Curriculum development on grid computing

Maozhen Li, Marios Hadjinicolaou

Abstract**—** The computational grid is rapidly evolving into a large-scale computing infrastructure that facilitates scientists and engineers to solve data and computationally intensive problems by utilizing various resources over the Internet. This paper presents the curriculum design of a one-year taught MSc course in Distributed Computing Systems Engineering currently running at Brunel University in the Unite Kingdom. It reports the design rationale and practice in introducing grid computing to the MSc course. A case study is given to demonstrate how grid computing can be used to speed up the process in solving data and computationally intensive problems.

Keywords**—** curriculum development, distributed systems, grid computing, service-oriented computing.

I. Introduction

THE Electronic and Computer Engineering (ECE) in the School of Engineering and Design at Brunel University has been successfully running a Masters course in Distributed Computing Systems Engineering for a over 10 years. This course currently runs off-campus in Esslingen, Germany. It will be running on-campus at Brunel University in Sept. 2008. Grid computing [1] is emerging as an effective computing paradigm for sharing various resources over the Internet. The computational grid is rapidly evolved into a large-scale computing infrastructure for scientists and engineers to solve data and computationally intensive problems. The past few years have witnessed tremendous development and deployment of grid systems and applications. Representative grid networks and projects being carried out include EGEE (enabling e-Science for Europe, http://public.eu-egee.org/) spanning more than 30 countries with over 150 sites to a myriad of applications, UK e-Science programme facilitating one national e-Science center, and a number of regional e-Science centers, US OSG (open science grid, http://www.opensciencegrid.org/), CNGrid (China National Grid, http://www.cngrid.org).

The ECE in the School of Engineering and Design at Brunel University has strong expertise in grid computing. We have been actively participating in grid computing projects including the EC funded Data Grid (http://eu-datagrid.web.cern.ch/eu-datagrid), GridCC (http://www.gridcc.org), UK PPARC funded GridPP (http://www.gridpp.ac.uk/). The Brunel Information Technology Laboratory (BITLab) in the School of Engineering and Design at Brunel University has various facilities for grid computing, including 3D imaging, a 128-processor computing cluster for science and engineering computations, a 5 Terabyte data store, 3D scanning equipment, a display system for virtual environment visualisation, hardware rendering equipment, a 6 camera motion capture suite, and a high-speed network link to collaborators in the e-Science consortium. BITLab is part of a London based, geographically distributed Grid Tier 2 computing center. BitLab is also a computing node of the EGEE grid network.

Compared with traditional distributed systems, grid systems are larger in sizes crossing over multiple institutions or organizations, more heterogeneous in resources that are dealt with, and more dynamic in computing capacities. In this paper, we present the educational rationale of introducing grid computing to the Masters course in Distributed Computing Systems Engineering.

The reminder of the paper is organized as follows. Section II introduces the aims and objectives of the course. Section III describes the structure of the course. Section IV discusses teaching and learning strategies. Section V presents a case study demonstrating how grid computing can be used to speed up the computation process in rendering computer animation frames. Section VI gives a brief review of current MSc courses in grid computing in the UK, and Section VII concludes the paper.

II. Course Aims and Objectives

In the process of developing this programme, guidelines and boundaries set by the following references were used:

- Brunel University Learning & Teaching Strategy.
- Brunel University Mission Statement.
- Brunel University Strategic Plan 2002-2007.
- QAA Framework for High Education Qualifications in England, Wales, and Northern Ireland [2].
- QAA Benchmark “Annex to Academic Standards – Engineering” [3].

Specifically, the emphasis on theoretical aspects to very practical problems, and other transferable skills, is clearly consistent with Brunel University’s “Mission Statement” to produce high quality graduates that are of use to the community’ and with the objective of Brunel University’s

Maozhen Li is with the School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH, UK (e-mail: Maozhen.Li@brunel.ac.uk).

Marios Hadjinicolaou is with the School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH, UK (e-mail: Marios.Hadjinicolaou@brunel.ac.uk). Manuscript received January 3, 2008
Revised manuscript received August 8, 2008

"Learning and Teaching Strategy" to ensure that our graduates are flexible and able to meet the changing needs of the world of work. The opportunities provided through this programme to tackle significant practical problems, in particular through workshops and implementation projects, is in the spirit of the Strategy's emphasis on providing student projects that are "problem focussed to simulate or reflect work and community related activity".

