2.1701	-0.4261	-0.53
25	6	6	0.8239	1.0152	-0.2147	0.0627	1.2536	0.9741	-1.9600	0.5978	0.5978	-0.6588	1.6954	1.1503	-1.2004	1.3106	-1.8119	-0.8596	0.13
26	7	7	-0.8965	0.2663	0.2663	-0.7225	-0.0878	-0.1383	0.0376	-0.0878	0.1383	0.3451	0.0125	0.8965	-1.3722	-1.1503	-1.0152	-0.3186	1.31
27	8	8	0.5101	-0.1383	0.9346	1.3106	-0.3989	1.5141	-0.4261	0.1383	0.4261	-0.5101	0.9346	1.2536	0.7225	-0.3451	-0.5681	-0.7554	0.51
28	9	9	0.3719	0.2924	-0.1637	0.2663	-0.5978	-2.5758	-1.4395	-0.1891	-1.5141	-0.2663	1.3106	-0.9346	1.2004	2.5758	-1.3106	-0.7892	-1.05
29	10	10	0.1130	-0.4817	-0.1383	-0.1130	0.0627	1.8119	0.6903	1.1031	-0.1891	0.9741	1.9600	-0.2404	-0.6903	0.7892	2.5758	-0.0627	1.05
30	11	11	-0.9741	-0.9741	-0.5978	-0.7554	0.3186	-0.2924	-0.9741	1.9600	1.3106	-2.5758	-1.0152	0.9741	-1.0581	1.9600	1.9600	0.6280	1.43
31	12	12	0.8965	1.1503	-0.2924	1.4395	0.9346	0.6588	0.2663	1.5982	-2.5758	-0.0125	1.4395	-1.2536	-0.3186	0.3451	-0.8239	1.3106	-0.45
32	13	13	-0.7225	1.3722	1.1031	1.9600	1.8119	-0.7225	0.9346	-0.6588	0.1891	0.0376	0.2147	-0.3719	-2.1701	-0.7554	-0.0376	1.8119	0.16
33	14	14	-0.2404	-0.7225	0.0125	0.7225	-0.1130	-0.1891	-0.9346	0.3719	-0.8596	-2.1701	-0.2404	0.5101	0.9346	-1.4395	1.8119	-1.5982	-0.24
34	15	15	-1.1503	0.0376	1.8119	-0.6280	0.1891	0.0376	-0.2404	-0.7554	-0.2147	0.5388	1.1503	-1.8119	-1.9600	1.5982	-1.1031	0.7554	-2.57
35	16	16	-0.0376	0.4261	1.4395	-1.3722	-1.2536	1.4395	-0.2663	0.3451	0.3451	0.3989	1.5982	0.6903	1.1031	-0.3186	-0.7554	-0.3719	0.18
36	17	17	-1.5141	0.5681	-0.0878	1.3722	0.5388	-1.3722	0.3451	0.9346	-0.0627	-0.5681	-0.9741	1.4395	-1.0152	-0.5978	-0.4817	2.5758	1.15

Grid-Enabling AO and RAS Applications

The Condor Java universe execution environment is designed for the execution of Java programs. Different Java programs (AO.class and RAS.class) and adapters (AO adapter and RAS adapter) have been developed for AO and RAS applications respectively. As shown in Figure 12, the AO.class/RAS.class communicates with the AO/RAS adapter to control the Excel-based AO/RAS application. The results of the simulation are written back to their respective out.txt files, which are then transferred back to the Condor node from which the jobs were originally submitted. The figure also shows the files that have been transferred to the remote Condor nodes from the job submission node. Both the AO and RAS applications are executed concurrently over the Condor pool.

The discussion now focuses on the Condor mechanism that allows the submission of multiple jobs. There are two applications in this case study.

