A framework for efficient cluster computing services in a collaborative university environment

Robert Laughlin Warrender

A doctoral report submitted in partial fulfilment of the requirements of the University of Sunderland for the degree of Professional Doctorate

Supervisors: Professor John Tindle and Dr David A Nelson

October 2015

Declaration by researcher

I hereby certify that this material which I now submit for assessment on the programme of study leading to the award of Professional Doctorate is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

SIGNED: _____

PRINT NAME: _____

DATE: _____

Abstract

Parallel computing techniques have become more important especially now that we have effectively reached the limit on individual processor speeds due to unacceptable levels of heat generation. Multi-core processors are already the norm and will continue to rise in terms of number of cores in the near future. However clusters of machines remain the next major step up in system performance effectively allowing vast numbers of cores to be devoted to any given problem.

It is in that context that this Professional Doctorate thesis and Portfolio exists. Most parallel or cluster based software is custom built for an application using techniques such as OpenMP or MPI. But what if the capability of writing such software does not exist, what if the very act of writing a new piece of software compromises the integrity of an industry standard piece of software currently being used in a research project?

The first outcome was to explore how grid/cluster computing teaching and learning facilities could be made accessible to students and teaching staff alike within the Department of Computing, Engineering & Technology in order to enhance the student experience. This was achieved through the development of VCNet, a virtual technology cluster solution, based on the design of the University of Sunderland Cluster Computer (USCC) and capable of running behind a dual boot arrangement on standard teaching machines.

The second outcome of this Professional Doctorate was to produce a framework for efficient cluster computing services in a collaborative university environment. Although small by national and international standards, the USCC, with its forty machines and 160 cores, packs a mighty punch in computing terms. Through the work of this doctorate, 'supercomputer class' performance has been successfully used in cross-disciplinary research through the development and use of the Application Framework for Computational Chemistry (AFCC).

In addition, I will also discuss the contribution this doctorate has made within the context of my community of practice by enhancing both my teaching and learning contribution as well as cross-disciplinary research and application.

Acknowledgements

Doctoral studies often seem like a lonely furrow to plough, and yet with the backing of the right supervisory team and plenty of encouragement, these studies felt both worthwhile and, dare I say, often pleasurable. Certainly in my case I need to give acknowledgement to many friends and colleagues who over the period have both helped and criticised my fledgling efforts in research as well as my efforts within my community of practice.

Firstly I would like to thank both of my Directors of Study. Most doctorial students only have one Director of Studies, but with the retirement of Professor John Tindle halfway through the process, a second, Dr David Nelson, had to be drafted in to fill the perceived void. As chief architect of the original USCC, John's knowledge of both computer hardware and networking is exceptional and has provided much of the inspiration for carrying on the work he started in designing the USCC. His knowledge of software engineering is also quite exceptional, although it did take a bit of pushing to get him to start programming in C# using the .NET framework. As emeritus professor since his retirement, I still meet up with John to talk through many of the issues concerning the cluster and greatly value his ongoing contribution and encouragement.

David, on the other hand brings a different perspective to the role of Director of Studies. Although also skilled in computer hardware and networking and a useful sounding board for many of my plans and ideas, it is his interest in teaching and learning within the university environment that makes his contribution and encouragement an essential ingredient to this Professional Doctorate.

In addition Dr Lynne Hall has been co-supervisor and provides periodic review and encouragement.

As well as my immediate supervisory team, I especially would like to thank the 'gang of four' who provided early inspiration during the compulsory module phase of the Professional Doctorate; Professor Peter Smith, Dr John Fulton, Dr Gail Sanders and Dr Judith Kuit.

In connection with the collaborative work undertaken with the Department of Pharmacy, Health and Well-Being, I would like to thank both Dr Mark Gray and Dr Peter Dawson for their close support over the last three years, as well as Professor Roz Anderson.

Finally I have to thank my wife and daughter who have had to put up with me during this extended period of study. Through 'dark' days to periods of elation, life cannot be lived in a vacuum and certainly cannot be isolated from the others in the household. With my daughter also studying for her PhD, my wife Denise has had to suffer more than most over the last few years. Her only worry now is what happens when we get back to 'normal'.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AFCC	Application framework for computational chemistry
CCAI	Cisco Certified Academic Instructor
CCM	Compute cluster multiprocessing
CERN	Conseil Européen pour la Recherche Nucléaire
COTS	Commodity off the shelf
CPU	Central processing unit
DC	Domain controller
DCET	Department of Computing, Engineering and Technology
DPHW	Department of Pharmacy, Health and Well-being
DHCP	Dynamic host configuration protocol
EC2	Amazon elastic compute cloud
EDP	Electronic Data Processing
EPCC	Edinburgh Parallel Computing Centre
GUI	Graphical user interface
HN	Head-node
HPC	High performance computing
IDE	Integrated development environment
JSP	Java server pages
LCD	Liquid Crystal Display
LED	Light Emitting Diodes
MCT	Microsoft Certified Trainer
MM	Molecular mechanics
MPI	Message passing interface
MSDN	Microsoft Developer Network
NTI	New Technology Institute

OpenMP	Open Multi-Processing
PC	Personal computer
QM	Quantum mechanics
RAL	Rutherford Appleton Laboratories, Oxford
RAM	Random access memory
RDP	Remote desktop protocol
SeIUCCR	Supporting e-Infrastructure Uptake through Community Champions for Research
SGE	Sun Grid Engine – now known as the Oracle Grid Engine
SMP	Symmetric multiprocessing
Socitm	Society of Information Technology Management
SSD	Solid State Drive
STEM	Science, technology, engineering and mathematics
STFC	Science & Technology Facilities Council
TCP/IP	Transmission Control Protocol / Internet Protocol
USB	Universal Serial Bus
USCC	University of Sunderland cluster computer
VCNet	Virtual Cluster Network
VHD	Virtual hard drive
VM	Virtual machine
VPN	Virtual private network

1 INTRODUCTION

According to Tennant (2004), a professional doctorate combines both knowledge and research skills to advance or enhance professional practice. It is this link to professional practice that, according to Lee (2009), has made this a very popular choice for career professionals in areas such as "*education, engineering, health and social care, business, marketing, art and design, musical arts and clinical psychology*".

The professional doctorate calls for two key pieces of work:

a) A report
b) A portfolio of evidence

The object of the report is to "*demonstrate advanced and systematic knowledge and skills in the candidate's chosen area*" (UoS, 2009a), while the portfolio shows how this "*forms a contribution to the creation and interpretation of new knowledge at the time of submission and must be set in the context of understanding of the field*" (UoS, 2009a).

The contributions made in this doctoral thesis are:

1. the pedagogical work done in providing a novel platform that can be used in the teaching of high performance computing to enhance the student experience and understanding of cluster computing architectures;
2. the contribution to my own professional practice of internationally recognised cross-disciplinary research through the creation of an application framework for computational chemistry providing an efficient means of preparing batches of jobs to run under high performance conditions on the University Cluster Computer.

1.1 Overview

During the period from 1985 to 2005, processor manufacturers such as Intel and AMD had largely driven the computing market with ever increasing clock speeds and levels of component integration, resulting in a performance increase of around 10,000 times without any substantial increase in power consumption (Borkar and Chien, 2011; Fuller and Millett, 2011). Since 2005, however, although component integration has continued to develop, clock speeds have all but stalled due to a massive increase in power consumption generated by any clock speed increase (Hoisie and Getov, 2009). This has

resulted in processor manufacturers switching their manufacturing resources to multiple core chips (Leiserson and Mirman, 2008; Fuller and Millett, 2011) rather than faster processors.

While multiple cores greatly benefit multitasking and often prevent 'time-slicing' a processor's activity, this does not mean a job can necessarily be carried out any faster on multi-processor chips, due to the fact that commercial software products are predominately developed for non-parallel environments. In addition to this, making use of these additional cores requires a good knowledge of parallel programming techniques (Gal-On and Levy, 2008), usually by either:

- breaking software activities into parallel branches and running these concurrently in different threads; or
- by running batches of jobs within an 'embarrassingly' parallel environment.

While multitasking has brought immense benefits to the desktop and laptop markets, the demands of the scientific community in areas such as graphics rendering, engineering and the newer bioinformatics, genetics and pharmacy communities has led to the further development of cluster computers to provide the raw computing speed to meet the research demands in these areas (Basili *et al.*, 2008; Shalf *et al.*, 2011).

Around 2008, the University of Sunderland developed a cluster computer comprising of some forty nodes each with twin dual core processors (i.e. 160 cores) (Dell, 2008). Although in world terms this is not huge, it did provide significant computing power that could potentially be used in research projects as well as teaching within the University. As time passed, however, it soon became apparent that without dedicated and skilled professionals working alongside the various research groups, the facility would largely remain underutilised. Cluster computing requires a broad spectrum of skills in areas such as programming, scripting, networking, virtualisation, system administration applied to both Windows and Linux operating systems as well as a detailed knowledge of hardware. These were skills I already possessed (discussed in Chapter 2). Other staff who fitted into this category did exist but were already committed to high teaching loads with no time allocated to get involved in collaborative research with other research groups.

For many computing staff, the cluster was far removed from a standard desktop computer. Whether the staff were intimidated by its complexity or felt they had no real use for a machine of its class we can only speculate but almost without exception, the

cluster was not being used either as a teaching or research facility within the department. The core team, which consisted of the lead architect and a researcher, had introduced Master's students to the cluster specifically in areas such as 3D rendering and client/server programming but the lack of space in the cluster area and need for close supervision restricted class sizes to around 10 students per session.

1.2 Aims of the Professional Doctorate

Previous skills gained as an electronics design engineer specialising in microprocessor based designs, my more recent knowledge of Cisco networking as well as Microsoft and Linux operating systems, coupled with my broad experience of various programming languages, enabled me to contribute towards the successful use of the cluster within the University. I wanted to see the cluster used for both teaching within the department especially as part of a drive to introduce skills into higher education (Rees *et al.*, 2006) as well as a vehicle for collaborative research across the University.

The aims of this Professional Doctorate are two-fold:

1. To provide grid/cluster computing facilities for use by students and teaching staff alike within the Department of Computing, Engineering & Technology to enhance the student experience. Chapters 3 and 4 provide the main contribution to this aim.
2. To produce a framework for efficient cluster computing services in a collaborative university research environment. Frameworks can be used to semi-automate multiple processes within a cluster environment to reduce the time spent by staff carrying a variety of tasks as well as ensuring the cluster runs at its most efficient point. Chapters 5 - 7 provide the main contributions to this aim.

1.3 Methodology

According to Wisker (2007), it is very important to select the appropriate research methodologies to support the work as this is what the dissertation will be based on.

'Positivism' is by far the oldest of the paradigms and linked with the 'Scientific Method' – an expression that implies a domain that is regular and ordered and can be investigated objectively (Popper, 1959). This ties in well with cluster computing where systems are

deterministic, free from malice, bias, race or creed. This also means that the three basic techniques associated with the scientific method also apply, namely:

Reductionism: the ability to break down complex issues to smaller units that can be more easily studied, then re-assembled into a composite hypothesis;

Repeatability: as previously stated, computers are by their nature deterministic. Repeating experiments many times can be a useful tool to expose hidden differences that can form part of any hypothesis;

Refutation: tests can be applied by different researchers on different machines in order to attempt to refute any findings and make the hypothesis strong.

With regard to research strategies used, these primarily break down into the following three categories:

Design and Creation: these according to March & Smith (1995) result in several types of IT products or artefacts being developed within the context of the research.

Experiment: this is closely linked to Design and Creation. Computing is one area where experimentation has few boundaries. Tichy (1998) argues in favour of more experimentation by computer scientists and practitioners.

Action Research: According to Checkland (1981), information systems and computing can be thought of as:

- Framework of ideas;
- Problem solving Methodology;
- An area of application.

As the project is related to an application framework for computational chemistry, this featured strongly.

1.3.1 Methodology adopted for the development of VCNet

The detailed technical development of VCNet has been included in Chapter 4 entitled Development of a Teaching Platform, as well as in conference publications which are included in section 1 of the Portfolio of Evidence. It had to fulfil specific practical design criterion such as be capable of working within the University's teaching environment and not cause any disruption to other classes that regularly used the teaching lab where VCNet had been set up. I also wanted it to closely adopt the look and feel of the main USCC cluster so that students would get a good practical experience of using a cluster.

During development, in terms of methodology, I was able to make full use of both reductionism and repeatability, two of the basic techniques associated with the scientific method. VCNet is a group of six virtual machines working together in harmony within one physical computer. The outer domain controller doubled as the development environment where programs could be developed and debugged. Within this domain controller sat five other machines – a head-node and four compute nodes. Using a series of test programs, I was able to run these either separately or collectively on the various machines ensuring that all aspects of these machines fully met the design criterion of the USCC. Indeed some of these test programs were specifically written and used on the USCC during its construction and demos. This meant, in terms of research strategies adopted, these included both 'Design and Creation' (including a definable artefact) as well as experiment.

1.3.2 Methodology adopted for the Application Framework for Computational Chemistry (AFCC)

The detailed technical development of the AFCC has been included in Chapter 6 as well as in the Portfolio of Evidence [Sections 2 and 3]. This largely follows the same methodology as used in VCNet. However there are many noted differences that should be aired in this regard.

During the early part of the framework development, I relied heavily on experimentation. Indeed one of the published papers (is almost exclusively given over to a series of experiments carried out on some Gaussian09 files in an attempt to maximise the efficiency of running Gaussian09 on the USCC (R. L. Warrender *et al.*, 2013c). This produced our own estimation of speed-up on the main cluster running batches of Gaussian09 jobs (x140) provided we were able to avoid scheduler job 'starvation'. This

figure was also shown to be real by subsequent repeated experiments carried out on several batches.

The resulting framework (AFCC) is a good example of reductionism of tasks leading to minimal operator involvement, thus reducing human error, as well as maximising throughput by breaking down what effectively was a complex issue into a group of smaller tasks that could be run from a batch file.