The aim of this programme is to equip high quality and ambitious graduates with the necessary advanced technical and professional skills for an enhanced career either in industry or leading edge research in the areas of distributed systems and Grid computing. Specifically, the main objectives of the programme are:

- To critically appreciate advanced and emerging technologies for developing grid systems;
- To practically examine the development of large-scale grid systems;
- To critically investigate the problems and pitfalls of grid systems in business, commerce, and industry.

The programme aims and learning outcomes, and design have been selected having in mind the QAA Framework for Higher Education Qualifications in England, Wales, and Northern Ireland, in particular the Descriptor for qualifications at Masters level.

These aims and learning outcomes are fully consistent with the Masters Level Descriptor of the QAA Qualifications Framework, in particular their emphasis on "systematic understanding of knowledge", "critical awareness of current problems", abilities to "demonstrate self-direction and originality in tackling and solving problems and act autonomously in implementing tasks at a professional or equivalent level" and "continue to advance their knowledge and understanding, and to develop new skills to a high level".

III. Course Structure

The course comprises 8 taught modules and an MSc dissertation:

- Computer Networks
- Distributed Systems
- Network Security and Data Encryption
- Network Computing
- Grid Middleware Technologies
- Grid System Analysis and Design
- Workshop
- Project Management
- MSc Dissertation

Each of the 8 modules carries 15 credits and the dissertation has 60 credits. Students receiving a Masters degree of the course need to achieve 180 credits. Fig.1 shows the correlation of these modules.

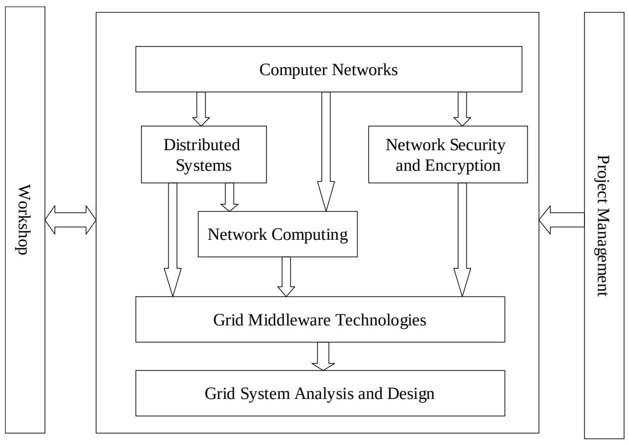

Workshop
Computer Networks
Distributed Systems
Network Security and Encryption
Network Computing
Grid Middleware Technologies
Grid System Analysis and Design
Project Management

Fig.1 The course structure.

A. Computer Networks

Computer Network module is the fundamental module for this course. Grid computing demands quality-of-service (QoS) supported networks especially when transmitting large amount of data over the Internet. For this purpose, the focus of Computer Network is on QoS aspects of IP networks. Topics covered in this module include:

- Network Basics
  - ISO/OSI Reference Model and TCP/IP Reference Model
  - Network Topologies (Start, Ring, Bus, Mesh, Tree)
  - Network Types (LANs and WANs)
  - Communication Control Methods (error detection and recovery, data flow control, connection management)
- Local Area Networks (LANs)
  - Standard IEEE802
  - CSMA/CD-Bus 802.3 (Ethernet)
  - Token Ring 802.5
  - Token-Bus 802.4
  - FDDI ANSI X3T9.5
  - Logical Link Control Protocol LLC802.2
- Wide Area Networks (WANs)
  - Asynchronous Transfer Mode (ATM)
  - Inter-networking (bridges and routers)
- Network Layer
  - Internet Protocol (IPv4/IPv6)
  - Packet Scheduling and Delay
  - IP QoS (RSVP, Integrated Service Model and Differentiated Service Model), MPLS, IP over ATM
  - Address Resolution Protocol (ARP)
  - Internet Control Message Protocol (ICMPv4/ICMPv6)
  - Routing Algorithms (static and dynamic algorithms)
  - Routing Protocols (RIP, OSPF, BGP)
  - IP Multicasting

- Transport Layer
  - TCP/UDP
  - OSI Transport Protocol
- Network Applications and Services

  HTTP, FTP, DNS, NFS, Email (SMTP, POP, IAMP), Telnet, SNMP

B. *Distributed Systems*

This module mainly focuses on issues related to traditional distributed systems such as file systems, naming services, replication services.

C. *Network Security and Encryption*

This module introduces the fundamental theory on network security and encryption with a focus on Public Key Cryptography which is the basis of the Grid Security Infrastructure [4].