Figure 12. Execution of RAS and AO applications on a Condor pool

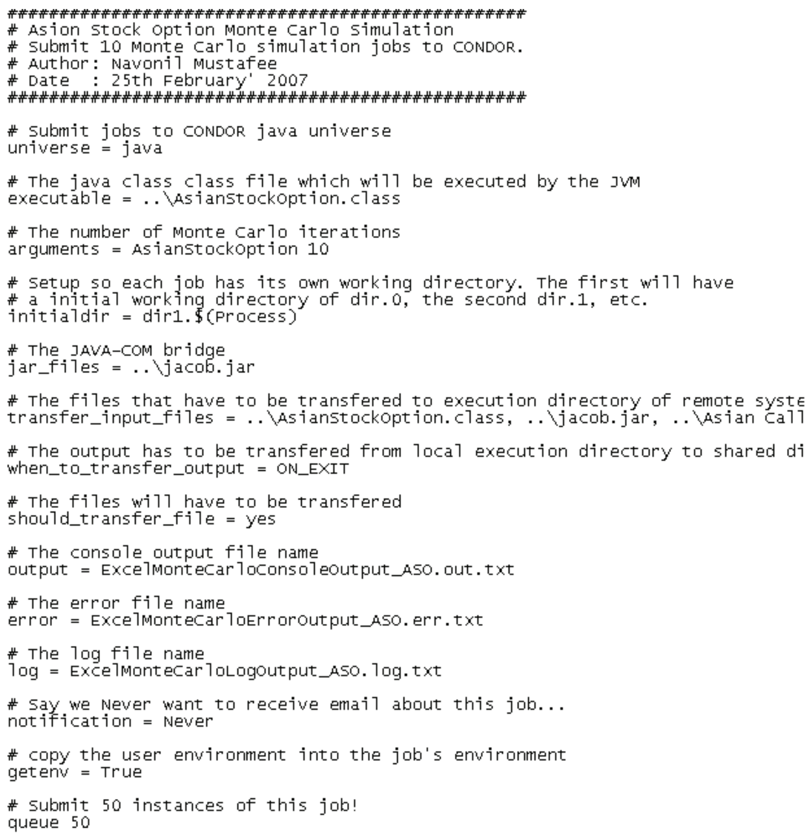


For supporting multiple applications it is generally required that it should be possible to submit multiple instances of each application over the Condor pool. The job submission file is used to achieve this. Every Condor job has a corresponding job submit file (*.sub* file) that defines variables that control different aspects of job submission. The most important of these Condor-defined variables, for the purpose of task farming, is the *queue* variable. The integer value assigned to this variable determines the number of replications of the same job that are to be executed over the Condor pool. Figures 13 and 14 show the .sub file for the AO and the RAS applications respectively. The value "50" assigned to the *queue* variable (the last variable in the screenshots) suggests that both the AO and the RAS applications will be executed for a total of 50 times over different grid nodes. Some of the other job submission variables shown in the .sub file are discussed next.

The *universe* variable is assigned a value "Java" because the Condor Java execution environment is being used to run the simulations. The *executable* variable defines the name of the Java class file that has the *main()* method. The *argument* variable is used to pass a command line argument to the Java program. For this hypothetical case study, the number of iterations for each simulation model has been set to a modest value of "10" through the use of this *argument* variable. The reader is however reminded that both AO and RAS applications will be executed 50 times over, and therefore the total number of simulation iterations for each application, taken as a whole, will be 500 (50*10).

Each simulation experiment will have a unique working directory associated with it. These directories should be present on the Condor node from which jobs are submitted, or on a network drive that can be accessed by the job submission