This also meant, in terms of research strategies adopted, these included both 'Design and Creation' (including a definable artefact) as well as experiment. One further strategy adopted for this part of the research process was that of Ethnography (Oates, 2006) – something that is perhaps a little unusual in terms of computing. The research being carried out was a collaborative venture between two university departments, namely the Department of Pharmacy, Health and Well-being (DPHW) and the Department of Computing, Engineering and Technology (DCET). This gave rise over the course of collaboration to some interesting areas where we needed consultation and understanding.

One specific issue we needed to resolve early was the division of chemistry activities from purely computing activities. While the first version of the AFCC did not face any major difficulties, part of this was the need for myself to understand the specific processes the chemists were trying to achieve and translate these into efficient meaningful framework code. Future versions of AFCC (e.g. Ver. 2 as discussed in Chapter 7) had to be architected in such a way that the computational chemists would take over the role of generating their own software needed to run within the new framework.

The proposed solution to this was interesting and one that was tested out within this doctoral thesis before being relegated to future work planned. Computational Chemists invariably learn their craft using Python scripts. Computer scientists and software engineers invariably use one of the C based languages to carry out their work. What we needed was a clear demarcation between the work of the computational chemists and that of the computational scientist. The solution was in fact to allow the chemists to develop and test their own Python scripts before passing these over to be run on the cluster. These so-called 'silver bullets' would be loaded into a software cartridge providing the necessary interface to run within a C# environment on the USCC thus allowing us full control over the 'intent' without the responsibility of generating the code. This has been tested at a low level but will become a major feature of the future AFCC Ver. 2. Effectively this allows for reductionism and ethnography to coexist side by side.

1.4 Report Structure

This report is divided into the following chapters:

- Chapter 2 describes my own personal journey combining three distinct career paths in electronics design, consultancy and education. It also sets the context of work on the cluster as it existed at the start of my doctorate.

- Chapter 3 provides a contextual review on how clusters are used in other settings for the teaching of high performance computing. It looks primarily at three approaches to this problem: the use of actual clusters in teaching; adopting a virtual approach to concentrate on a number of aspects of cluster computing, such as threading, scheduling and message passing; and finally through the use of multiple Live DVDs within a lab context.

- Chapter 4 discusses the learning processes I had to go through in order to become an effective practitioner in the use of the cluster. I discuss the development of VCNet as an environment both for off-line development as well as teaching, and assess its impact on the student experience.

- Chapter 5 introduces the concept of software frameworks applied to the University of Sunderland Cluster Computer (USCC). Software frameworks can provide a relatively safe and productive working environment for cluster program development.

- Chapter 6 introduces the Application Framework for Computational Chemistry. It highlights the basic design philosophy, its use with Gaussian09 as well as discussing its overall success in terms of work processed over a three year period.

- Chapter 7 continues on the discussion of frameworks highlighting the AFCC as an example of a successful framework which has led to the external funding of a PhD studentship in the area of computational chemistry.

- Chapter 8 discusses and reviews the work accomplished within this professional doctorate. It also looks at future work and new directions that need to be considered to further develop the future of HPC within the university.

Chapters 3 and 4 provide the main contribution to the first aim of this Professional Doctorate related to learning and teaching, while Chapters 5 to 7 focus on the second aim related to collaborative research.

1.5 Portfolio of Evidence Structure

The Portfolio of Evidence is a record of items of professional practice over the time of this doctorate and is divided up into the following sections:

- Section 1 consists of three conference papers related to the work for this professional doctorate.
    - (Robert Warrender *et al.*, 2012) – Virtual Cluster to Enhance Student Experience within a Teaching Environment.
    - (R. L. Warrender *et al.*, 2013a) – Development of a Virtual Cluster.
    - (R. L. Warrender *et al.*, 2013c) – Job Scheduling in a High Performance Computing Environment.

- Section 2 consists of two journal papers related to the work for this professional doctorate.
    - (Tindle *et al.*, 2012) – Application Framework for Computational Chemistry (AFCC) Applied to New Drug Discovery.
    - (R L Warrender *et al.*, 2013b) – Evaluating the use of Virtual Machines in High Performance Clusters.

- Section 3 consists of a book chapter related to the work for this professional doctorate.
    - (Tindle *et al.*, 2014) – Further Development of an Application Framework for Computational Chemistry (AFCC) Applied to New Drug Discovery.

- Section 4 consists of work submitted prior to commencement of this professional doctorate.
    - (Robert L Warrender *et al.*, 2004) – UoS Linux – A Linux LiveCD distribution for use in Higher Education
    - (Nelson *et al.*, 2004) – A Comparison of the Product Model and the Third Manifesto
    - (R L Warrender, 2003) – A textbook called 'Databases'

- Section 5 consists of four presentations which are directly related to the work for this professional doctorate and represents some of the additional tasks undertaken within my own community of practice.
    - British Computer Society (BCS) Newcastle & District Branch
    - Society of Information Technology Management (Socitm)
    - University of Sunderland Annual Lecture given to Level 1 Computer Science students
    - Joint Digital Innovation Research & Health Sciences and Wellbeing Beacons seminar

- Section 6 consists of the User Manual for VCNet for use in a teaching environment. This development pushes the boundaries of a practical virtual computing system, integrating a complete four node cluster plus head-node and domain controller within a single virtual hard drive (VHD).

- Section 7 consists of various certificates appropriate to the work for this professional doctorate. It highlights the broad range of specialist knowledge required in addition to my many years of experience as an electronics design engineer and consultant.

- Section 8 consists of personal statements given by two colleagues providing evidence of the contributions to my 'community of practice' both in teaching and learning in the field of computing as well as computational chemistry research related to pharmaceutical drug discovery.

- Section 9 presents evidence of on-going research work taking forward the work already accomplished within this professional doctorate.

Where appropriate, reference will be made to these portfolio items in the main body of the report using the style [Portfolio – Section X.x].

2 Personal Viewpoint

This chapter introduces the situation at the start of my Professional Doctorate. I have included some details of my personal background spanning, what would be for many, three distinct career paths. I have also included details of the University of Sunderland Cluster Computer (USCC) on which much of my work for the Professional Doctorate is based as well as a discussion on what the cluster was being used for prior to this Professional Doctorate.

2.1 Personal Background

My introduction to computers came as an electronics engineering degree student with a short course in ALGOL on the college mainframe computer. While by today's standards this can only be considered as fairly low level, this was to have a major impact on both my life and career. Data was entered by computer operators using punched paper tapes and its output printed directly to tractor-fed fanfold computer paper. While use was made of the system for analysing laboratory data throughout my college career, it wasn't until the final year project that I was able to make use of the digital computer to model an electronic circuit consisting of various transistors, diodes, capacitors, inductors and resistors. I was able to directly compare results measured for a relatively small electronic circuit (in this case a small amplifier) against a computer simulation of the same circuit carrying out a complete nodal analysis, utilising Ebers-Moll equations (Daniel, 1967), and complex matrix inversion. This all took place at least two years before the appearance of the circuit emulation tool 'SPICE' which is commonly used today (Nagel and Enterprises, 1996).

After graduating, I became an electronics design engineer specialising in digital electronics. It was also around the time (Nov 1971) that Intel produced the world's first microprocessor (Faggin *et al.*, 1996), forever changing the digital landscape and heralding in new low cost personal computers. Soon after this I constructed my own computer system using a variety of shop bought and self-designed parts. To gain experience in the different facets of business, I regularly changed jobs roughly every two years. These included managing a production line in the television component industry as well as setting up a thick film facility, a period in semiconductor sales targeting the major electronic data processing (EDP) markets, sales of military grade electronics into the UK space and defence markets, applications engineering of programmable logic controllers in

energy related industries such as power stations and an oil company. Finally I set up my own design company, designing control and monitoring systems building the systems and writing machine level code. I developed intruder alarm systems and digital communicators for a national burglar alarm company – which included designing a system for the protection of the Scottish Crown Jewels at Edinburgh Castle. I developed remote start-up systems for the testing and running of standby diesel generator systems, using telephone lines as well as microcomputer start-up and monitor systems for a fleet of mobile generators.

Following a technical visit to the Middle East in connection with control systems, I decided to embark on a second career as a consultant specialising in control and communications on a government infrastructure project in Saudi Arabia. My initial plans were to carry out this function for two years, but I was ultimately appointed as Senior Project Manager over a major part of the project. I ended up staying for fourteen years gaining valuable project management experience on a multi-billion dollar project. During this period, I became interested in teaching – something I regularly undertook as part of my work with the client. After gaining an MSc in Computer Based Information Systems via the University of Sunderland's Distance Learning Programme, I came back to the UK and took up the post of Senior Lecturer in the then School of Computing, Engineering and Technology at the University of Sunderland.

My interest in computing during my time as Senior Lecturer at the University has largely followed the early part of my career. I have a good understanding of the 'physical' aspects of computing, such as systems administration, operating systems, hardware and architecture, with less emphasis on aspects of computing such as web, HCI, games and graphics. In a drive to introduce technical skills into higher education (Rees *et al.*, 2006), I became both a Cisco Certified Academic Instructor (CCAI) in networking as well as Microsoft Certified Trainer (MCT) in systems administration helping to establish industry recognised computing skills and qualifications into higher education [Portfolio – Section 7]. Indeed, I was appointed by the School as New Technology Institute (NTI) manager for the Sunderland hub – a government sponsored initiative to incorporate skills into higher education. During the early part of my career as a Senior Lecturer, I joined a group of researchers in databases – attempting to become involved in database research. Indeed I am listed as a contributor to one paper during this period (Nelson *et al.*, 2004) [Portfolio – Section 4.2]. Although my contribution was very small, it was an attempt by

the group to involve me in writing research papers in this area, and my introduction to academic research.

Around this time I also authored a database textbook (R L Warrender, 2003) [Portfolio – Section 4.3]. An opportunity had arisen to be involved in a series of study guides in the computing field. As part of this project, I wrote a book on databases for this series. I also worked with a student on UoSLinux - a Linux LiveCD distribution for use in higher education (Robert L Warrender *et al.*, 2004) [Portfolio – Section 4.1] and presented this at the Linux Technical Conference in Leeds that year. This was the first opportunity I had to present essentially my own work at a national conference. Although the student had done most of the practical work as part of his final year project, I had written up the paper and presented the case at the conference. UoSLinux was based on the Knoppix distribution. Unknown to us before the conference, Knoppix had sent Fabian Franz to the conference – the engineer who had written the Knoppix installation program. After giving an initial presentation based on the paper, we decided to let Fabian and my student give a live presentation of a LiveCD remastering, which they had been working on that morning. The results are still displayed on the Linux 2004 Conference and tutorials website (UKUUG, 2004).

It is also interesting to note that although this is a relatively old paper, research papers in the community can be built on at any time. In this case, a recent publication (Auliansyah and Kusniawati, 2014) has just made reference to this paper proving the value of published work to the community at large. Although a reference in its own right doesn't prove value, it does demonstrate the longevity of such papers.

2.2 USCC Cluster Development

Although I was not involved in the initial USCC design and construction, I was nevertheless very happy to see our university take on the challenge of building a high performance cluster from commodity machines. I attended the inaugural event and remember thinking that I was now at an institution where leading-edge research could well be carried out in an area, where significant growth was expected over the next period.

The Portfolio of Evidence [Section 5] provides an overview of current state in the art of high performance clusters. In comparison, the USCC Cluster is, by world standards, a small system yet is by far the most powerful computing resource available within the

University, primarily designed for advanced scientific calculations. Examples of this type of application, where the USCC had been previously used, include crash test analysis as well as 3D rendering.

It consists of twin head-nodes[1] (one for Windows and the other for Linux) together with forty slave or worker nodes each capable of running in either Windows or Linux under a dual boot arrangement. Each node has twin dual core processors, 8 GB of RAM and 2 TB of local storage. Overall, the nodes can access some 160 processor cores as well as 80 TB of distributed local storage. The head-nodes similarly have dual twin core processors as well as 2 TB of local storage but also have access to a further 10 TB of common central storage. Interconnecting all system nodes is a triple network arrangement (enterprise, application and private) using a high performance Cisco 6509 Ethernet switch, with the whole system housed in a six-cabinet rack, as shown in Figure 1.

Figure 1 - University of Sunderland Cluster Computer (USCC)

Although there had been a core team working on the cluster's developments over a two year period attempting to shape its role within the university, usage of the cluster had declined during its third year. This was due to the sudden illness and premature death of the chief architect's wife. Without his drive and vision, work on the cluster had largely stopped, prompting criticism from many in senior management positions within the faculty to doubt the wisdom of the original investment.

[1] A head-node is a management node used to farm out jobs to the worker or slave nodes on the cluster.

During the earlier period of development of the USCC I had other commitments preventing me from joining the original core group. Within the remit of a Professional Doctorate, I now was in a situation where I could not only help in the USCC development, but could bring other personal skills, particularly my experiences related to systems engineering, project management, systems administration, networking, virtual computing and to some extent programming. My aims were to work in two basic areas, namely:

1) To develop a platform for teaching of cluster related activities within the department.
2) To participate in collaborative research with other research groups who have a significant need for a cluster solution to their problems.

After the chief architect had returned to work, we set about looking for areas of interest around the university which could be solved using the USCC. The USCC needs a particular type of work to be effective. The first requirement is that the programs be capable of running in remote command line mode without any need for user dialogue. Although the local monitor can be set up to display any specific node activity and output, this is only normally used under test conditions. It is important to be able to run each node remotely, driven from either scripts or programs stored in the head-node. I was also looking for processor intensive problems that either take a substantial time to resolve or require many instances to be run in order to utilise the cluster to its capabilities.