D. *Network Computing*

This module covers object oriented programming with Java programming language with a focus on middleware technologies. Topics covered include:

- Object Oriented Design and Programming in Java
- Java Exception Handling
- Java Files and Streams
- Java Multithreading
- Java Collections and Algorithms – Lists, Stacks, Queues, Trees
- Java Database (JDBC)
- Computing Models including Client/Server Model and Peer to Peer Model
- Socket Programming
- Remote Procedure Call (RPC)
- Java Remote Method Invocation (RMI)
- Common Object Requestor Broker Architecture (CORBA)
- Web Computing Technologies (HTTP, Java Applet/Servlet, JSP)
- XML technologies (DTD, XML Schema, XML Parsing Models such as DOM and SAX)
- Web Services (WSDL, SOAP, UDDI)

The material on Web services provides direct support for the module of Grid Middleware Technologies as the computational grid is rapidly evolved into a service-oriented computing infrastructure [5].

E. *Grid Middleware Technologies*

Grid computing is evolved from parallel computing, cluster computing, meta-computing, then service-oriented computing, so Grid Middleware Technologies module mainly covers MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) for parallel applications, Condor (a high throughput batch system, http://www.cs.wisc.edu/condor/) for cluster computing systems, Globus (http://www.globus.org) for grid middleware technologies, OGSA (open grid services architecture) [6], a standard architecture for developing service-oriented grid systems. Job scheduling, which is defined as a process of mapping jobs to resources, plays a crucial role in a Grid environment. A scheduling system has the responsibility of selecting resources and scheduling jobs in such a way that the user and application requirements are met. Scheduling algorithms such as min-min, max-min, greedy, scheduling jobs with genetic algorithms are also covered in this module. It is worth noting that this module was designed based on a grid textbook [7] which provides a systematic way in introducing grid core technologies.

F. *Grid System Analysis and Design*

This module is built on the understanding of grid core technologies. This module helps students build knowledge on the way a grid system works, the current practices of grid applications and systems such as EGEE and GridPP, and issues in the design of grid systems such as scalability, reliability, testing, maintenance. Topics covered by this module include

- System Analysis Methodologies
  - Overview
  - UML
- Construction of Models using Appropriate Techniques
  - Process Modelling, Static Class Modelling, Dynamic Modelling, Interface Modelling
- Management of Large-Scale Grid System
  - Grid Portal
  - Concurrent Version System (CVS)/Wiki
- Grid System Analysis: Case Study (GridPP)
  - The Problem Domain
  - An Anatomy of GridPP System
    - System Components
    - Management of Virtual Organisations
    - User Certificate Management
    - Management of Grid Sites
- Grid System Analysis: Case Study (LCG/EGEE)
  - The Problem Domain
  - An Anatomy of LCG/EGEE System
    - System Components
    - Management of Virtual Organisations
    - User Certificate Management
    - Management of Grid Sites
- System Design
  - User/Problem Requirement Analysis and Specification
  - Object Oriented Design
    - UML
  - System Architecture
    - Performance Consideration
    - Open Standards
  - Design for Usability

G. *Project Management*

This module helps students understand the critical issues in management of projects and build their knowledge in the application of approaches.

Hands-on laboratory assignments associated with each of the taught modules are organised in the Workshop module to help student build knowledge and capabilities in problem solving.

IV. Strategies for Learning and Assessment

During the 1980's the focus on teaching and learning in higher education was on the teaching part, i.e. how the lecturer organized and managed learning activities [8]. During the 1990's the focus gradually shifted from teaching to learning [9]. Learning is a process that involves mastering abstract principles, understanding proofs, remembering factual information, acquiring methods, techniques and approaches, recognition, reasoning, debating ideas, or developing behaviour appropriate to specific situations [10].

Extensive researches have been carried out to classify the high quality learning styles and the low quality learning styles, e.g., meaningful learning vs. rote learning [11], logical forming vs. mnemonic concrete [12], generative processing vs. reproductive processing [13], deep learning vs. surface learning [14], transformational learning vs. reproductive learning [15], holistic learning vs. atomistic learning [16].

Learning outcomes in terms of knowledge and understanding are achieved through a mix of lectures, workshops, seminars, self-study, and individual and group project work. In lectures key concepts and ideas are introduced, definitions are stated, results and techniques are explained, and immediate queries discussed. Seminars provide students with the opportunity to raise at greater length issues arising from the lectures and from private study, for the lecturer to test student understanding through discussion of relevant problems. Workshops and projects are used to foster practical engagement with the taught material. The dissertation plays a key role in deepening understanding, in developing research and literature review skills, and in applying knowledge and skills gained in the programme to plan, execute, and evaluate a significant investigation into a current problem area related to distributed systems and grid computing.