Figure 13. Job submit file for AO application

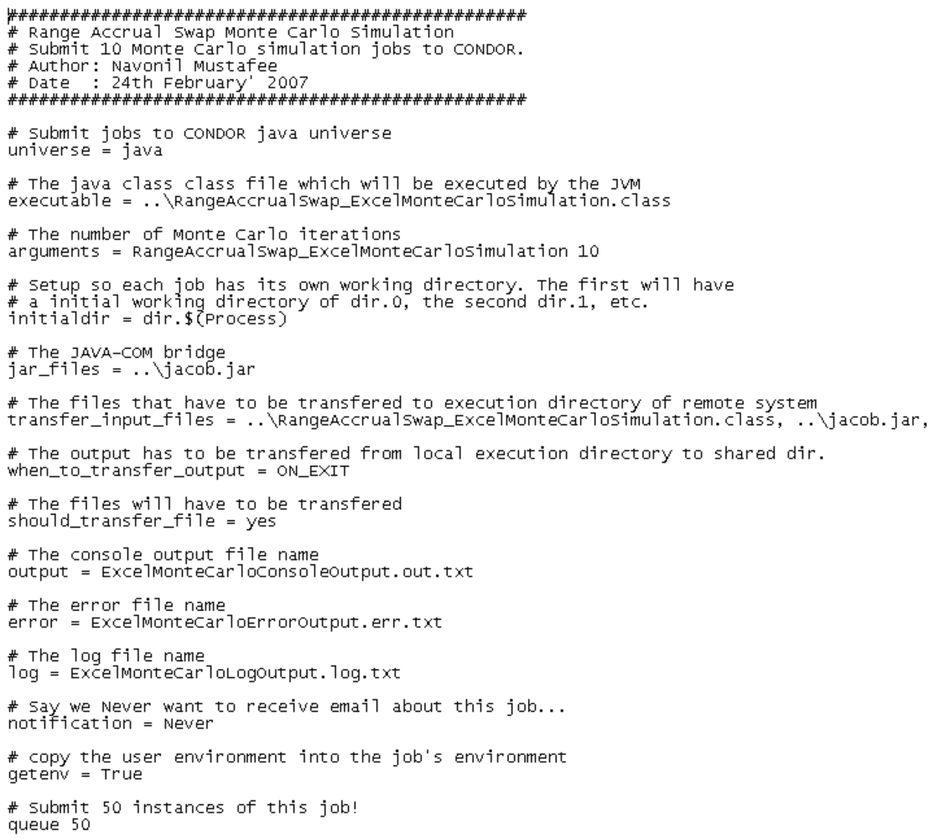


```
##################################################
# Asion Stock Option Monte Carlo Simulation
# Submit 10 Monte Carlo simulation jobs to CONDOR.
# Author: Navonil Mustafee
# Date  : 25th February' 2007
##################################################

# Submit jobs to CONDOR java universe
universe = java

# The java class class file which will be executed by the JVM
executable = ..\AsianStockOption.class

# The number of Monte Carlo iterations
arguments = AsianStockOption 10

# Setup so each job has its own working directory. The first will have
# a initial working directory of dir.0, the second dir.1, etc.
initialdir = dir1.$(Process)

# The JAVA-COM bridge
jar_files = ..\jacob.jar

# The files that have to be transfered to execution directory of remote syste
transfer_input_files = ..\AsianStockOption.class, ..\jacob.jar, ..\Asian Call

# The output has to be transfered from local execution directory to shared di
when_to_transfer_output = ON_EXIT

# The files will have to be transfered
should_transfer_file = yes

# The console output file name
output = ExcelMonteCarloConsoleOutput_ASO.out.txt

# The error file name
error = ExcelMonteCarloErrorOutput_ASO.err.txt

# The log file name
log = ExcelMonteCarloLogOutput_ASO.log.txt

# Say we Never want to receive email about this job...
notification = Never

# copy the user environment into the job's environment
getenv = True

# Submit 50 instances of this job!
queue 50
```

node. The working directories are represented by the variable *initialdir*. In the case of the AO and the RAS applications the values assigned to this variable are "dir1.$(process)" and "dir.$(process)" respectively. *$process* is a Condor-defined integer variable that is incremented automatically depending on the number of instances of a particular job that have been submitted. Thus, if *queue*=50 then the value of the *$process* variable will start from 1 and will end at 50. This in turn suggests that the working directory for the first job will be "dir1.1" and for the last job it will be "dir1.50" (in case of AO application). These working directories are important because they will contain the results of the individual experiments and the log files that are output by Condor during execution of each experiment (Figure 15). The variables that define the names of the three different Condor log files for console output, error information and Condor-specific messages are *output*, *error* and *log* respectively. It has to be added, however, that a Condor job is in-effect executed under a temporary directory that it created by Condor on the grid node that is assigned the task of processing the job (Figure 16 shows a temporary directory called "dir_3768" that has been created for executing one instance of a simulation). Once the simulation is complete, the results from the temporary directory are transferred to the individual working directories and the temporary directory deleted.