2.3 Upgrade to the Cluster Facilities

In January 2011, following the refurbishment of the student terrace area, I decided to embark on a major upgrade to the USCC. During the intervening years, technology had advanced and I wanted to make use of the latest HPC software on the USCC. The 'real estate' area associated with the cluster facility had been substantially reduced to make way for a changed terrace layout to enhance teaching of practical computing subjects. The cluster at this point used the Linux based SCALI cluster management tool as a means of managing the dual boot arrangement between Linux and Windows operating systems. The nodes were set up to use both Scientific Linux as well as Windows 2003 Server. This was 'state of the art' at the time of construction of the cluster in 2007.

My work on virtual machines, and in particular VCNet, confirmed that Windows High Performance Computing (HPC) Server 2008 R2 was a worthy successor to the earlier Windows Compute Cluster Server (CCS) 2003, with many of the bugs resolved for cluster

use. Windows CCS 2003 was Microsoft's first commercial attempt at Cluster management (Gardner, 2006) and broke considerable new ground for them in terms of market credibility in this field. Their 2008 R2 HPC server product came with more tools and facilities to manage a cluster and made script driven interface development easier to use.

As a consequence of the upgrade, I was aware that I would lose SCALI. SCALI had been installed by Dell sub-contracted engineers during the construction of the cluster. Despite frequent requests to obtain the original product disks, these were never supplied. To add to our troubles, the company SCALI A.S. had been sold and their products were at the time unavailable. Although SCALI had been very reliable in its operation, it had been set-up to default to a Linux cluster – not what we generally used. Indeed nearly all of our work was in Windows so booting up of the cluster to Windows had always been a slow manual task.

Microsoft, in the meantime, had been developing a new windows boot arrangement for Windows 2008 and Windows 7. Known as Boot Configuration Data (BCD), this opens up the boot-up process to be more accessible to scripts to control its action. This made it relatively easy to start-up in either Windows directly or branch to the Linux GRUB[2] bootloader for start-up in Linux.

Windows 2008 R2 HPC Server has performed well over the last four years and fully justified my decision to carry out the upgrade. However as with many other situations, we have to move on. Windows 2012 R2 HPC is now a reality and beckons the future for the cluster. Windows 2012 solves many of the problems that Windows 2008 has – primarily related to its ability to upgrade and resize Virtual Hard Drives (VHDs) in-situ, as discussed in Chapter 4.4. Although Windows 2008 claimed this was possible, in reality this involved substantial workarounds to effect a solution. Having this facility opens up the way for the use of VHDs on the cluster making these more versatile to manage and optimise.

The decision (discussed in Chapter 4.2) to adopt Windows HPC and the Sun Grid Engine was purely a pragmatic one. Unlike most other cluster facilities, we do not develop our own software – indeed the very nature of the projects we have worked on demand the use of standard off-the shelf software used elsewhere within the university (such as Gaussian09 and LSDyna). Newer systems such as Hadoop are showing great promise especially in the 'Big Data' context where applications can scale over thousands of nodes.

[2] Linux GRUB loader is one of the most common boot loaders used in Linux, its name taken from the acronym GNU **GR**and **U**nified **B**ootloader.

However this is not the area we are in. Indeed, there is growing evidence that these types of systems are primarily inefficient by nature (Leverich and Kozyrakis, 2010), something we have to take great care over with a small cluster to ensure we get the maximum advantage in terms of performance and efficiency in the area we work.

2.4 Use of Cluster in Research

Previously the core team had worked in the area of 3D graphics rendering, presenting two papers at a conference (Ginty *et al.*, 2009a; Ginty *et al.*, 2009b). While this was of interest to the computing community, there was little research collaboration taking place due to the nature of the software packages being used. 3D graphics packages such as Blender, 3D Studio Max, Lightwave3D and Cinema 4D can be difficult to initially set up on a cluster but are complete in their own right in that they come with all necessary client/server components designed to run across a network on multiple machines. They do, however, require major artistic skills for the creation of storylines, scenes and movies. While complete movies such as Toy Story or Shrek could conceivably be rendered on the USCC, the sheer artistic infrastructure needed for such a project is not a viable option within a university context to produce a commercial product.

Other areas that stood out as being potential targets for use of the cluster were:

i) Engineering (including automotive);
ii) Molecular biology (including chemistry, pharmacy, biochemistry, genetics).

Although computing, engineering and technology had been part of the same University school structure for years, a recent restructuring into faculties had brought together an enlarged Faculty of Applied Sciences to include departments such as computing, engineering, pharmacy and chemistry. This gave the project an added incentive as we were being actively encouraged by senior management to work together within the new faculty structure. Previous barriers related to cost centres now no longer applied being all part of the same administrative cost centre.

The area where we felt the computationally intensive cluster capabilities could most influence our engineering research was in areas such as finite element analysis. This could be, for example, analysis of stresses in structures such as space frames, through to deformation and crash analysis of moving objects such as vehicles. After several discussions with concerned personnel, we found one group who claimed they were

regularly attempting to use LS-Dyna software (LSTC, 2015), both for research and in their teaching. While this was indeed setup and a few jobs run on the cluster, the actual usage turned out to be very low and has not been used for some time.

Around the same time we started speaking to a research group over in the Department of Pharmacy, Health and Well-being (DPHW). Their interest lay in exploring potential new drugs that could target particular cells, resulting in new ways to cure diseases at the cellular or genetic level. The cost of developing new drugs has become so prohibitive (often in excess of several billion dollars) resulting in drug companies primarily targeting areas where they are certain to recover their investment. Traditionally this process can take more than ten years in the production, purification and evaluation of many thousands of compounds before clinical trials can commence making drug advancement an exceedingly slow process (Tindle *et al.*, 2012; Tindle *et al.*, 2014), [Portfolio – Sections 2.1 and 3.1].

In the area of computational chemistry, much of this traditional lab work can be modelled on powerful computers in order to determine the most likely potential candidates for production and testing. This means that a great deal of lab work can be minimised, reducing the overall cost as well as the time taken to bring a drug to the clinical trials stage.

The DPHW group described the work they had achieved to date in the area of computational chemistry. They wanted to use a combination of molecular modelling and quantum mechanics techniques to determine whether a target was worth further investigation. The difficulty they were having was that quantum mechanics work is highly computationally intensive. They wanted to use Gaussian09 (2012a) which can find approximate solutions to the Schrodinger Equation for a group of atoms, and is regarded (within their community) as one of the best software packages in this field. However even fairly small molecular compounds processed on their departmental computer were taking between one and two weeks to process. Their initial goal was to process at least 200 to 300 compounds of varying complexity but clearly that target would result in massive time difficulties.

2.5 Use of Cluster in Teaching

At this time the cluster was being used periodically in teaching certain Network Systems groups at Master's level under the supervision of one of the core team members.

Demonstration programs had been developed mainly in Java to demonstrate the principles of running software on a cluster. The students were encouraged to test out their own solutions to small problems to gain some experience in cluster computing. While this has to be regarded as one of the initial successes of the cluster, it only serves to highlight some of the difficulties in adopting this approach, namely:

- While students are working on the cluster, the cluster cannot sensibly be used to undertake any serious scientific computing work which would require the use of all nodes.
- Conversely, if the cluster is loaded up with some serious scientific computing work, students would not be able to gain access during their normal scheduled teaching time slots.
- In the original design of the cluster area, ten workstations had been provided for access by students or staff. This meant that provision could only be given to small class sizes. If used for scientific computing work, this provision was largely unnecessary and hence took up 'real estate' without any tangible return on the space.
- The other danger of allowing students free and unencumbered access to the cluster was always going to be one of risk mainly to the software integrity of the cluster. While the doors of all cabinets are locked preventing any serious health and safety issues, little can be done to protect the software integrity of the system. In order to program all aspects of the cluster, students need to have full admin rights – something we do not give them on the open terraces. Any serious loss of software integrity could potentially mean a total rebuild of the system, taking around two weeks to accomplish.

2.6 Summary

In this chapter I have set the context of this Professional Doctorate giving an outline of my personal background as well as the development of the University of Sunderland Cluster Computer (USCC). Cluster computing involves many disciplines requiring a unique and detailed knowledge of hardware, software, networking and operating systems, Windows and Linux. In the next chapter I will start to focus on the first aim of this Professional Doctorate by investigating what others are doing in this field particularly with regard to a teaching context.

3 Contextual Review with regard to Teaching HPC

In order to be able to provide a suitable Learning and Teaching environment for our students, it is important to review similar approaches in both the academic and professional fields. Therefore this chapter firstly provides an overview of high performance computing, before discussing some of the different approaches being taken in the teaching of this subject.

3.1 High Performance Computing (HPC)

Parallel computing has been around for many years although the physical architecture has changed substantially over time. With the development of fast Ethernet and low cost commodity computing machines, most so-called 'supercomputers' these days consist of networked clusters of standard or 'commodity' machines rather than a custom built monolithic architecture. The ability to accomplish more than one sub-task at a time is key to reducing the time taken to process complete tasks. While these tasks were tied to slower devices or interfaces, solutions such as 'time-slicing' at the processor level had become popular, effectively allowing the processor to work on several tasks (apparently) simultaneously (Silberschatz *et al.*, 2001). However, with the implementation of physical memory cache solutions, even slower devices could be made to appear as though these were capable of running at speeds not previously thought possible.

While most of these technologies and solutions still remain, the period from the late-seventies through to around 2004 had largely been dominated by the huge performance increase in processor speeds (Fuller and Millett, 2011). This increase was made possible by the shrinking of processor die sizes, reduction in operating voltages as well as the corresponding reduction in overall heat dissipation within the devices. This so-called 'free lunch' situation continued until around 2004 when it was found that increasing the processor speed resulted in quite unacceptable levels of heat dissipation within the devices (Parkhurst *et al.*, 2006; Borkar and Chien, 2011; Fuller and Millett, 2011). During this period, the increase in processor performance was so significant that for most applications this had pushed parallel computing into the background.

Since 2004, there has been an increase in the number of cores diffused within a processor effectively maintaining 'Moore's' law without the corresponding increase in processor speeds (Sodan *et al.*, 2010). While most existing applications do not use multi-cores in a truly parallel situation, low-level symmetric multiprocessing is used to 'load-shed' tasks making sure that multiple tasks do not get handled by one core while another sits idle. This has the effect of allowing applications to run concurrently without the need to forcibly 'time-slice' or wait for a gap in proceedings before it can continue. Even simple netbooks now have multi-core processors and can handle simultaneous applications with ease. For example, you can playback music while 'surfing' the internet or updating your 'Facebook' account. It is likely we have yet to see further advances in processor speeds due to changes in semiconductor materials used for the production of processor cores. Newer materials such as Graphene, Boron Nitride, Molybdenum Disulphide, Silicene and Germanene all show some interesting prospects in this regard (Dean *et al.*, 2010; Schwierz, 2010; Hong Li *et al.*, 2012; Scalise *et al.*, 2012) but require a great deal more research before these become real contenders.

In Amdahl's seminal paper (Amdahl, 1967), he expressed concerns about the use of clusters (or parallel machines) as a way forward to achieving significant performance enhancements due to the existence of sequential components within any given computation. While some regions can be made to operate in a parallel configuration, significant portions (around 40%) were less likely to be "amenable to parallel processing techniques". Although Amdahl did not enumerate his famous 'law' in his paper, this was later taken to be:

$$\boldsymbol{Speedup}\ (\boldsymbol{Sp}) = \mathbf{1}/(\boldsymbol{S} + \frac{\boldsymbol{P}}{\boldsymbol{N}})$$

where 'S' and 'P' represent the serial and parallel components and 'N' the number of nodes.

Consider a 100 core cluster. If the serial component 'S' approaches zero, the parallel component would approach 1. This would make the speedup equal to the number of cores i.e. 100. However, if the average serial component was 40% as suggested by Amdahl, then the speedup would drop to 1/(0.4 + 0.6/N) or 2.463, clearly a huge drop in performance.

Gustafson took a different view in his analysis of the likely speedup achieved by a cluster (Gustafson, 1988) when the serial portion was much less than the parallel portion within a given computation.

$$\boldsymbol{Speedup}\ (\boldsymbol{Sp}) = \boldsymbol{P} - \boldsymbol{S}.(\boldsymbol{P} - \mathbf{1})$$

where 'P' represents the number of processors and 'S' represents the serial fraction of the process.

He considered serial components of between 0.004 and 0.008 (i.e. 0.4% to 0.8%) and gave documented speedups closely approaching the number of installed processors. The increase in performance centred round his argument that the problem can expand to utilise the number of available processors.

Clearly Gustafson was looking at some very specific computations being carried out at his company – it is unlikely that any general purpose software could be written entirely to achieve such high levels of parallel programming without some major shift in current software modelling techniques (Hwu *et al.*, 2008).

3.2 An investigation on how other institutions teach High Performance Computing

In many areas of engineering and life sciences, high performance clusters have become an essential part of modern day research. The ability to model many of the processes within a high performance computer environment not only speeds up development time but also has a huge impact on research costs. Development of new drugs, for example, is one such area where use of high performance computers are paving the way for research into conditions which only a few years ago would have been regarded as wholly uneconomic for private enterprise to even consider investment (Tindle *et al.*, 2012). In order to meet the growing demand for specialists in this area, computing must play its part particularly with regard to teaching and preparing students for life in this new order. Computer science has to work in partnership with many of the areas of engineering, science and mathematics in order to meet future demands for services in these highly specialised areas (Rees *et al.*, 2006). All too often we expect biologists, chemists and engineers to cross the divide into computing forcing them to learn new skills rather than work collaboratively for the benefit of the project.

Teaching of computer clusters is best met by a full hands-on experience. However, according to Albrecht (2009), smaller institutions rarely have such facilities and those institutions that do (typically the larger universities) tend to reserve these for research use rather than be kept for student use within a teaching and learning environment. Clusters are a relatively expensive resource and as such usually have to pay their way within a modern university environment. The following five papers outline the strategies adopted by various universities with regard to the teaching of high performance computing (HPC).