A key part of the strategy for learning has been to provide a solid experiential learning platform based on the Kolb learning cycle [17], and by using small groups [18]. The strength of this approach is clearly the tutorial style with students able to progress at their own pace with a structured work plan to facilitate learning. We also promote student-oriented learning [19, 21, 22]. Collaborative group work and peer review prove effective and useful. In addition, case studies are used in lectures to facilitate students to build pragmatic capabilities in problem solving.

This course employs a range of assessment methods to promote a rounded, diverse and independent approach to learning by the student. Summative assessment (examination) features heavily in the more theoretical modules while formative assessment (e.g. report writing, oral presentation, group project work) features heavily in the workshops.

V. A Grid Computing Case Study

As mentioned in [20] practical work is important for computing lectures. Case studies are used in lecture modules to help students understand lecture materials. In this section, we present a case study on rendering computer animation frames using a grid computing environment managed by Sun Grid Engine (http://gridengine.sunsource.net/). This case study was used to demonstrate how grid computing can be used to solve data and computationally intensive problems.

A. *Sun Grid Engine (SGE) Architecture*

Hosts (machines or nodes) in SGE are classified into four categories, master, submission, execution, administration, and shadow. Fig.2 shows the SGE architecture.

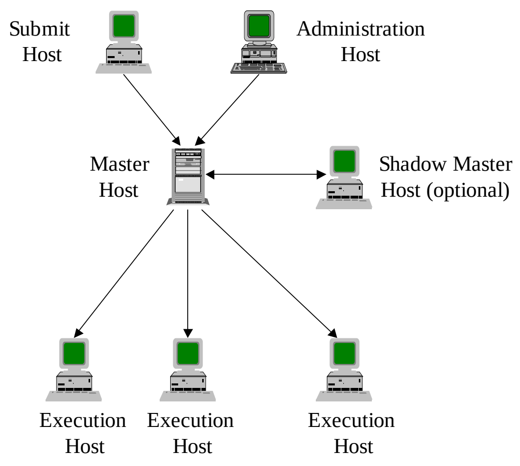


Fig.2 The SGE architecture.

- *Master Host.* A single host is selected to be the SGE master host. This host handles all requests from users, makes job-scheduling decisions, and dispatches jobs to execution hosts.
- *Submit Host.* Submit hosts are machines configured to submit, monitor, and administer jobs, and to manage the entire cluster.
- *Execution Host.* Execution hosts have the permission to run SGE jobs.
- *Administration Host.* SGE administrators use administration hosts to make changes to the cluster's configuration, such as changing distributed resource management parameters, configuring new nodes, or adding or changing users.
- Shadow Master Host. While there is only one master host, other machines in the cluster can be designated as shadow master hosts to provide greater

availability. A shadow master host continually monitors the master host, and automatically and transparently assumes control in the event that the master host fails. Jobs already in the cluster are not affected by a master host failure.

1) The Daemons in a SGE Cluster

As shown in Fig.3, to configure a SGE cluster, the following daemons need to be started.

- sge_qmaster – the Master daemon

The *sge_qmaster* daemon is the centre of the cluster's management and scheduling activities; it maintains tables about hosts, queues, jobs, system load, and user permissions. It receives scheduling decisions from *sge_schedd* daemon and requests actions from *sge_execd* daemon on the appropriate execution host(s). The *sge_qmaster* daemon runs on the master host.

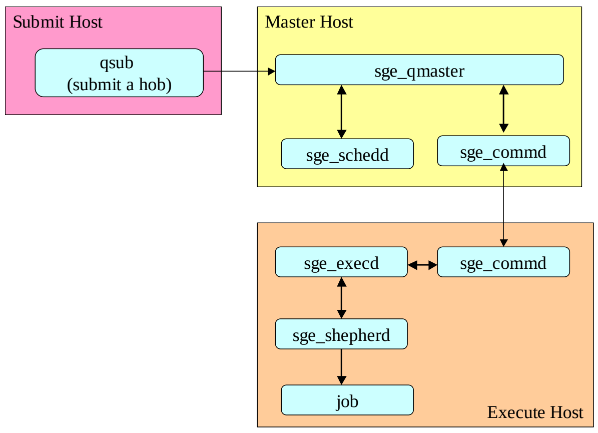


Fig.3 The daemons in SGE.