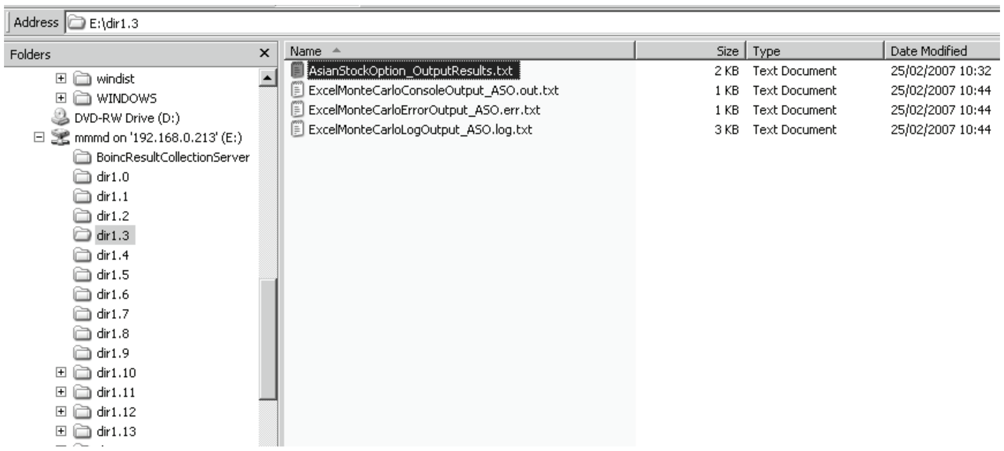
The files to be transferred to the execution host are indicated by the *transfer_input_files* variable. These files are transferred to the temporary execution directory created by the job executing node. The variable *when_to_transfer_output* and its

Figure 14. Job submit file for RAS application



```
##################################################
# Range Accrual Swap Monte Carlo Simulation
# Submit 10 Monte Carlo simulation jobs to CONDOR.
# Author: Navonil Mustafee
# Date  : 24th February' 2007
##################################################

# Submit jobs to CONDOR java universe
universe = java

# The java class class file which will be executed by the JVM
executable = ..\RangeAccrualSwap_ExcelMonteCarloSimulation.class

# The number of Monte Carlo iterations
arguments = RangeAccrualSwap_ExcelMonteCarloSimulation 10

# Setup so each job has its own working directory. The first will have
# a initial working directory of dir.0, the second dir.1, etc.
initialdir = dir.$(Process)

# The JAVA-COM bridge
jar_files = ..\jacob.jar

# The files that have to be transfered to execution directory of remote system
transfer_input_files = ..\RangeAccrualSwap_ExcelMonteCarloSimulation.class, ..\jacob.jar,

# The output has to be transfered from local execution directory to shared dir.
when_to_transfer_output = ON_EXIT

# The files will have to be transfered
should_transfer_file = yes

# The console output file name
output = ExcelMonteCarloConsoleOutput.out.txt

# The error file name
error = ExcelMonteCarloErrorOutput.err.txt

# The log file name
log = ExcelMonteCarloLogOutput.log.txt

# Say we Never want to receive email about this job...
notification = Never

# copy the user environment into the job's environment
getenv = True

# Submit 50 instances of this job!
queue 50
```

Figure 15. Results from the simulation experiments



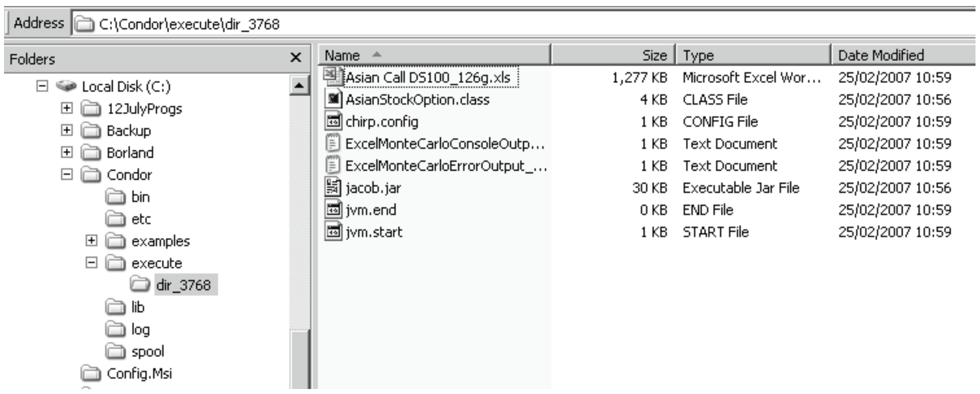
Address E:\dir1.3

Folders

windist
WINDOWS
DVD-RW Drive (D:)
mmmd on '192.168.0.213' (E:)
BoincResultCollectionServer
dir1.0
dir1.1
dir1.2
dir1.3
dir1.4
dir1.5
dir1.6
dir1.7
dir1.8
dir1.9
dir1.10
dir1.11
dir1.12
dir1.13