In the paper by Sinnot et al. (2005) the authors describe in some detail how grid computing may be taught to students. A timetable for a twenty week program of lectures is provided with topics based upon the Globus Toolkit, Java web services using Tomcat and scheduling. A similar paper by Farian et al. (2008) was presented on the subject of teaching high-performance computing in the undergraduate college CS curriculum, demonstrating the importance of this topic at an early stage in a student's development.

In a paper by Walters et al. (2006) the MGrid system is described as having three main components entitled: submit, server and process. The system is based upon Java Server Pages (JSP) and Tomcat and the use of Applets. The authors argue that MGrid may be used three different ways in teaching; for demonstrations, algorithm distribution and code distribution. It was found that MGrid worked well in both a lecture/demonstration context but challenging in a student tutorial environment – the paper discusses the challenges in detail. There is also a section dealing with grid security, which would clearly be of importance in teaching grid and cluster computing. MGrid is also used to teach high performance computing to first year undergraduate students, also demonstrating the importance of this topic at an early stage in a student's development.

In a paper by Murshed and Buyya (2002) the GridSim global model is described. This paper considers issues related to resource modelling, Java based object request brokers and job scheduling which they highlight as key topics within a teaching environment for students to have a full understanding of high performance computing.

In a paper by Apon et al. (2004) the authors list four "suggested course components" required to teaching grid or high performance computing. The authors of this paper are employed in four universities based in Arkansas, Monash, Los Angeles and Oregon and have embedded these common components into their respective curricula. The advantages of a grid design based upon COTS (commodity off the shelf) hardware

components are described in some detail. Students are taught how to write parallel programs using, Java, C++ and MPI.

In a paper by Rehr et al. (2011) the authors have applied Amazon Web Services (AWS) and the Amazon Elastic Compute Cloud (EC2) to solve a physics problem, demonstrating the potential for the teaching of cloud computing using clusters.

These papers discuss the topic of high performance computing and outline typical curricula which have been recommended as appropriate for university education in environments both within the UK and worldwide.

There is also a common theme running through these five papers discussed above and that is the application of web services based upon Java running under Linux/Unix. The restrictions implied by this, i.e. that students do not have physical access to the cluster but only access through a managed web interface, and the dependence on Java development within Linux/Unix operating systems, could impose inflexibility of students access to resources. Development of a virtual cluster, e.g. VCNet, would not impose these limitations.

In addition to a conventional teaching model, Microsoft has provided a family of virtual labs available through their main and Technet websites (Microsoft, 2012a; Microsoft, 2012b; Microsoft, 2012c; Microsoft, 2012d). These virtual labs allow users or students to connect to preconfigured virtual machines across a network using a browser. In a learning context, students are able to follow a prescribed path towards a particular targeted outcome by following instructions given in training e-books provided for each learning goal. Students are given full access to a real machine that operates in a virtual and hence safe environment. Students are able to take full control of the virtual machine and do everything to that machine that can be done in the real world. This may include completely corrupting the operating system, because of user inexperience. As the virtual machine has been created for individual use, such usage does not adversely affect the underlying system and a simple reload will reset the virtual machine back to its original operating state. Students are able to use Microsoft Virtual Labs to carry out specific tasks such as scheduling a job in a cluster. The advantages of using virtual labs over conventional labs are well documented in other fields (Ertugrul, 2000; Bates, 2003) although some degree of caution is also recommended in situations where conventional labs are still possible (Scheckler, 2003).

North Carolina State University (NCSU) developed their Virtual Computing Lab (Schaffer *et al.*, 2009) using this basic cloud based system which primarily open-source and was used for IT Education at many universities (Peng Li *et al.*, 2009; Endo *et al.*, 2010). This was later taken up by the Apache Foundation and offered as an open-source solution (Dreher and Vouk, 2013)

In a paper by Thiruvathukal et al. (2010) the authors discuss the advantages associated with the virtual machine (VM) operating system (OS) concept. This paper describes how a virtual cluster was implemented using Ubuntu and Virtualbox. In the paper the authors consider the system design requirements, for example:

- The allocation of available RAM for program execution;
- Head-node (HN) configuration;
- Creation of disk partitions for VMs;
- Disk storage capacity required for the operating systems and user data storage;
- Support for code development using level programming languages;
- Support for MPI Networking including the Dynamic Host Configuration Protocol DHCP.

A wholly alternative and novel approach has been adopted in a paper by Sulistio et al. (2012) where the authors describe the use of a Linux Live DVD to provide the environment necessary to build a working virtual cluster using a group of locally network connected machines creating what the paper terms a "virtual laboratory". Using a set of Torque scripts which are present on the Live DVD allows each machine to configure itself quickly as part of a virtual cluster. A major advantage of this setup is that it can then be taken down at the end of a session to allow the teaching area to be used in its original configuration, which was seen as a major benefit in the development of VCNet (see Section 4.3).

A review of teaching High Performance Computing to both undergraduate faculty and students as well as best practices and pitfalls associated with teaching of Parallel Computing to the Science Faculty is presented in two papers (Joiner *et al.*, 2006; Fitz Gibbon *et al.*, 2010).

In this review I have examined both the academic and professional approaches to enhancing teaching and learning related to high performance computing. The main benefits include:

1) The use of virtual labs as a non-pervasive teaching environment
2) Essentially a no-cost solution which is advantageous to universities in the current economic climate
3) The potential use of VCNet in many subject areas such as:

- Cloud Computing
- Job Scheduling
- Parallel processing with Java, C# and MPI applications
- Network design
- Network security experiments, e.g. port scanning
- Intrusion detection systems (IDS)
- Testing firewalls
- Web based systems
- Server side programming
- Distributed database programming
- Virtual private networks (VPNs) and tunnelling

3.3 Summary

Before I could be useful to other collaborative groups around the University or indeed to use the USCC as a vehicle for teaching, it was important that I first learned how to 'drive' the system and master its usage. I needed to be able to rebuild the system if needed as well as be able to write appropriate software that could be run on the system. In the next chapter I will discuss the construction of a virtual cluster (VCNet) for use in teaching as well as my initial work with the department of Pharmacy, Health and Well Being in the area of computational chemistry.

4 Development of a Teaching Platform

In this chapter I discuss the processes I went through to become familiar with cluster computing and the USCC in particular. This has specific relevance to one of my stated aims, namely that of developing a platform for the teaching of cluster related activities within the department. I then set out the goals that had to be met to achieve this and how VCNet worked out in practice.

4.1 The University Of Sunderland Cluster Computer (USCC)

The USCC was designed as a collaborative facility for use with different research groups around the University. While the writing of custom parallel software is not being ruled out, it is unlikely this could realistically be taken on board for several reasons, including:

- Lack of dedicated programming staff to write such software;
- Lack of staff available to manage and coordinate maintenance of such software

Realistically this type of work needs to be either taken into a commercial situation for on-going exploitation or funded as a specific educational project leaving the artefacts in the public domain when complete. Where we do have a significant role to play is in the use and exploitation of existing software currently being used by the different groups across the university. By running this within a cluster environment, we can in many situations significantly impact the overall speedup of the software to levels unachievable by the various research groups. An example of this was presented in a recent paper (R. L. Warrender *et al.*, 2013c) [Portfolio – Section 1.3] at an international conference in Finland where measured speedup was shown to be around 140 on a 160 core / 40 node cluster. This will be discussed in more detail in Chapter 6.

4.2 Introduction to VCNet

VCNet was originally started as a way of learning about HPC computers without exposing either myself or indeed the cluster, to perilous situations. A cluster relies heavily on the combined resources of many nodes and its head-node working in mutual harmony (along with other system servers). I wanted to understand the physical makeup of a cluster and its software without compromising the real cluster. Later, it became apparent that by taking on meaningful work for the cluster, the cluster availability would drop and therefore couldn't realistically be used (without major disruption) as part of our teaching

facilities. It was something to show the students rather than something that could be used by the students. This was a major blow to my initial goal of helping to introduce cluster computing within the mainstream curriculum – allowing students to add the use of clusters to their CV's.

There are different ways of using cluster computers. Most institutions (such as the Edinburgh Parallel Computing Centre) with large clusters handle jobs to be processed within an access framework allowing the user to specify the resources needed in terms of nodes and processors and RAM for a particular job. It presupposes that the user has written their own software code, requiring only a versatile platform on which to run the code. The job would be fed out to a scheduler which would run the program when resources became available. This gives every user reasonably fair access and limits the 'damage' that can be done to the system. Both the Sun Grid Engine (SGE) (later to be called the Oracle Grid Engine (OGE) before being purchased by UNIVA Corporation) as well as Microsoft's HPC software largely follow this principle although the former is by far the most widely used system within the cluster community making Linux the most popular environment. Microsoft's HPC software operates within a Windows .NET environment.

While this approach is fairly safe and even-handed to all users, it does rely heavily on a large availability of unused nodes, otherwise the head-node scheduler has to queue the work for later processing adding significant latency to the system. Unfortunately, our cluster is not large, even by national standards, and when running batches of quantum mechanics jobs often taking several weeks to clear, job latency becomes an insurmountable issue within a teaching context.

Another issue with regard to the main cluster was giving students administrator access to all elements of the cluster. I was keen to open up use of the cluster to areas such as network design, port scanning, intrusion detection systems (IDS), firewalls, virtual private networks (VPN's) as well as server-side programming and distributed databases. All of these demands require students to have full administrator access to the various operating systems making up the constituent parts of the cluster. While this is good for the students allowing them to effectively learn by their mistakes, it carries immense risk with regard to the on-going viability of the main cluster. Rebuilding a cluster is a major task and not something to be taken lightly. Even worse is the integrity of results that could emanate from a partially corrupted system.

That basically left two directions to take the cluster:

1) To follow most other facilities and opt for the Sun Grid Engine or Microsoft HPC approach, providing a reasonably high degree of isolation between user and the cluster.
2) To follow a much more open approach that gave far more 'intimate' access to the various parts of the cluster.

Few research establishments appeared to be using Microsoft HPC, most opting for the SGE. I had a chance to visit and use the cluster facilities at the Edinburgh Parallel Computing Centre (EPCC) where I attended one of their 2-day OpenMP courses. I also attended a 4-day SelUCCR e-Infrastructure summer school at the Rutherford Appleton Laboratories (RAL) just outside Oxford and was able to speak with many researchers from different HPC facilities as well as vendors such as Microsoft, Amazon EC2. Using the SGE means that institutions could standardise on open source Linux rather than adopting a closed source (and expensive) operating system such as Microsoft Windows.

In many ways, our situation was different from the others:

- We had a relatively small cluster that could be licensed via the Microsoft Developer Network (MSDN) academic alliance licence agreement granting free software licences to academic institutions.
- As an institution, almost all of our software development teaching centred on the .NET framework – typically C# for most courses and C++ for games technology. In all cases Visual Studio was the prime development environment. This meant that any developments would most likely take place in a Windows environment.
- Most of the licenses for developed packages and tools around the university were for the Windows platform e.g. Gaussian09, LSDyna, AutoDesk 3DS Max, AutoDock Vina

This therefore led to a combination of two directions – namely the use of Microsoft HPC as the primary cluster software for communications and scheduling, but run in a way that permitted full access (or intimacy) with all parts of the cluster via a custom software framework developed specifically for the USCC. This framework will be discussed in more detail in chapters 5 and 6.

The USCC is a dual-boot system and can equally be booted into either Windows or Linux environments. For the Linux environment, I decided to utilise the SGE as the means of

driving the cluster. To date, however, I have not had any requests to run Linux software on the USCC but checks have been carried out to ensure support for the SGE if required.

The net conclusion was that we needed to separate students from usage of the main cluster for system integrity issues and find a different solution for use in the teaching environment. This is discussed in more detail in the next section.

4.3 Design Philosophy for Teaching Environment

With the work I had done previously on virtual computing and after review of available literature with regard to teaching and learning, I felt it was time to specify and design a cluster system that could be used within the teaching environment. The basic specification was as follows:

1. A student should get access to his own cluster;
2. The cluster (built using virtual computing) had to be self-contained within an existing software lab machine (standard PC) running Windows 7 (or later);
3. The topology was required to follow that of the main computer cluster to permit software developed to be subsequently run on the main cluster;
4. The cluster to be essentially hidden for normal teaching sessions in the designated lab area making the software lab dual purpose;
5. A script be written to allow the cluster to be enabled (after a simple reboot);
6. Similarly a script be written to return the lab machine back to a normal teaching environment (after a simple reboot);
7. The virtual cluster to be totally self-contained with its own domain controller, head-node and four compute nodes;
8. Repair of system to be achievable by simple copy of file from a back-up portable drive.

VCNet essentially consists of an outer 'bare-metal' domain controller that also hosts a variety of development environments such as Microsoft Visual Studio 10 for software development using the .NET framework and NetBeans Integrated Development Environment (IDE) for software development using Java. This closely relates to the

current teaching environment used on our standard teaching setup. This is the machine used primarily for the development work and for communication with the virtual cluster machines. It was on this machine that Hyper-V services were installed onto which I built five virtual machines consisting of a head-node and four compute nodes – using the same three TCP/IP network structure used on the main USCC (see Figure 2). It was also booted from a new-style virtual disk drive (VHD) meaning that the whole system could be backed up as one fairly large file (around 100 GB).

VCNet was installed in one of the prime teaching labs (DGIC-220) on twenty student machines as well as on the instructor's machine – the latter being linked to the lab projection facility for live demonstrations – full details are given in the VCNet User Manual included in the Portfolio of Evidence [Section 6].