- sge_schedd – the Scheduler daemon

The *sge_sched* is a scheduling daemon that maintains an up-to-date view of the cluster's status with the help of *sge_qmaster* daemon. It makes the scheduling decision about which job(s) are dispatched to which queue(s). It then forwards these decisions to the *sge_qmaster* daemon, which initiates the requisite actions. The *sge_schedd* daemon also runs on the Master host.

- sge_execd – the Execution daemon

The *sge_execd* daemon is responsible for the queue(s) on its host and for the execution of jobs in these queues by starting *sge_shepherd* daemons. Periodically, it forwards information, such as job status or load on its host, to the *sge_qmaster* daemon. The *sge_execd* daemon runs on an Execute host.

- sge_commd – the Communication daemon

The *sge_commd* daemon communicates over a well-known TCP port and is used for all communication among SGE components. The *sge_commd* daemon runs on each Execute host and the Master host in a SGE cluster.

- sge_shepherd – the Job Control daemon

Started by the *sge_execd* daemon, the *sge_shepherd* daemon runs for each job being actually executed on a host. The *sge_shepherd* daemon controls the job's process hierarchy and collects accounting data after the job has completed.

2) Job Management in SGE

SGE supports four job types - batch, interactive, parallel and array. The first three have obvious meanings, the fourth type - array job, is where a single job can be replicated a specified number times, each differing only by its input data set, which is useful for parameter studies.

Submitted jobs are put into job queues. A SGE queue is a container for a class of jobs allowed to execute on a particular host concurrently. A queue determines certain job attributes; for example, whether it may be migrated. Throughout their lifetimes, running jobs are associated with their queues. Association with a queue affects some of the actions that can happen to a job. For example, if a queue is suspended, all the jobs associated with that queue will also be suspended.

In SGE, there is no need to submit jobs directly to a queue. A user only needs to specify the requirement profile of the job (such as memory, operating system, and available software) and SGE will dispatch the job to a suitable queue on a lightly loaded host automatically. If a job is submitted to a particular queue, the job will be bound to this queue and to its host, and thus SGE daemons will be unable to select a lightly loaded or better-suited resource.

3) Job Runtime Environments in SGE

SGE supports three execution modes – batch, interactive and parallel. Batch mode is used to run straightforward sequential programs. In interactive mode, users are given shell access (command line) to some suitable host via, for example X-windows. In a parallel mode, parallel programs using the likes of MPI and PVM are supported.

4) Job Selection and Resource Matching in SGE

Jobs submitted to the Master host in a SGE cluster are held in a spooling area until the scheduler determines that the job is ready to run. SGE matches the available resources to a job's requirements; for example matching the available memory, CPU speed, and available software licenses, which are periodically collected by Execution hosts. The requirements of the jobs may be very different and only certain hosts may be able to provide the corresponding services. Once a resource becomes available for execution of a new job, SGE dispatches the job with the highest priority and matching requirements.

Fundamentally, SGE uses two sets of criteria to schedule jobs – job priorities and equal-share

a) Job Priorities

This criterion concerns the order of the scheduling of different jobs, a *first-in-first-out* (FIFO) rule is applied by default. All *pending* (not yet scheduled) jobs are inserted in a list, with the first submitted job being at the head of the list, followed by the second submitted job, and so on. SGE will attempt to schedule the FIFO queue of jobs. If at least one suitable queue is available, the job will be scheduled. SGE will try to schedule the second job afterwards no matter whether the first has been dispatched or not.

The cluster administrator via a priority value being assigned to job may overrule this order of precedence among the pending jobs. The actual priority value can be displayed by using the *qstat* command (the priority value is contained in the last column of the pending jobs display entitled P). The default priority value that is assigned to a job at submission time is 0. The priority values are positive and negative integers and the pending job list is sorted correspondingly in the order of descending priority values. By assigning a relatively high priority value to a job, it is moved to the top of the pending list. A Job will be given a negative priority value after the job is just submitted. If there are several jobs with the same priority value, the FIFO rule is applied to these jobs.

b) Equal-Share-Scheduling

The FIFO rule sometimes leads to problems, especially when users tend to submit a series of jobs at almost the same time (e.g., via a shell-script issuing a series of job submissions). All the jobs that are submitted in this case will be designated to the same group of queues will have to potentially wait a very long time before executing. *Equal-share-scheduling* avoids this problem by sorting the jobs of a user already owning an executing job to the end of the precedence list. The sorting is performed only among jobs within the same priority value category. Equal-share-scheduling is activated if the SGE scheduler configuration entry *user_sort* switch is set to TRUE.