Name	Size	Type	Date Modified
AsianStockOption_OutputResults.txt	2 KB	Text Document	25/02/2007 10:32
ExcelMonteCarloConsoleOutput_ASO.out.txt	1 KB	Text Document	25/02/2007 10:44
ExcelMonteCarloErrorOutput_ASO.err.txt	1 KB	Text Document	25/02/2007 10:44
ExcelMonteCarloLogOutput_ASO.log.txt	3 KB	Text Document	25/02/2007 10:44

Figure 16. Condor jobs getting executed in temporary execution directory



Address C:\Condor\execute\dir_3768

Folders

Local Disk (C:)
12JulyProgs
Backup
Borland
Condor
bin
etc
examples
execute
dir_3768
lib
log
spool
Config.Msi

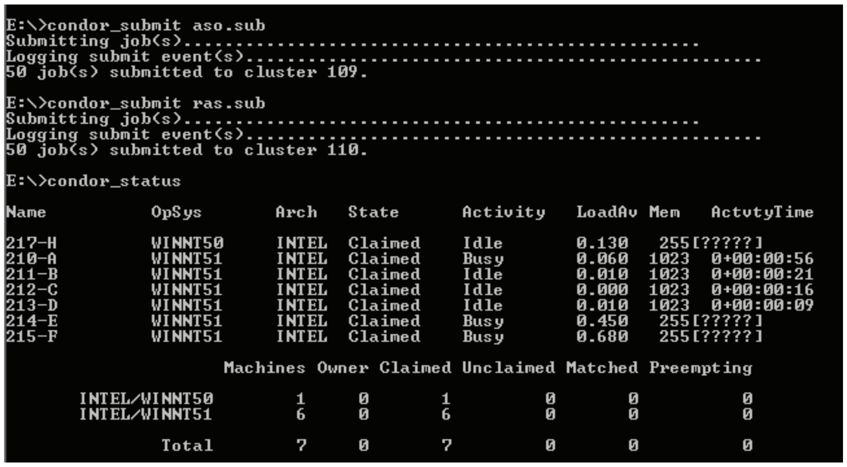
Name	Size	Type	Date Modified
Asian Call DS100_126g.xls	1,277 KB	Microsoft Excel Wor...	25/02/2007 10:59
AsianStockOption.class	4 KB	CLASS File	25/02/2007 10:56
chirp.config	1 KB	CONFIG File	25/02/2007 10:59
ExcelMonteCarloConsoleOutp...	1 KB	Text Document	25/02/2007 10:59
ExcelMonteCarloErrorOutput_...	1 KB	Text Document	25/02/2007 10:59
jacob.jar	30 KB	Executable Jar File	25/02/2007 10:56
jvm.end	0 KB	END File	25/02/2007 10:59
jvm.start	1 KB	START File	25/02/2007 10:59

corresponding value "ON_EXIT" suggest that the simulation results (and the Condor log files) are transferred back from the temporary execution directory to their respective working directories. This concludes the discussion on the variables defined in the Condor submit files.

Jobs are submitted for execution using the Condor command *condor_submit*. The argument to this command is the job description file associated with each job. Figure 17 shows that .sub files for both the AO application (aso.sub) and the RAS application (ras.sub) are submitted using this command, and that 50 instances of each application are created automatically by Condor (see message: "50 jobs(s) submitted to cluster 109/110"). Once the jobs have been submitted the status of the Condor pool can be determined using the command *condor_status*. Figure 17 shows that at present three grid nodes (computers with names 210-A, 214-E and 215-F) are executing the jobs that have been submitted (Activity="Busy"), while the remaining are "Idle". However, all the nodes have been claimed by Condor (State="Claimed") and it is expected that these will soon start executing the simulations.