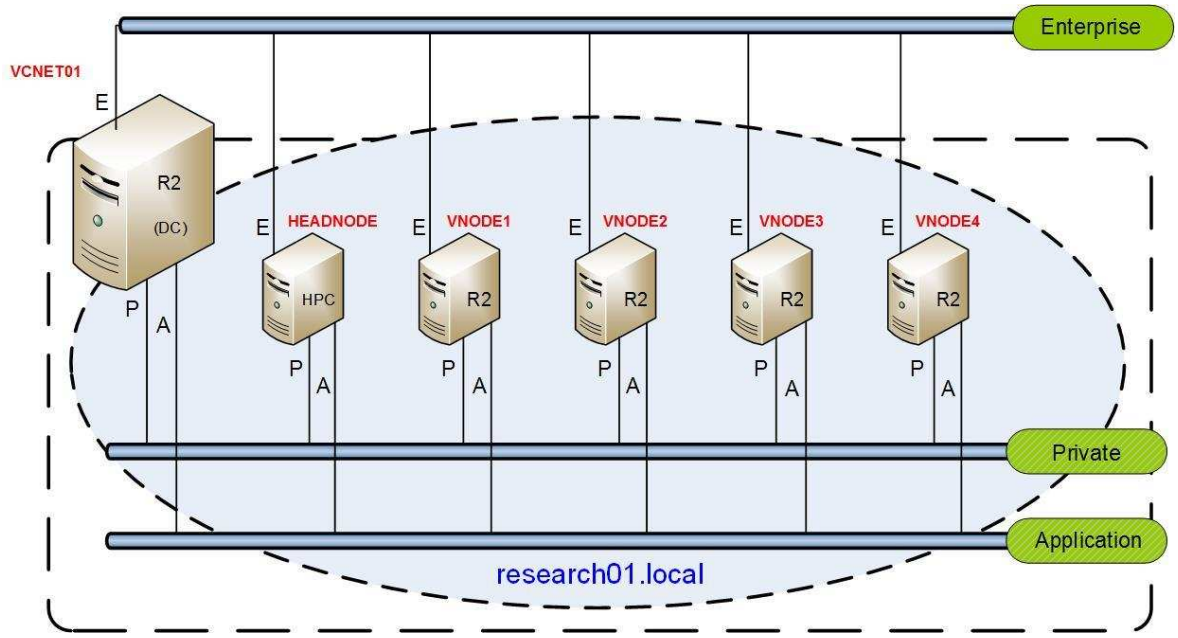

Figure 2 – VCNet System

4.4 Reflections on the VCNet System

As far as the project in its own right was concerned, VCNet was an undoubted success and resulted in the publishing of two papers [Portfolio – Sections 1.1 and 1.2] – a Teaching and Learning paper at a national HEA STEM conference in London (Robert Warrender *et al.*, 2012) and the other (a more technical paper) presented at an international conference in Finland (R. L. Warrender *et al.*, 2013a). As well as being installed in the teaching lab, I installed this on both my work computer and home computer. It was on the VCNet platform that I developed and tested the AFCC framework

and scripts prior to transferring these to the USCC for use on our collaborative project with DPHW (see Chapter 6). VCNet was also used on a Masters project where a student was able to successfully build a client/server project demonstrating this to his project supervisor.

However, there have also been negative issues that have blighted VCNet universal acceptance and ongoing development. The first was the room chosen for deploying VCNet. DGIC-220 is a software lab with reasonably high specification 4-core computers. This was necessary to get reasonable performance from the VCNet system. This turned out to be the most popular room occupancy-wise in the department severely curtailing student access to these machines within normal teaching hours.

Three other negative factors conspired to limit the impact VCNet was to have in the department. The first centred on the lack of response given by other members of the teaching staff to the use and incorporation of cluster technology into their teaching. This is perhaps unsurprising in that clusters are not the highest matter on everyone's agenda.

The second negative factor arose when an integral piece of hardware failed and had to be replaced. The system had been designed for quick repair by storing each VCNet VHD onto a portable Universal Serial Bus (USB) hard drive. This meant that a repair could normally take place by copying a single file to a machine prior to booting the system. However, failure of a motherboard or even network card was so tightly bound to the virtual machine that their replacement necessitated rebuilding the VHD from scratch which was a lengthy process. Fortunately this did not happen too often.

The final negative factor came to light later when my own VCNet machine eventually stopped working after a Microsoft upgrade. The problem related to the footprint the Windows operating system takes up after being upgraded. Having five virtual machines built inside a 'bare metal' machine (stored as a single VHD) means that the size of the virtual hard drives had to be set according to the expected footprint each machine would consume. That calculation was made according to the initial installation of Windows Server 2008 R2 and Windows Server 2008 HPC editions. However, as time passed, the Windows footprint had risen causing the machines to run out of space and fail to upgrade.

Today, this would not now be a major problem for two reasons:

1) It would have been a simple matter to specify at the start larger internal VHDs – in turn perhaps requiring a larger physical hard drive to cope with larger resulting main VHD. At the time 1 TB drives were considered very large but, of course, no longer.
2) My other concern was associated with the time I would have needed to copy this VHD to the hard drive in the event it was corrupted or needed replacement. Now with faster USB-3 becoming mainstream technology, this problem largely vanishes so dealing with larger VHDs and physical hard drives would no longer be an issue.

4.5 Summary

In the early stages, VCNet proved itself to be extremely useful for cluster software development and testing out of ideas without compromising the USCC. It provided a much needed 'test bed' to facilitate my own learning and directly led to a major operating system upgrade to the cluster using Windows Server 2008 R2 and HPC editions (from Windows Server 2003). The technical problems that ultimately beset the VCNet 'project' I believe are retrospectively solvable. These would include the following:

- To construct the system on top of Windows Server 2012 R2 and HPC version. This version of Windows Server has much improved hardware virtualisation (Hyper-v) handling as well as the ability to automatically remove all GUI parts of the code and run in command mode only. This has the effect of making a much smaller Windows footprint especially for the four cluster nodes. The improved Hyper-v capabilities in Server 2012 means that shrinking (or expanding) a VHD is easily achieved – something that did not work properly in the Server 2008 edition. Using command-line interface (CLI) versions of the operating systems also carries with it the added bonus that most Windows updates relate to the Windows GUI and not the command line version – keeping the VHDs much smaller for much longer.
- Powershell is such an integrated part of Windows 2012 that the development of a fully scripted installation is now possible even for a VCNet system. Thus in situations where a system has to be rebuilt from scratch, after say a motherboard or network card change, this could be achieved relatively painlessly leaving the machine to install VCNet on its own with minimal intervention.

Overall, the VCNet phase of this work provided a useful virtual learning and teaching environment which enables students to gain experience in high performance computing and related subjects without substantial investments in either hardware or software infrastructure. The next chapter moves onto the second aim of this professional doctorate by investigating frameworks as a means of running standard off-the-shelf software within a high performance environment.

5 Frameworks and their uses

The last chapter discussed the development of the VCNet virtual cluster environment to enhance the student experience in the field of high performance computing. This chapter discusses both the use of software frameworks as well as parallel running of tasks on a cluster. Frameworks play an important role in software engineering particularly when full source code implementations are not possible. They are ideal in cases where controlled flexibility is required to implement new and often complex multifaceted programs on most computing platforms. I examine the use of frameworks in connection with the USCC as an effective means of minimising the workload required both in the design and operation of software applied to a collaborative project processing a batch of jobs. I also look at the steps necessary to ensure efficient operation of the nodes in a cluster configuration.

5.1 Framework Introduction

Designing a new software application needs careful and detailed specification to make sure that the end product fully meets both its current goals as well as likely future developments. In research, however, we are often faced with situations where we need to see the results of one set of calculations or activities before we can formulate the next step. Often the next step is simply to abandon the current path, backtrack to an earlier time and proceed down a completely different pathway.

Attempting to write a single high performance piece of software to meet the changing requirements of a research project would be expensive, badly structured, and unmaintainable. An altogether different approach is needed to get through this type of scenario. This is typically where software frameworks can be used to put together a sequence of activities that can easily be altered, maintained or automated to simulate a much larger and complex piece of software.

The Oxford English Dictionary (OED, 2015) defines Frameworks as "*A structure made of parts joined to form a frame;* esp. *one designed to enclose or support; a frame or skeleton*".

In construction activities, this would be, for example, scaffolding erected around a workplace to provide a safe working environment to carry out a whole series of otherwise difficult and often hazardous tasks. Scaffolding provides a 'staging' or 'landing' area where

temporary or periodic works can be carried out without the need to provide more permanent and expensive solutions.

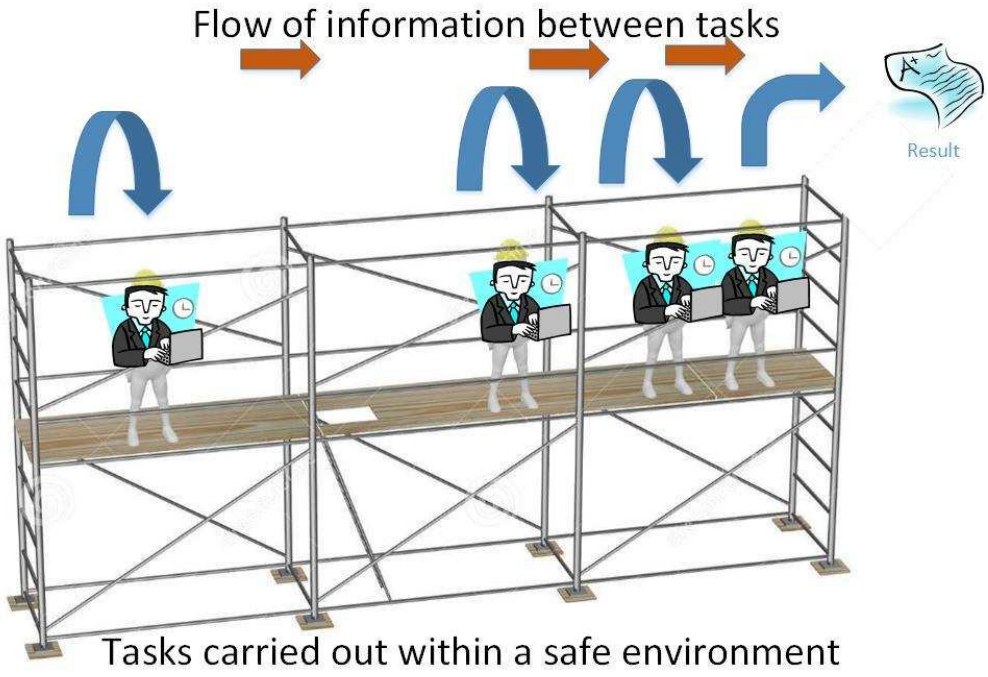

Figure 3 – Software Framework

In software engineering, we often need the equivalent 'staging' or 'landing' area to carry out a range of routine tasks in a safe manner (see Figure 3). The software will likely consist of a mix of both proprietary and custom software – often with quite tenuous connections. For example, after running a piece of proprietary software, we might require to extract a single value using some preselected criterion in order to utilise this knowledge within a quite different software application. Having the safe 'staging' or 'landing' area for one piece of software will almost certainly require some form of 'conditioning' of the results before this can be used within the context of another software program. Initially this type of process, common within most research and development environments, would likely be undertaken manually especially while the usefulness of the overall process was being evaluated. However, once deemed an essential element of the process, a more automated approach would allow this sequence of tasks to be applied to a larger range of research data.

This is still a long way from writing a custom piece of software to implement the whole process – indeed the whole integrity of the results may well hinge on the use of specific proprietary software whose source code is commercially unavailable.

Software frameworks manifest themselves in many formats – the simplest being a set of separate commands or instructions written down on paper where individual results can be monitored at each stage. As the research process becomes more predictable, these would then be elevated to batch files to provide a more automated set of solutions for the task in question. At the next level, batch files would be modified to accept some degree of input 'overloading' to make these more general and responsive to external influences.

An alternative approach would be to utilise a 'drag and drop' type solution or building exercise where the intended functionality is represented in workflows (Barga *et al.*, 2008) such as Taverna (Hull *et al.*, 2006), Accelrys Pipeline Pilot (Pilot, 2011), Bioclypse (Spjuth *et al.*, 2007), Kepler Scientific Workflow (Altintas *et al.*, 2004). Whatever solution that is selected, each uses the implied rigidity of the framework to provide a safe 'haven' for intermediate results before progressing these on to an even greater goal or solution.

5.2 Parallel Running on the USCC

The object of parallel programming is to break down a large task into smaller pieces and carry these out, as far as possible, within the same time space using separate resources thereby reducing the overall time to complete the original task (Vile and Liddle, 2009). In computing terms there are many ways to achieve this, such as spawning multiple threads to separate processors to accomplish the parallel pieces of work in a shared memory environment before recombining the results. Alternatively this could involve levels of message passing between machines encouraging each machine to carry out some specialised section of work. True parallel processing usually starts with source code and software programmers with a vision of how parallelism for their application is best achieved.

But what happens when source code is not available and as a practitioner you are faced with 'closed source' applications – can these benefit from a cluster computer to achieve significant levels of performance speedup? The answer to that largely comes from a careful study of the application's documentation.

Occasionally, it is possible to find client/server versions of software that run naturally on a cluster. Examples I have used in this category include 3D Studio Max (AutoDesk, 2015), Lightwave3D (NewTek, 2015), Cinema 4D (Maxon, 2015) and Blender (2015). In these cases, the software developers have allowed for the use of large numbers of clients running on multiple machines. Breaking the overall rendering work down, groups of frames are sent to different machines for processing with the resulting work collected centrally and forming an integrated single job.

Others examples of parallel activities found in an application is where it can make use of Symmetric Multiprocessing (SMP). This is where a microprocessor typically contains multiple processor cores within a shared memory environment. Gaussian09 (2012a) is an example of this where the number of processors can be specified in a configuration file. The original software programmer has likely adopted a threading model within his program which can take advantage of the multiple cores to effect a program speedup. In the case of Gaussian09, valid core configurations are one, two, four or eight cores.