B. Rendering Computer Animation Frames using SGE

q3D [23] is a computer animation rendering application written in C that can render 3D-like frames using either 2D geometric shapes or raster images as input primitives which are organized in layers called *cels*. *q3D* has basic 3D features such as lighting, perspective projection and 3D movements. It can handle hidden-surface elimination (*cel* intersection) when rendering *cels*. Fig.4 shows four frames taken from an animation rendered by *q3D.*

In the animation, the balloon moves gradually approaching the camera and the background becomes darker. Each frame in the animation has two *cels*, a balloon *cel* and a lake *cel*. Each frame is rendered individually from an input file called *stack* that contains the complete description of the frame such as the 3D locations of the *cels* involved. These *stack* files are generated by *makeStacks* from a script that describes the animation such as the camera path, *cels* path and lighting. *makeStacks* is a C program developed for *q3D*. We modeled rendering of animation frames as grid jobs, and a SGE cluster to render up to 500 frames.

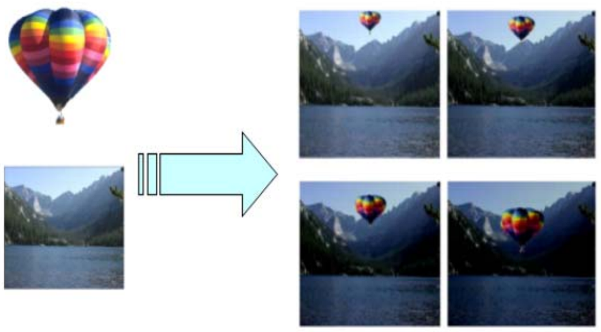
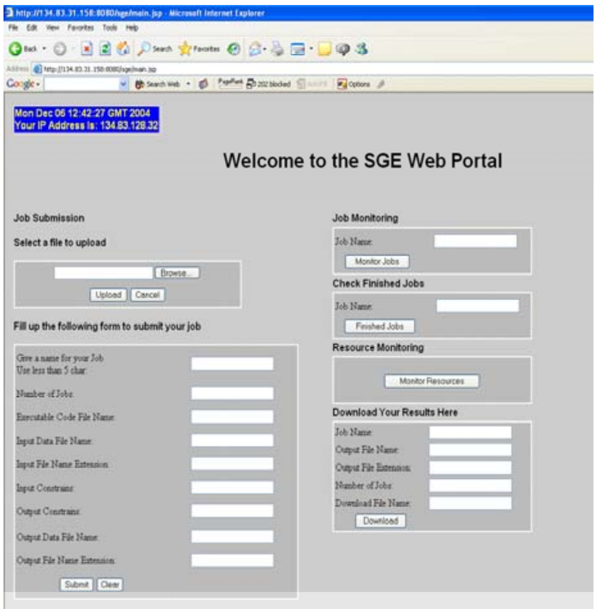


Fig.4 Four frames rendered by q3D with two cels.

A Web user interface was implemented as shown in Fig.5 for submitting jobs to the SGE environment. Jobs once submitted to the SGE can be monitored and results can be downloaded.


http://134.83.31.158:8080/sge/main.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://134.83.31.158:8080/sge/main.jsp
Browse...
Upload
Cancel
Monitor Jobs
Finished Jobs
Monitor Resources
Download
Submit
Clear

Mon Dec 06 12:42:27 GMT 2004
Your IP Address is: 134.83.128.32

Welcome to the SGE Web Portal

Job Submission

Select a file to upload

Fill up the following form to submit your job

Give a name for your Job
Use less than 5 char:
Number of Jobs:
Executable Code File Name:
Input Data File Name:
Input File Name Extension:
Input Constrains:
Output Constrains:
Output Data File Name:
Output File Name Extension:

Job Monitoring
Job Name:

Check Finished Jobs
Job Name:

Resource Monitoring

Download Your Results Here
Job Name:
Output File Name:
Output File Extension:
Number of Jobs:
Download File Name:

Fig.5 A web user interface for SGE.

C. SGE Performance Evaluation

A SGE cluster was built with five computers connected by a 100Mbps local area network. In the SGE cluster, one computer was used as a SGE server. It was also used for job submission and execution; the other four computers were only used as SGE workers for job execution. Each computer is a desktop with a Pentium IV 2.6GHz processor and 512MB RAM, running Redhat Linux. Among the five computers, two of them run Linux 9.0 and the other three run Linux 7.3. The five computers in the SGE cluster shared a network file system. The *q3D* legacy application was only installed on the SGE server.