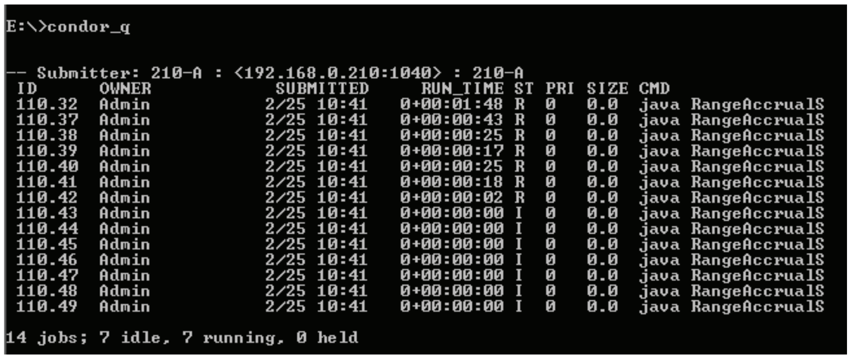
The status of jobs that have been submitted can be found using the command *condor_q*. However, only jobs that are yet to be completed or are presently running are displayed by this command (Figure 18). The jobs that have been completed are not shown.

Figure 17. AO and RAS applications execution over Condor pool



```
E:\>condor_submit aso.sub
Submitting job(s)..................................................
Logging submit event(s)..................................................
50 job(s) submitted to cluster 109.

E:\>condor_submit ras.sub
Submitting job(s)..................................................
Logging submit event(s)..................................................
50 job(s) submitted to cluster 110.

E:\>condor_status

Name          OpSys       Arch   State      Activity   LoadAv Mem   ActvtyTime

217-H         WINNT50     INTEL  Claimed    Idle       0.130   255[?????]
210-A         WINNT51     INTEL  Claimed    Busy       0.060  1023  0+00:00:56
211-B         WINNT51     INTEL  Claimed    Idle       0.010  1023  0+00:00:21
212-C         WINNT51     INTEL  Claimed    Idle       0.000  1023  0+00:00:16
213-D         WINNT51     INTEL  Claimed    Idle       0.010  1023  0+00:00:09
214-E         WINNT51     INTEL  Claimed    Busy       0.450   255[?????]
215-F         WINNT51     INTEL  Claimed    Busy       0.680   255[?????]

                     Machines Owner Claimed Unclaimed Matched Preempting

         INTEL/WINNT50       1     0       1         0       0          0
         INTEL/WINNT51       6     0       6         0       0          0

                 Total       7     0       7         0       0          0
```

Figure 18. Status of job queue displayed using Condor command “condor_q”



```
E:\>condor_q

-- Submitter: 210-A : <192.168.0.210:1040> : 210-A
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 110.32  Admin           2/25 10:41   0+00:01:48 R  0   0.0  java RangeAccrualS
 110.37  Admin           2/25 10:41   0+00:00:43 R  0   0.0  java RangeAccrualS
 110.38  Admin           2/25 10:41   0+00:00:25 R  0   0.0  java RangeAccrualS
 110.39  Admin           2/25 10:41   0+00:00:17 R  0   0.0  java RangeAccrualS
 110.40  Admin           2/25 10:41   0+00:00:25 R  0   0.0  java RangeAccrualS
 110.41  Admin           2/25 10:41   0+00:00:18 R  0   0.0  java RangeAccrualS
 110.42  Admin           2/25 10:41   0+00:00:02 R  0   0.0  java RangeAccrualS
 110.43  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.44  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.45  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.46  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.47  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.48  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.49  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS

14 jobs; 7 idle, 7 running, 0 held
```

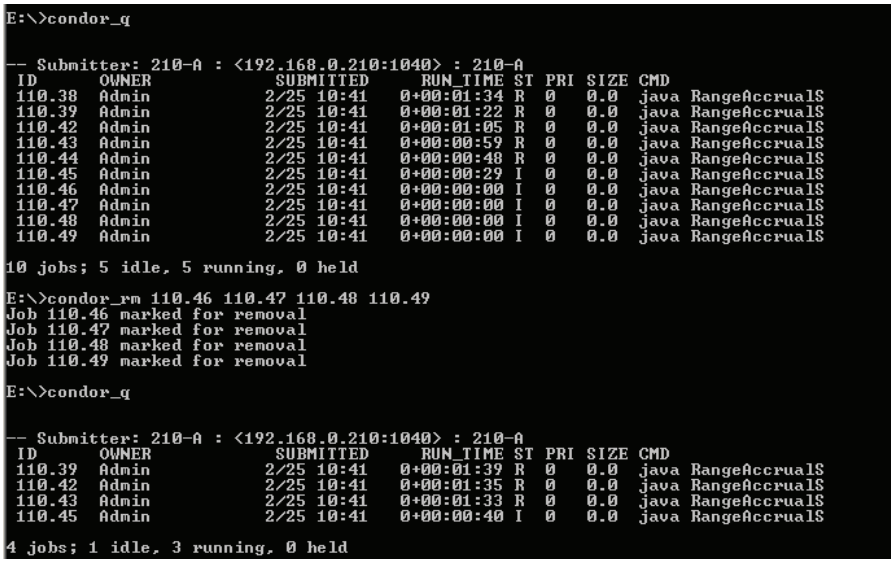
Finally, it is possible to mark submitted jobs for removal from the job queue. This is done using the command *condor_rm*. The job number that represents the job to be deleted has to be provided as an argument to this command. The job number can be determined from the output of the command *condor_q* (field ID). The output of *condor_rm* command is shown in Figure 19.