If the software application has not been written to be 'parallel amenable' in its own right (Bernstein, 1966), then we can still resort to running the software as an 'embarrassingly parallel' application (Régin *et al.*, 2013). Embarrassingly parallel is where no attempt has been made to divide the original problem into parallel tasks other than to run different instances of the application on separate nodes. This is useful when processing a batch of jobs through a cluster. The USCC has forty separate compute nodes so being able to run 40 jobs at a time in parallel can still provide a significant speedup when there are many jobs to process in a batch.

5.3 Summary

In this chapter, I have discussed the use of software frameworks in a research scenario and how these could be applied to allow many tasks or jobs to be scheduled out to multiple nodes in a cluster computer. In the next chapter I will show how this approach was used to develop the Application Framework for Computational Chemistry (AFCC).

6 Application Framework for Computational Chemistry (AFCC)

In this chapter I discuss the development and use of a software framework in a university collaborative project between the department of Pharmacy, Health & Wellbeing (DPHW) and the department of Computing, Engineering & Technology (DCET). I will look briefly at the intent of the collaboration and how a fairly simple framework run on the USCC was used with great effect to take the project from conceptual ideas to more in-depth research and future collaboration with a local pharmaceutical company.

6.1 Overview of Molecular Modelling

Although the work carried out within this Professional Doctorate relates primarily to cluster computing, we will look at one specific collaborative project which has had considerable success. Working with a group from the DPHW, I was introduced to the world of Molecular Modelling. The research group were having limited success modelling molecules and compounds with their own resources due to the time required to successfully reach any form of satisfactory result. Scientists apply their knowledge of quantum mechanics to simulate what will happen to molecules and compounds at the atomic level. Indeed the whole field of 'in silico' research is fuelling the use of high performance computers in the pharmacy industry, more specifically in the area of new drug discovery.

According to DPHW, Gaussian09 is one of the foremost and well respected pieces of quantum mechanics software. The first version of the Gaussian computer program was written by Professor John Pople to make his computational techniques easily accessible to researchers (Tindle *et al.*, 2012; Tindle *et al.*, 2014). Professor Pople was awarded the Nobel Prize in chemistry in 1998 for his pioneering contributions in developing computational methods (Van Houten, 2002). By entering details of molecules of interest, scientists can get an accurate representation of both the molecules' structural geometry and electronic positions of its constituent atoms and electrons. Gaussian09 generates what is known as a Potential Energy Surface (PES) typically shown in Figure 4.

The valleys in the PES diagram represent areas of least energy and as such equate to stable physical structures for these molecules. The saddles represent paths that have to be crossed in order to take a given molecular isomer from one stable state to another.

Although the chemistry constituents are identical, the size and shape of the resulting molecular isomers will be different, as will be how they react with other molecules in close proximity.

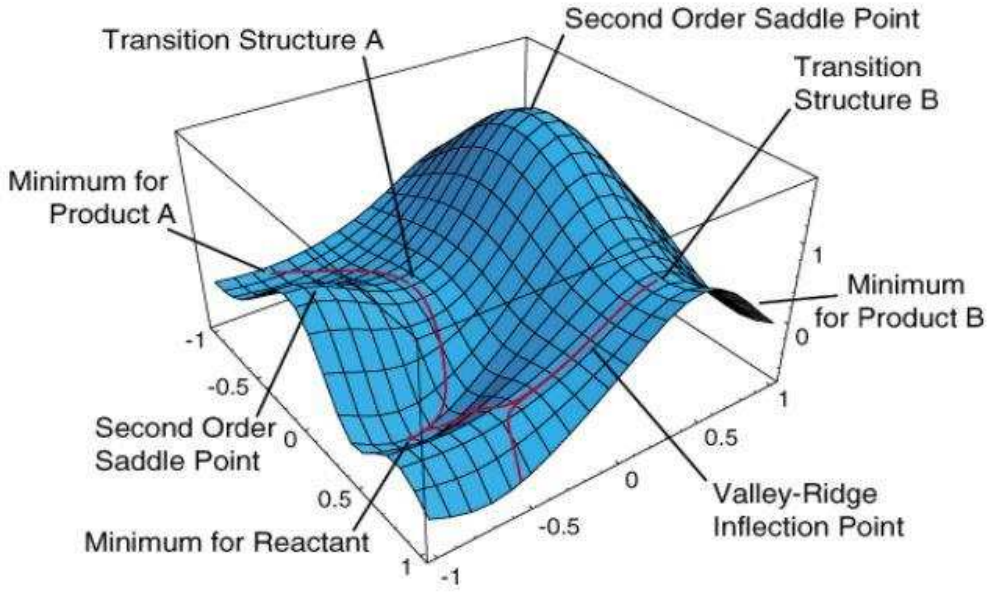

Figure 4 – Typical Potential Energy Surface Diagram (Schlegel, 2008)

Using this type of information, scientists can test out a whole variety of compounds in order to select the best candidate drugs for manufacturing and possible testing, discarding all others at an early stage – thus saving a great deal of time and expense in their test selections.

6.2 Design Philosophy

It became apparent at an early stage that some form of software framework would be necessary to run Gaussian jobs efficiently on the USCC. Gaussian relies heavily on three key sources of information to run any job:

i) The software utilises an environment variable to keep track of the location of Gaussian 'links' (Gaussian, 2012b) or internal programs called up by the main software. This has to be setup each time Gaussian, as a piece of software, is invoked.

ii) The software utilises a Gaussian input file (with an extension of .gjf). As well as providing details of the molecules to be analysed, the .gjf file can

contain many instructions to be passed to the Gaussian software to tailor the solution or method of calculation. Optionally Gaussian can use already calculated binary .chk files to provide details of the molecules being analysed, but the .gjf file is always required for any additional instructions necessary to process the job. Use of .chk files will be discussed later as this is a way of importing already calculated details for this molecule from a previous run, thus reducing the time for Gaussian to complete its task.

iii) Gaussian accepts command line arguments, primarily related to files, locations as well as types of outputs expected.

While setting these files up manually is not necessarily difficult, it is both time consuming and error prone. Gaussian, for example, carries information related to the computer on which the input (.gjf) files were created. As this invariably related to work done at another campus, these had to be corrected to run on the USCC. What was needed was a simple way of copying batches of files to a directory and automatically resolving contentions without serious user involvement.

This was run as a collaborative venture between DPHW and DCET but the demarcation between our responsibilities was always clear. DPHW provided all information as to the purpose and chemistry of the work being carried out. DCET's role in the collaboration was to provide the equipment with myself providing the optimised framework to ensure that these jobs ran efficiently and accurately within a parallel computing cluster. I was able to carry out some basic analysis on the results and provide information as to whether each particular job within the batches had completed successfully without any reported errors. However it was DPHW task to put meaning on the results and formulate future directions for where the research should proceed in terms of their overall aim to find potential drug targets that could be taken to a more advanced level of development.

Over the course of a few months, a series of scripts and C# programs were developed which formed the basis of the original AFCC. The basis of the AFCC is shown in the Fig 5. Most of the code written for the AFCC was developed by the author and coded in C#. This formed the basis of the second aim of the Professional Doctorate – as defined in Chapter 1.2.

It should be noted here that there are ways to speed up the use of Gaussian09 software (R. L. Warrender *et al.*, 2013c). One way would have been to use TCP Linda (Scientific Computing Associates, 2015) which is recommended by Gaussian and allocates multiple

nodes to solve a single job. However this approach was not adopted due to the parallel speed-up curves published by Gaussian for this type of approach (Gaussian, 2012c). As the number of nodes increase (i.e. distributed processors) there is a substantial drop-off in speed-up indicating a loss of efficiency. We therefore chose to stay with maximising the use of Symmetric Multiprocessing (SMP) within each node and allocating a node for each job.

Gausian09's own published results for increasing cores within each node (Gaussian, 2012c) also shows a marked dip in performance as the number of cores increase but as the USCC uses twin dual core processors, the performance degradation was not that significant. This dip in performance for large number of cores is discussed in a journal paper (R L Warrender *et al.*, 2013b) [Portfolio - Section 2.2] and is likely related to contention in multicore processors – this has been the subject of many research papers (Fedorova *et al.*, 2010; Hood *et al.*, 2010; Zhuravlev *et al.*, 2010; Reineke and Wilhelm, 2014).

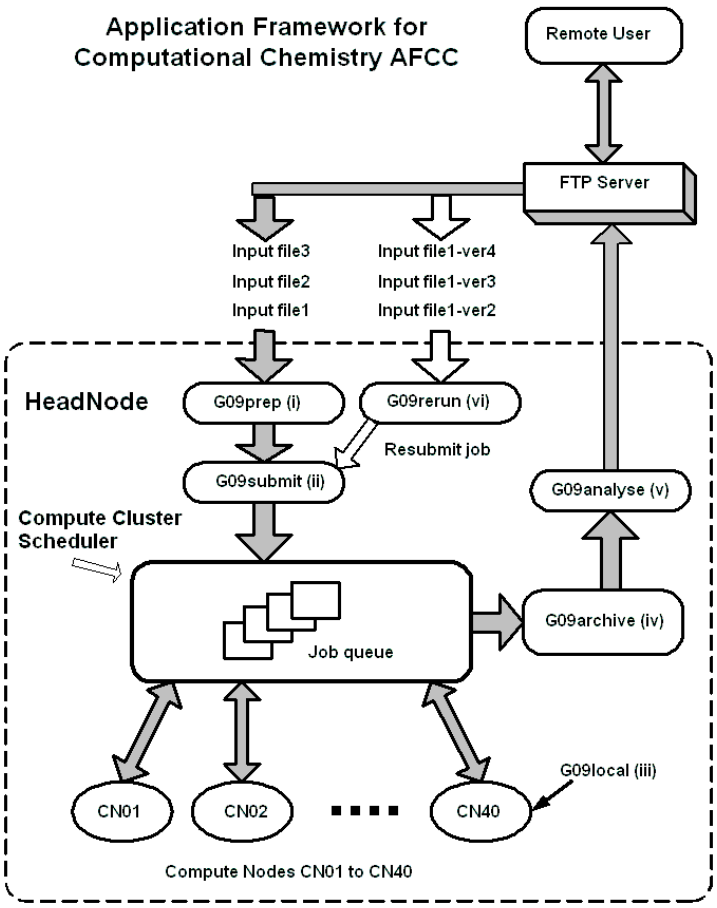

Figure 5 - Application Framework for Computational Chemistry (AFCC)

Another way to speed-up Gaussian09 jobs would have been through the use of GPUs in the calculations (R. L. Warrender *et al.*, 2013c). However at the time of writing this dissertation, the software for this approach is still not available despite the Gaussian announcement back in 2011 (Gaussian, 2012d).

The six main constituent parts of the framework (as shown in Figure 5) are as follows:

i)	**G09Prep**:	used to prepare an existing input file (.gjf) to run within the cluster environment;
ii)	**G09Submit**:	used to send individual jobs to the main HPC scheduler;
iii)	**G09Local**:	used to accept scheduler data and prepare this to run on a local node environment;
iv)	**G09Archive**:	used to isolate the batch results into an archive directory to permit its re-use;
v)	**G09Analyse**:	used to analyse the success of the job as well as calculate time spent within a node;
vi)	**G09Rerun**:	same function as G09Prep but allows re-use of existing .chk files.

One of the problems found when dealing with computer clusters relates to naming conventions of the various associated drives – for example, Drive C on the head-node is clearly a different drive from Drive C on a compute node. We get over this usually by using Universal Naming Convention (UNC) paths which take the form:

\\servername\sharename\filepath

Using this convention, we can easily access any file across a network for which we hold permissions. Unfortunately, however, Microsoft HPC scheduler does not support UNC paths so an alternative way of passing filenames to the compute nodes had to be adopted. To solve this problem, I arranged to pass the constituent elements of the UNC path via the scheduler using C command line arguments. The scheduler effectively makes a call to run the appropriate G09Local program at the node rather than the eventual Gaussian Software. The G09Local program parses the parameters passed in, and reconstructs the full UNC path before passing this information to the actual Gaussian

software at the node. Once complete, control passes back to the G09Local program to perform a 'clean-up' operation before returning control back to the HPC scheduler, to schedule out the next unit of work. The scheduler typically uses first-come, first-served (FCFS) techniques (Zhang *et al.*, 2000; Feitelson *et al.*, 2005; R. L. Warrender *et al.*, 2013c) in deciding the order of job execution.

On completion of a batch of jobs, I wrote two utility programs to aid with the cataloguing of results and determination of the success for the Gaussian software. G09Archive was a convenient way of mass copying of all files used in a particular batch run to an appropriate sub-directory. This frees up the work directory so that we can immediately start the next batch of jobs. G09Analyse was written as a simple tool to determine the success or otherwise of each individual job within the batch. Gaussian produces some very large text files often several megabytes in size (.out files) which need analysis, first to determine whether a successful outcome had been achieved. These also give us indirectly the time spent by each job within a node and as such can be considered a 'cost' metric for each job. A single Gaussian run may have several passes within the Gaussian software and as such each lists out a series of different times scattered throughout the output file. G09Analyse 'combs' through the output file first to establish whether the overall outcome was successful and also to calculate the total time spent in each of the Gaussian passes.

The final component of the AFCC framework is G09Rerun. This is identical to the function of G09Prep other than it first ensures there are pairs of configuration (.gjf) files and checkpoint (.chk) files with the same root name present in the directory. In situations where further work was deemed important to the molecule, requiring a further Gaussian run, considerable time could be saved by using an existing binary check file (.chk) rather than generate this from scratch.

6.3 Gaussian

Working with computational chemists from the DPHW proved to be an interesting experience and learning curve all around. Clearly they had difficulty in carrying out any level of molecular modelling using Gaussian software. In theory they knew what they wanted to explore but the shear physical limitations of their computers proved a step too far for their initial work. Each simulation was taking at least one week on a spare departmental computer severely hampering the number of jobs that could be tackled in a

realistic fashion. After several exploratory meetings, the chemists supplied around 50 test jobs to see if these could be run on the cluster. We were all keen to see if we could find a way of processing these in a much more efficient manner which could form the basis for future work. Gaussian also provided with their software some test jobs that could be run within the environment. To be fair these were much smaller jobs but provided a good set of test cases to help find out what needed to be done to efficiently handle batches of work that we would eventually expect to receive from the computational chemists.