The latency to render one frame on the SGE server is 30 seconds. We made 8 job submissions to the SGE cluster. We observe that the overhead in sequentially running the *q3D* legacy code on one computer is linearly increased with the number of frames rendered. SGE speeds up the process in rendering image frames, e.g., the latency to render 200 frames in SGE is 4.27 times less than that of sequentially running the *q3D* legacy code on one computer for 200 image frames.

We note that the performance gain of SGE is less than 5 in rendering image frames. This is because SGE incurs overhead in resource management.

According to Amdahl's Law [24], the potential speedup of a computation with parallelization can be defined using equation (1):

$$Computation_{speedup} = \frac{1}{\frac{P}{N} + S} \qquad (1)$$

where

- P is the parallel fraction
- N is number of processors
- S is the serial fraction

Consider a computing environment where 5 computers are involved for solving a domain problem, and P is 100% meaning that the domain problem can be fully decomposed into small independent jobs, and S is 0% meaning that no serial work or communication is involved. Then the maximum speedup in computation is 5 times faster than using 1 computer. This further explains why the SGE environment achieved 4.27 times faster in performance when 5 computers were used in the computation compared with the performance of running the jobs on 1 computer.

VI. EXISTING UK COURSES IN GRID COMPUTING

In 2004, Cranfield University launched an MSc in Grid Computing and e-Engineering, which was thought to have been the first in UK and the second in the world after University of Amsterdam. Six grid-related modules have been designed, which are: Grid Fundamentals, Grid Middleware, Grid Infrastructure, Grid Development and Applications, High-Performance Computing on the Grids, and Nature Inspired Grid Resource Management.

The MSc in Advanced Computing at Imperial College London is a full-time course of 12 months' duration. The programme offers students the opportunity to study a wide variety of topics in depth and with dedicated experts and aims to prepare students for a rewarding career in computing in particular, and in IT in general. The Grid Computing module includes: Service oriented architectures, Web services, programming models for Grid environments, Grid infrastructures (Globus, Condor, Condor-G), Open Grid Services Architecture, security issues, resource election and job placement, computational economics models.

The e-Science and Grid module has been part of the MSc in Advanced Computer Science at the University of Manchester since 2004, The module is also taken by MSc students in Advanced Computer Science and ICT Management. The module is unique in the UK, being based on the Unicore software (http://www.unicore.eu/).

Compared with these existing courses in grid computing, our course provides a more coherent and systematic way in introducing grid computing technologies with a well-designed structure.

VII. CONCLUSION

In this paper, we have presented the curriculum design of a one-year taught MSc in Distributed Computing Systems Engineering course currently running at Brunel University. The MSc course has been running successfully in Esslingen Germany as an off-campus course for over 10 years. We introduced grid computing to the course in 2007 and 24 students are currently on the course. Most of the students on the course are from industry. The feedback from the students is highly positive. We will be running this course on campus at Brunel University in the Unite Kingdom in Sept. 2008.

REFERENCES

[1] I. Foster and C. Kesselman, “The grid, blueprint for a new computing infrastructure,” Morgan Kaufmann, 1998.
[2] QAA Framework for High Education Qualifications in England, Wales, and Northern Ireland, http://www.qaa.ac.uk/academicinfrastructure/FHEQ/EWNI/default.asp
[3] QAA Benchmark “Annex to Academic Standards – Engineering, http://www.qaa.ac.uk/academicinfrastructure/benchmark/masters/MEngintro.asp
[4] Grid Security Infrastructure, http://www.globus.org/security/overview.htm
[5] M. P. Atkinson, D. De Roure, A. N. Dunlop, G. Fox, P. Henderson, A. J. G. Hey, N. W. Paton, S. Newhouse, S. Parastatidis, A. E. Trefethen, P. Watson, J. Webber, “Web service grids: an evolutionary approach,” *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 377-389, 2005.
[6] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke, “Grid services for distributed system integration,” *IEEE Computer*, vol. 35, no. 6, pp. 37-46, 2002.
[7] M. Li and M.A.Baker, “The grid: core technologies,” Wiley, England, 2005.
[8] M. J. Dunkin, and J. Barnes, J, “Research on teaching in higher education,” in M. C. Wittrock (Ed.) Handbook of Research on Teaching**,** 3rd edition**.** New York: Macmillan. pp 754-777, 1986.
[9] P. Ramsden, “Learning to teach in higher education,” London: Routledge, 1992.
[10] H. Fry, S. Ketteridge, and S. Marshall, “Understanding student learning,” in Fry, H., Ketteridge, S. And Marshall, S. (Ed.) A Handbook for Teaching and Learning in Higher Education: Enhancing Academic Practice, 2nd edition. London & New York: RoutledgeFalmer. pp 9-25, 2004.
[11] D. P. Ausubel, J.D.Novak, and H. Hanesian, “Educational psychology: a cognitive view,” New York: Holt, Rinehart & Winston, 1968.