The concluding section discusses the contribution of the research presented in this book chapter and identifies future research that can be conducted in this area.

DISCUSSION

The research presented in this chapter has been motivated by the advances being made in the field of grid computing and the realization that simulation in industry could potentially benefit through the use of grid computing technologies. This research recognises that end-user adoption of grids could be facilitated by focusing on software tools that are commonly used by employees at their workplace. In the context of simulation in industry, the end-users are the simulation practitioners and the tools that are generally used to model simulations are the CSPs. Thus, this re-

Figure 19. Jobs removed from the queue using Condor command "condor_rm"



```
E:\>condor_q

-- Submitter: 210-A : <192.168.0.210:1040> : 210-A
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 110.38  Admin           2/25 10:41   0+00:01:34 R  0   0.0  java RangeAccrualS
 110.39  Admin           2/25 10:41   0+00:01:22 R  0   0.0  java RangeAccrualS
 110.42  Admin           2/25 10:41   0+00:01:05 R  0   0.0  java RangeAccrualS
 110.43  Admin           2/25 10:41   0+00:00:59 R  0   0.0  java RangeAccrualS
 110.44  Admin           2/25 10:41   0+00:00:48 R  0   0.0  java RangeAccrualS
 110.45  Admin           2/25 10:41   0+00:00:29 I  0   0.0  java RangeAccrualS
 110.46  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.47  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.48  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS
 110.49  Admin           2/25 10:41   0+00:00:00 I  0   0.0  java RangeAccrualS

10 jobs; 5 idle, 5 running, 0 held

E:\>condor_rm 110.46 110.47 110.48 110.49
Job 110.46 marked for removal
Job 110.47 marked for removal
Job 110.48 marked for removal
Job 110.49 marked for removal

E:\>condor_q

-- Submitter: 210-A : <192.168.0.210:1040> : 210-A
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 110.39  Admin           2/25 10:41   0+00:01:39 R  0   0.0  java RangeAccrualS
 110.42  Admin           2/25 10:41   0+00:01:35 R  0   0.0  java RangeAccrualS
 110.43  Admin           2/25 10:41   0+00:01:33 R  0   0.0  java RangeAccrualS
 110.45  Admin           2/25 10:41   0+00:00:40 I  0   0.0  java RangeAccrualS

4 jobs; 1 idle, 3 running, 0 held
```

search has investigated how grid computing can further the field of CSP-based simulation practice and, thereby, offer some benefits to simulation end-users.

This research has identified the form of grid computing, namely PRC in an enterprise context and EDGC, that can be used to grid-enable existing CSPs. This research has shown that cluster-based grid computing is generally unsuitable for integration with Windows-based end-user applications like the CSPs. Using PRC and EDGC forms of grid computing for CSP-based simulation in industry can not only speed up simulation experimentation, replication, optimization, etc., but it can also maximize the utilization of hardware, software and technical resources within an organization.

Yet another contribution of the research is the identification of specific grid computing middleware, namely BOINC and Condor, which can be used to interface with CSPs. BOINC and Condor are also considered appropriate for use by simulation users since they 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).

Although this research had focused on end-users who were considered experts in modeling and simulation but were not expected to be IT specialists, the CSP-grid integration technology that has been proposed in this work 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, 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. This is an area for future research.

ACKNOWLEDGMENT

In the course of this study we have collaborated with fellow researchers and people in industry, and we would like to express our gratitude to Prof. Eduardo Saliby, Jingri Zhang, Jonathan Berryman, Jon Saville, Rahul Talwalkar and Robert Watson.

REFERENCES

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

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

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

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

Brooks, R., Robinson, S., & Lewis, C. (2001). *Simulation and inventory control (Operational Research Series).* Hampshire, UK: Palgrave.