During this period, I was able to work on the idea of a framework which could be used to give some level of automated handling of the batches. Early tests suggested that a time of about five minutes would be needed to prepare each job to work on the cluster environment – much of which came from the somewhat error-prone typing out of extended chemical filenames used by the chemists to describe the molecular compounds being simulated. All too often, similar names with minor differences were used; some with and some without embedded spaces.[3] I was able to work with the chemists to come up with a more computer friendly naming convention which was unique, contained no embedded spaces, of fixed length and did not have any special chemical significance (that information would be embedded in the Gaussian Input file if required).

Once the original AFCC framework was written and tried out on the test batch of Gaussian jobs, it was time to step up and attempt some real jobs. Eventually a set of jobs were delivered, entitled "the Über batch" – the significance of which was never explained by the chemists, however was accepted in good faith. The batch consisted of some 276 jobs and was completed in less than two weeks. The average time for the jobs being processed in a node (each using 4 cores) was 15.9 hours. The maximum time taken to process a successful job was around 66 hours (see Figure 6).

[3] Note that similar names with minor differences to a 'layman' often has major significance to the chemist originating the job name.

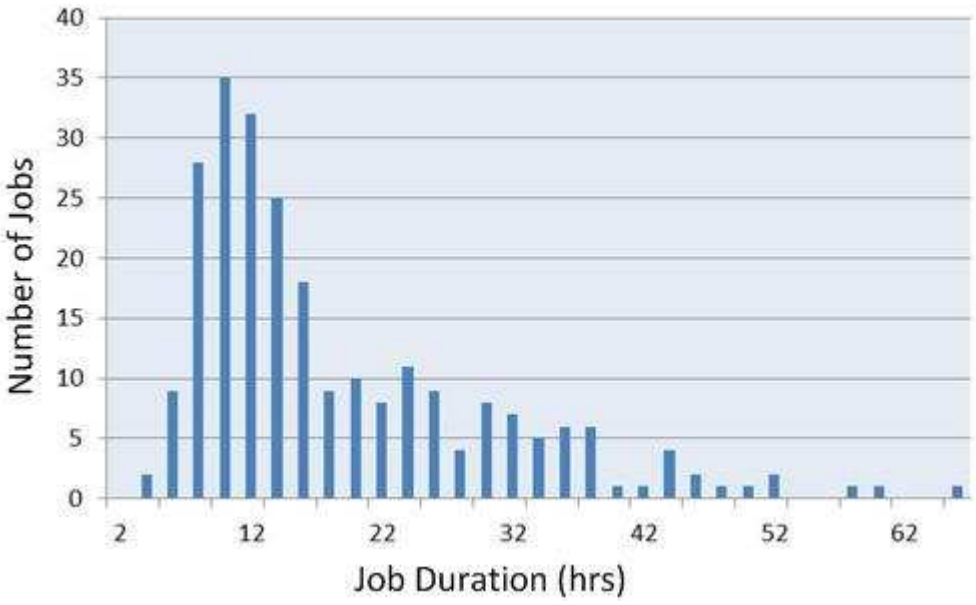

Figure 6 – Job Time in Nodes (sorted by duration)

The success rate of obtaining results on this initial batch was approximately 89%. This left around 11% of the jobs that failed to reach a satisfactory conclusion. This we were able to jointly break down with the chemists into three categories:

Table 1 – Summary of batch job outcomes

Comment	%
Total number of jobs that failed to converge in a reasonable time. In these cases the run time was typically more than three days.	5%
Total number of jobs that failed because an error was detected in the input file. In these cases the run time was typically less than 4 seconds.	4%
A small number of Compute Nodes crashed during program execution.	2%
Failed jobs total	11%

The first category of jobs that failed to converge were of most concern from a cluster viewpoint. These represented a high cluster 'cost' to implement with no apparent measurable success. Largely these were caused by problems in the 'chemistry' being attempted. Going through the Gaussian output files the chemists were usually able to spot the source of the problem and make the necessary corrections. By returning the binary (.chk) files along with the amended configuration (.gjf) files for reprocessing on the cluster using the G09Rerun software, not only (in most cases) allowed the job to reach a converged state but in a fraction of the time taken during the initial run.

The second category of jobs were more annoying than 'costly' from a cluster viewpoint. It was generally related to bad 'housekeeping' errors such as incorrect file names or incorrect commands in the configuration file. The Gaussian software manages to detect this type of error quickly and terminates the job with a suitable error message. The wasted time came from having to return these files for correction by the chemists before they could be run once again on the USCC.

The third category of error was one that did concern me as this generally related to hardware issues on the cluster or software errors within the framework. Usually Gaussian software performs its own 'clean-up' on the node after each job, however in one specific batch run I had a mass failure of almost the entire batch for no apparent reason. Sometimes it appeared to work, but when I re-ran it again it would fail. The chemists checked their configuration files but could not locate the problem. I tried to run one of the jobs on my laptop one evening to try to get to the bottom of the problem. After the job had run successfully once, when it was re-run it continually failed. I decided to save the files to a pen drive for analysis the following day, only to find the computer trying to copy around 15GB of data to the pen drive. Far from clearing up the work directory, Gaussian was leaving huge data files which were affecting the following run on that node. I modified the framework to carry out a mandatory clean-up before returning control back to the HPC scheduler. This cured the problem. With regard to hardware failures, other than issues related to power supplies and the loss of two servers, this (now seven year old hardware) has proved itself to be remarkably stable which has to be a tribute to the design team who specified and built the original system.

6.4 AFCC Compared with Manual Approach

Adopting the AFCC approach has dramatically speeded up the time taken to process batches of Gaussian jobs. Indeed in the paper I presented in 2013 (R. L. Warrender *et al.*, 2013c) [Portfolio – Section 1.3], I was able to show a performance gain (or speedup) using the AFCC on the USCC of approximately 140 times provided the input queue contains outstanding jobs ready for processing.

The chemists from their side have been working on ways to increase the 'first-time' success rate as well as providing jobs in a more amenable form thereby reducing the time to process a job. This is done primarily by running the Gaussian job through a low resolution pass prior to sending the files on for more detailed processing. This not only

traps out most of the 'housekeeping' type of errors such as misnamed files or indeed 'bad' chemistry[4], but also provides a much better starting position for the various atoms in the molecules through the supply of an initial binary (.chk) file. As a result of these improvements, I have successfully run more than 10,000 Gaussian jobs in the three year period working on this collaborative project with DPHW.

6.5 Summary

To date, the AFCC has been used successfully with both Gaussian 09 and AutoDock Vina (Trott and Olson, 2010; AutoDock, 2014) software packages within the chemistry domain. However it has also been tested to good effect with a custom financial package written in MatLab (MathWorks, 2014) code. AFCC provides a safe working environment on which to:

a) Prepare any input files associated with a batch of jobs to run effectively within the cluster environment
b) Submit jobs to a scheduler using C passing parameters which can be intercepted and resolved at the node level
c) At the node level, call a ready built software package in its own native format before tidying up the node making sure files are returned to the appropriate directory in the head-node as well as clear up any temporary files generated locally on the node.

Using this technique means that jobs are being run in an 'Embarrassingly Parallel' configuration utilising as much Symmetric Multi-processing (SMP) as the original ready built software will support. The feeding of jobs through in large batches means that node 'starvation' doesn't become a problem and the overall efficiency of the resulting processing is high.

This chapter describes the development and use of the AFCC software framework over a period of some three years automating the processing of many thousands of Gaussian jobs, in a cross-disciplinary research project between DCET and DPHW. Over the period, the AFCC has proved to be a stable and largely trouble free framework fully meeting the second aim of this Professional Doctorate.

[4] The term 'bad' chemistry here is being used to signify errors in either the chemical structure or composition of the molecules and compounds being used.

While this technique worked well for homogeneous batch work, it doesn't lend itself to continuous processing where different jobs can be at different stages and indeed may need to follow different paths based on intermediate results. In the next chapter I will present the work currently being done to extend the AFCC into a more powerful passport driven solution which is being adopted for the follow-on work of this professional doctorate in a new commercially funded cross-disciplinary PhD project.

7 Application Framework for Computational Chemistry (AFCC) Mark II

Over the last three years, the AFCC in its original form, has run more than 10,000 Gaussian jobs for the DPHW, resulting in the creation of a 300 member boronic acid fragment library. A decision was taken earlier in 2014 that the collaboration between DPHW and DCET would continue with new funding being sought to extend the project. After securing funding for a full-time PhD studentship, interviews took place over the summer and a Masters graduate awarded the studentship entitled the 'Development of Functional Computational Workflow Methods for Quality Automated Molecular Modelling-based Drug Design' [Portfolio - Section 9.1]. As DPHW's intention was for me to be a second supervisor to the student, I was also part of the interview panel.

In this chapter we look at what has to change in the AFCC in its Mk II version to fulfil such a role and briefly explore the possible future use of Cloud Computing facilities to augment our USCC facility.

It should be clearly stated at this stage that this chapter demonstrates the link to further research activities directly attributable to the work done under this professional doctorate. The on-going research of these activities and direction of this chapter will form the main work of a further doctoral report in the computational chemistry field.

7.1 New Design Philosophy

In the book chapter (Tindle *et al.*, 2014) [Portfolio – Section 3.1], we investigated how the AFCC could be adapted to implement the various iterative processing stages automatically within the AFCC without the need for human intervention either by a chemist or systems manager. Originally we considered a scripting language to take care of the iteration (as shown in Figure 7).

While this on the surface seemed to satisfy the needs of the chemists on the project, it soon became apparent that more substantial changes would be required to meet the demands of the project.

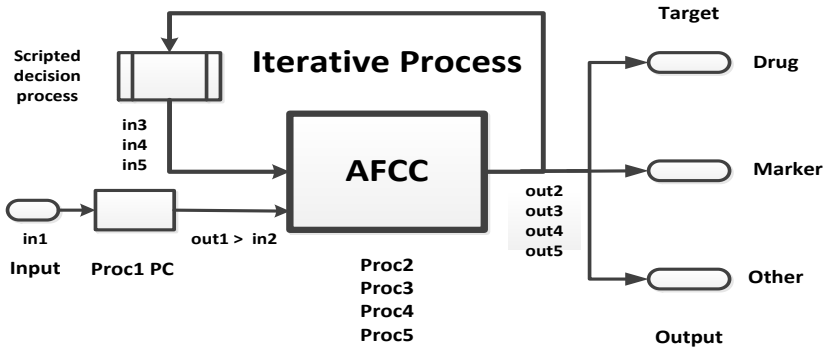

Figure 7 – AFCC Iterative Processes (Tindle et al., 2014)

At a University Joint Research Beacon Seminar (between DPHW and DCET) in April 2014 in the Dale Building [Portfolio – Section 5.4], I presented details of our continued collaboration on the drug discovery project. I also introduced details of a much more ambitious software artefact which would be developed within DCET which I called AFCC Mk II. The main details are shown in Figure 8.

As with the original AFCC, each job will carry its own unique base filename. It will also carry its own 'passport' file – the concept of a passport is to record how the job has 'travelled' (i.e. what processes it has already encountered) as well as details of likely future 'destinations' (i.e. the tasks the chemists still expect to be carried out). It will also keep track of its current position in its 'journey'. As the passport file will be a simple text file, the program can quickly determine what it has to do next.

All job files are kept in a convenient folder representing a batch of work to be started at a particular point in time. Although there will be several folders in existence keeping the work flowing, the folders help reinforce the concept of a batch of work being processed. At the completion of an individual job's journey, all associated files relating to its unique base filename will be archived into an archive folder and take no further part in the system.

The overall system will have two queues – a serial queue for short duration preparatory work and a parallel queue to store jobs ready to send out to worker nodes on the cluster.

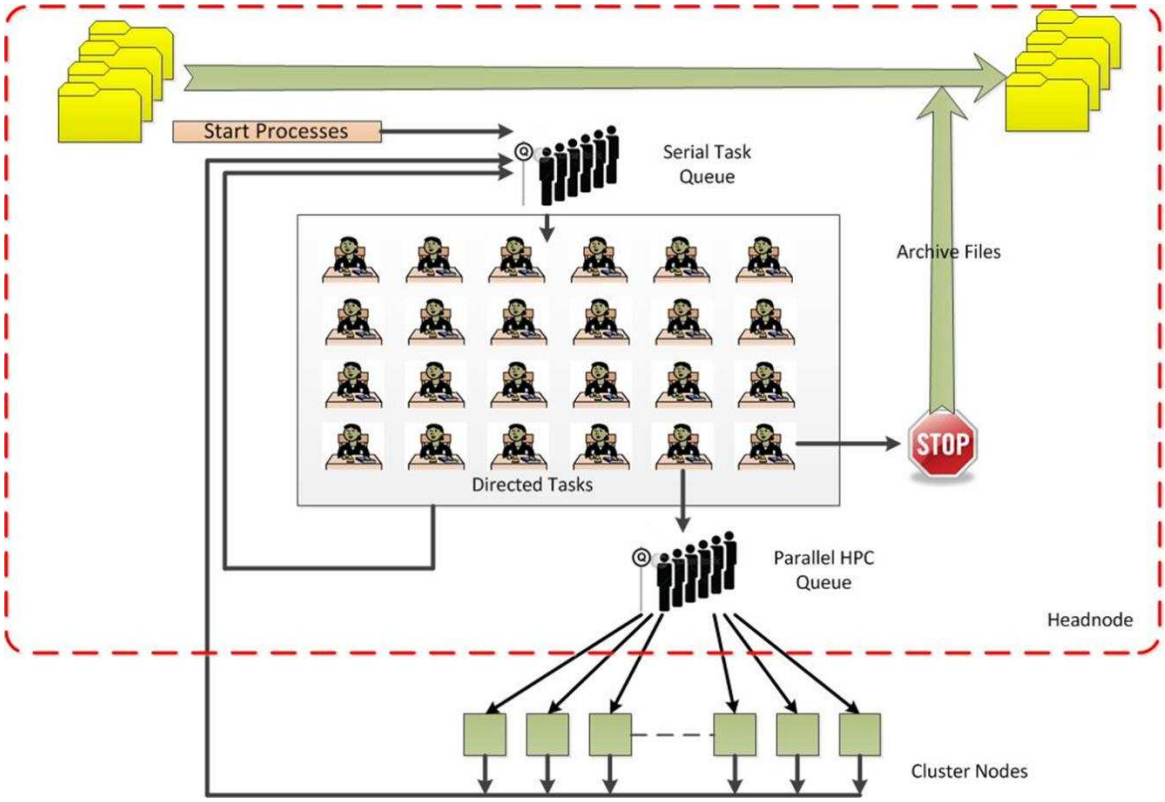

Figure 8 - AFCC Mk II presented at University Research Beacon April 2014

To start the system off, a program will generate a set of passports – one for each job found in the particular work folder. Depending on the work which has been put in that folder, the passport will contain the 'visas' for an expected journey within the system. The passports will be stored in the work folder but details of their locations will be placed in the serial queue.