[12] R. Goldman and R. Warren, “Configuration in discriminate space: a heuristic approach to study techniques,” *In Proc. of Western Psychological Association*, Portland, 1972.
[13] F. Marton and R. Saljo, “On qualitative differences in learning: outcome and process,” *British Journal of Educational Psychology*, Vol. 46 pp 4-11, 1976.
[14] M.C. Wittrock, “Students' thought processes,” In Wittrock M.C. (Ed.) Handbook of Research on Teaching. Third Edition. New York: Macmillan, 1986.
[15] P.R.Thomas and J.D. Bain, “contextual dependence of learning approaches: the effects of assessments,” *Human Learning*, vol. 3 pp 227-240, 1984.
[16] L. Svensson, “On qualitative differences in learning: III - study skill and learning,” *British Journal of Educational Psychology*, Vol. 47 pp 233-243, 1977.
[17] D.A. Kolb, “Experiential learning: experience at the source of learning and development,” Prentice-Hall, Englewood Cliffs, NJ, 1984.
[18] S. Brown, “The art of teaching small groups,” New Academic, Spring 1997, pp3-6.
[19] R.M. Felder, and R. Brent, “Effective strategies for cooperative learning,” *J. Cooperation & Collaboration in College Teaching*, vol. 10, no. 2, pp. 63–69, 2001.
[20] Z. Mahmood, “Provision of computing education in Pakistan: an experience report,” *International Journal of Education and Information Technology*, vol. 1, no. 1, pp. 157-160, 2007. NAUN Press.
[21] Z. Mahmood, “A framework for software engineering education: a group projects approach,” *International Journal of Education and Information Technology*, vol. 1, no. 1, pp. 153-156, 2007. NAUN Press.
[22] S. Campanella, G. Dimauro, A. Ferrante, D. Impedovo, S. Impedovo , M. G. Lucchese, R. Modugno, G. Pirlo, L. Sarcinella, E. Stasolla, C. A. Trullo, “Engineering e-learning surveys: a new approach,” *International Journal of Education and Information Technology*, vol. 1, no.1, pp.127-135, 2007. NAUN Press.
[23] M. Qi, and P. Willis, “Quasi3D cel based animation”, *Proc. of Vision, Video and Graphics 2003 (VVG03)*, Bath, UK, 2003.
[24] G. Amdahl, “Validity of the single-processor approach to achieving large scale computing capabilities,” *AFIPS Conference Proceedings*, vol. 30, AFIPS Press, 1967, pp. 483–485.

Maozhen Li is a Lecturer in the School of Engineering and Design at Brunel University, UK. He received the PhD in 1997 from Institute of Software, Chinese Academy of Sciences, Beijing. He joined Brunel University as a full-time lecturer in 2002. His research interests are in the areas of grid computing, distributed problem-solving environments for large-scale simulations, intelligent systems, service-oriented computing, semantic web. He has over 50 scientific publications in these areas including journals of IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Neural Networks, IEEE Distributed Systems. He authored “*The Grid: Core Technologies*”, a well-recognized textbook on grid computing which was published by Wiley in 2005. This book introduces grid computing in a systematic way. He has served the Technical Programme Committee of over 30 conferences. He is on the editorial boards of Encyclopedia of Grid Computing Technologies and Applications, and the International Journal of Grid and High Performance Computing (IGI). He is the course director for MSc in Distributed Computing Systems Engineering. He is a member of IEEE.

Marios Hadjinicolaou received the BSc (honours) degree in Electronics from the University of London in 1979 and the M.Sc. and Ph.D. degrees in Electronic and Electrical Engineering from Brunel University, U.K., in 1982 and 1986 respectively. He is a Senior Lecturer at the Department of Electronic and Computer Engineering at Brunel University engaged in teaching wireless communications. Also, he is a Member of the Centre for Media Communications Research at Brunel University. His research interests are in the field of multiple access systems, video-on-demand, telemedicine, teletraffic engineering and QoS studies for multimedia applications. He has published more than 50 papers in refereed journals and conferences. He is a regular reviewer for IEEE/IET journals. He is a Chartered Engineer, Member of the IET and Member of the IEEE.