Burke, S., Campana, S., Peris, A. D., Donno, F., Lorenzo, P. M., Santinelli, R., & Sciaba, A. (2007). *gLite 3 user guide, manuals series.* Document identifier CERN-LCG-GDEIS-722398. Retrieved June 28, 2008, from https://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf

Casanova, H. (2002). Distributed computing research issues in grid computing. *ACM SIGACT News, 33*(3), 50-70.

Chance, D. M. (2004). *Monte Carlo simulation, teaching note 96-03.* Retrieved June 28, 2008, from http://www.bus.lsu.edu/academics/finance/faculty/dchance/Instructional/TN96-03.pdf

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

Condor Version 6.9.1 Manual. (2007). *Platform-specific information on Microsoft Windows, Condor 6.9.2 manual.* Retrieved June 28, 2008, from http://www.cs.wisc.edu/condor/manual/v6.9/6_2Microsoft_Windows.html

EGEE. (2007). *Enabling grids for e-science project*. Retrieved June 28, 2008, from http://www.eu-egee.org/

Eldabi, T, Jahangirian, M, Mustafee, N, Naseer, A., & Stergioulas, L. (2008). Applications of simulation techniques in commerce and defence: A systematic survey. A paper presented at *4th Simulation Workshop (SWO8)* (pp. 275-284). OR Society, UK.

Foster, I. (2005). *A globus primer (draft version).* Retrieved June 2008, from http://www.globus.org/toolkit/docs/4.0/key/

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

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

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

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

Globus Alliance. (2005). *GT4 administration guide*. Retrieved June 28, 2008, from http://www.globus.org/toolkit/docs/4.0/admin/docbook/index.html

Herzog, T. N., & Lord, G. (2002). *Applications of Monte Carlo methods to finance and insurance*. Winstead, Conn: ACTEX Publications. Retrieved June 28, 2008, from http://books.google.com/

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

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.

Levine, D., & Wirt, M. (2004). Interactivity with scalability: infrastructure for multiplayer games. In I. Foster, & C. Kesselman (Eds.), *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, chapter 13. San Francisco, CA: Morgan Kaufmann.

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

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

Marins, J. T. M., Santos J. F., & Saliby, E. (2004). Variance reduction techniques applied to Monte Carlo simulation of Asian calls. *2004 Business Association of Latin American Studies (BALAS) Conference*. Business Association of Latin American Studies.

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

Mustafee, N., & Taylor, S. J. E. (2008). Investigating grid computing technologies for use with commercial simulation packages. Paper presented at *2008 Operational Research Society Simulation Workshop (SW08)* (pp. 297-307). OR Society, UK.

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

Pidd, M. (2004). *Computer simulation in management science (5th edition)*. Chichester, UK: John Wiley & Sons.

Reed, D. A. (2003). Grids, the teragrid and beyond. *IEEE Computer, 36*(1), 62-68.

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

Robinson, S. (2005b). Distributed simulation and simulation practice. *Simulation, 81*(5), 5-13.

Shannon, R. E. (1998). Introduction to the art and science of simulation. *30th Winter Simulation*

Conference (pp. 7-14). Los Alamitos, CA: IEEE Computer Society Press.

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. Retrieved June 28, 2008, from http://www.lionhrtpub.com/orms/orms-12-05/frsurvey.html

Swain J. J. (2007). INFORMS simulation software survey. *OR/MS Today*. Institute for Operations Research and the Management Sciences (INFORMS), USA. Retrieved June 28, 2008, from http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation.html

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

Virtual Data Toolkit. (2007). *What is in VDT 1.6.1 (supporting platforms)?* Retrieved June 28, 2008, from http://vdt.cs.wisc.edu/releases/1.6.1/contents.html

Yang, X., Chohan, D., Wang, X. D., & Allan, R. (2005). A web portal for the national grid service. *2005 UK e-Science All Hands Meeting*, (pp. 1156-1162). Retrieved June 28, 2008, from http://epubs.cclrc.ac.uk/bitstream/1084/paper05C.pdf

Zhang, J., Mustafee, N., Saville, J., & Taylor, S. J. E. (2007). Integrating BOINC with Microsoft Excel: a case study. *29th Information Technology Interfaces Conference* (pp. 733-738). Washington, DC, USA: IEEE Computer Society.