The parallel queue is the HPC scheduler and as such runs continuously feeding jobs out to available nodes when these become free. If no jobs are loaded in the HPC scheduler, then the system will idle until this starts to fill up again.

The serial queue will be designed to run as a service on the head-node and consist of a list of passports (including their locations) waiting to be processed. The first passport name will be removed from the queue and its details passed to one of the many 'directed tasks' available within the system – according to the next destination listed in the passport. A directed task is simply a program or script written to carry out a simple task – such as G09Prep or G09Submit in the original AFCC system (see Figure 5 in Chapter 6.2). First it will read any necessary details from the passport file before carrying out its task. Normally on completion of the task, the passport will be updated in the original work

folder and its details added to the end of the serial queue. As soon as this passport name 'bubbles' up to the head of the queue, the next directed task can take place on this job.

Some of the directed tasks will be HPC specific – such as when Gaussian09 or AutoDock Vina (AutoDock, 2014) is being called. In this case, the directed task (such as G09Submit) will complete when the job has been sent to the HPC scheduler and will return for the next passport details without placing this in the serial queue. It will in fact now be the task of each node to update the passport and return its details to the end of the serial queue.

The last directed task will be to take all associated files relating to its unique base filename and archive these – effectively removing these from the system.

This overall approach creates a 'cartridge' style of software writing where indeed the chemists will be responsible for their own 'directed tasks'. It will be 'passport driven' with the overall system creating a form of software 'peristalsis' action driving the various software tasks forward from start to completion. Whether they choose to program these in C, C#, Python or indeed any other language will largely be immaterial. We will provide sample code in their desired language to read and write parameters to the passport and allow them the freedom to express themselves in terms of the desired chemistry. They also do not have to worry about the complexities of parallel programming and yet still have a system that should be efficient and operating at its peak performance. The major delays in this type of system will be the software running on the HPC. The use of serial queues running directed tasks effectively on a single core should result in little or no performance delays to the whole operation as well as give them the maximum freedom to look for ways to accomplish the desired modelling.

The chemistry for where we are going is untried, untested and in truth no more than a figment of some fertile minds. It is therefore likely that the whole project will make use of an incremental development methodology testing out ideas until more concrete knowledge is gained. Having this cartridge based approach means that each directed task can be manually tested and results checked as we go along. Depending on what is found might result in other tests being carried out and results stored in the passport file for use at a later stage.

At this moment all programs being linked to this project are Windows based – primarily because of the use of Windows within the DPHW. What if this situation changes and we

find that some cluster software needs to run on a Linux platform? It would be a simple task to keep a list of which passports required the next task to be run on a Linux or Windows environment. The cluster design is that of a dual boot system and we have tested scripts that allow us to take nodes down that run under Windows HPC and start these up within a SGE cluster. It would then be a matter of trying to keep the balance of operating systems to that of the needs of the serial queue.

Another approach would be to look at gaining access to a cloud cluster that could be provisioned into either a Windows or Linux cloud (Assuncao *et al.*, 2009). Cloud is fast becoming the way forward in clusters.

7.2 Summary

This chapter has briefly outlined future work that has arisen as an outcome of the development of the AFCC and therefore this professional doctorate. The next chapter will summarise the main outcomes of the Professional Doctorate.

8 Conclusions

The object of this doctorate report has been to *"demonstrate advanced and systematic knowledge and skills in the candidate's chosen area"* (UoS, 2009a). The portfolio has to show how this *"forms a contribution to the creation and interpretation of new knowledge at the time of submission and must be set in the context of understanding of the field" (UoS, 2009a).* In this chapter I will draw together the key points of the professional doctorate, what has been achieved, the contribution to knowledge and professional practice that have been achieved as well as how each of the various elements of this submission meet the key learning objectives as laid down in the programme regulations.

8.1 Development of VCNet for use in teaching within the department

VCNet was developed as a non-invasive technology to enhance student learning opportunities by giving each individual student access to a virtual cluster without the need for specialised hardware. It takes a standard lab machine transforming its function into a six-machine virtual cluster (domain controller, head-node and four compute nodes – see Figure 2 in Chapter 4.3) using a single password controlled re-boot script. Similarly, use of a second controlled re-boot script returns the virtual cluster back to a normal lab machine for general purpose teaching.

Although VCNet is not being used currently within the mainstream university curriculum, it does not imply that this will always be the case. Indeed, with advanced multicore processors becoming more affordable and available, this means that the significance of the work will largely grow with time. VCNet has been used on selected Master's student projects and its value to my own research has been significant – especially in the early days when access was not readily available due to refurbishment of the USCC area that turned into a six month shutdown of the facility. Being almost entirely software dependant (with the exception of the multiple network cards and larger hard drives) makes this an almost 'zero cost' option. Being VHD bootable to a separate virtual hard drive also gives each student full 'admin' access to facilities being denied on all other machines for security reasons. A further successful outcome from the development of VCNet was in its use as an environment for the development and testing of the AFCC software framework before its deployment on the USCC.

One other bright note is the prospect that low cost 3D Vertical NAND flash memory (Hang-Ting *et al.*, 2010) could start to appear as 10 TB solid state drives (SSD) at prices to at least rival current magnetic drives (Mellor, 2014). This would have a significant effect on the performance of VCNet in universities and make its use even more important with universities having to function on ever-tightening budgets.

8.2 Collaborative Work Success

The development of the AFCC has provided a highly efficient framework for optimising the running of batches of quantum mechanics jobs on the USCC – in many cases running unattended for several months at a time. The AFCC takes jobs from input batch folders, optimises the parameters for use on the USCC, utilises a scheduler to carry the detailed parameters to a custom program located on each node which in turn picks up configuration and input data across the cluster network, runs the job locally on a standard piece of proprietary software before returning the completed results back across the network to the original folders.

The collaborative work undertaken between the DPHW and DCET has been a major contribution to this professional doctorate and significant for both groups of researchers. It has enabled academic publication at both international and national conferences as well as two international journals in our field as evidenced in the Portfolio [Sections 1 and 2]. In addition, the book chapter [Portfolio - Section 3] gave us a chance to update our previous journal publication with significant work undertaken in the succeeding period.

This collaboration now continues with the sponsoring of a PhD studentship (by a UK drug company) and commencement of a full-time PhD student [Portfolio Section 9.1]. I am second supervisor for this PhD student which provides further evidence of ongoing contribution to my academic practice in research. Although the student appointed is a chemistry master's graduate, it is expected that joint departmental publications between DPHW and DCET will continue over the duration of the studentship as well as individual papers generated within DCET related to the novel approaches being adopted within the computing domain.

8.3 Community of Practice

In addition to the purely research aspects, this professional doctorate has also been successful at a 'community of practice' level. Each year I deliver a lecture on grid and

cluster computing to the level 1 computer science students. In this lecture I cover many topics such as:

a) Architectural issues of grid and cluster computers;
b) Early supercomputers through to the current Top 500 machines (TOP500, 2015);
c) Board level architectures and the transition from single to multi-core processors;
d) Demonstrations of different programming techniques using software code to generate prime numbers;
e) Discussion with examples of the research undertaken with various groups around the University;
f) Visit to the cluster room to see the cluster.

The lecture has also been used with selected master's groups on a periodic basis. Feedback from both undergraduate and master's students has primarily been positive and examples are given in the Portfolio of Evidence [Section 5.3].

In addition to teaching on-campus, I have also given presentations to two different groups:

a) British Computer Society Newcastle and & District Branch (BCS) 15th Dec 2010;
b) Society of Information Technology Management (SOCITM) 21st Sep 2012.

I also shared a presentation with Professor John Mellis, BT Technology Service & Operations on "BT innovation in the Pharma and Health Sciences sectors" 11th April 2014 at a joint Digital Innovation Research & Health Sciences and Wellbeing Beacons seminar [Portfolio – Section 5.4].

8.4 Contribution to Knowledge and Practice

Although the design of the cluster was fairly unique seven years ago – i.e. a general purpose computer cluster with twin head-nodes capable of being booted (in whole or in part) into either Windows or Linux operating systems, time has taken its toll and designs have moved on – especially in areas such as GPUs (Vishakha Gupta *et al.*, 2009; McIntosh-Smith, 2011) and cloud computing (He *et al.*, 2010; Ostermann *et al.*, 2010; Abhishek Gupta *et al.*, 2012). However probably the most significant legacy of this professional doctorate has been the work detailing the steps necessary to implement

cluster solutions involving traditional workstation software without having to rewrite even a single line of code. The AFCC is built on the premise of carrying out an optimised running of standard software on each node making as much use of symmetric multiprocessing as possible and scheduling in an 'embarrassingly parallel' configuration. We have demonstrated that clusters of this type can run at relatively high levels of speed-up taking into account the available number of processors in the system – in our case a speed-up in Gaussian09 of 140 with a total node processor count of 160 provided the scheduler does not get starved of jobs. Indeed under this arrangement, jobs of different types can successfully utilise the same scheduler ensuring a constant supply of outstanding work.

The VCNet is also a significant legacy of this professional doctorate in that cluster computing can be considered viable for use within a teaching environment. Being able to run a script to boot directly to a fully implemented cluster carries with it significant attractions especially as invoking a similar script can be used to boot the system back to a 'normal' teaching lab setup.

One of the potential future contributions of this work has been the identification within the department for a new flagship Masters in Cloud and High Performance Computing. With an accreditation event for the department's Master's provision scheduled for the 2016/17 academic year, now would be the time to start planning out how the cluster would feature in this context, and VCNet in particular, giving all students access to their own personal cluster to enhance their learning experience.

The most significant contribution of this doctorate has been felt not so much within the Computing and Engineering department but in the department of Pharmacy, Health and Well-Being. Molecular modelling has long been part of the chemistry syllabus but without any serious means of modelling complex molecules. As part of their courses, students now get a chance to carry out some molecular modelling using industry standard Gaussian09 software and run on the USCC. Although the impact on staff within CET has been limited, the impact on research by staff in DPHW has been significant. This impact has been publically referred to by the senior management within the faculty, and the cluster is now promoted as a university-wide research resource.

8.5 Evaluation on the success of using the Cluster in Teaching

Use of both the main USCC cluster as well as VCNet has not been taken up by many staff teaching at the university. Each year I give a presentation to Level 1 Computer Science students which generally produces some excellent feedback [Portfolio – Section 5.3]. Although there is a good interest in high performance computing amongst the students in Year 1, this appears to fade fast in later years primarily in my opinion due to the nature of the syllabus. Students are striving for a good degree classification and only want to get involved in subjects that will enhance their career prospects.

Concurrency, multi-threading and distributed computing are already part of an existing computing module but the staff concerned have their own way of teaching these concepts. A more appropriate use for clusters might have been within the Software Engineering Masters programme but alas that programme no longer recruits due to market demand. The possibility of a new Master's degree in High Performance and Cloud Computing would be a strong possibility but will need access to cloud computing resources to make this a reality. As discussed in Chapter 7, it is hoped that the future research in AFCC Mk 2 will indeed take us firmly into cloud computing to augment our existing cluster based USCC facility.

8.6 Meeting the Learning Outcomes

According to the University of Sunderland regulations for Professional Doctorate Report and Portfolio (UoS, 2009b), students must be able to demonstrate the following knowledge, skills and abilities (Fulton *et al.*, 2013):

To meet this requirement, I have related the various applicable Chapters from this report and Sections from the Portfolio:

	Learning Outcomes	How this was achieved	Reference
K1	Deep understanding of the recent developments in their profession nationally and internationally	Thorough understanding shown in contextual review with regard to teaching of HPC. Understanding of state-of-the art in the HPC	Chapter 3 Section 5

		area, e.g. top-500 clusters International recognition through published papers and the REF.	Sections 1 and 2
K2	Deep understanding of current theoretical frameworks and approaches which have direct relevance to their own professional context	Thorough investigation and adoption of suitable research methodologies. Pedagogical aspects of L&T – development of a novel virtual learning environment to enhance student learning Main outcome investigation and utilisation of a software framework within a research environment.	Chapter 3 Chapter 4 Chapters 5 & 6
S1	Make a significant contribution to practice within their chosen area	Development of VCNet as a Learning and Teaching test environment AFCC Implementation maintaining high productivity in a research environment.	Chapter 4 Chapter 5
S2	Apply theory and research methodology within the workspace and feel comfortable in integrating different approaches to address "messy" multidisciplinary problems in a rigorous yet practical manner	Development methodology Cross-disciplinary research Software development (plus networking, operating systems, etc.) including the architecting of new frameworks to keep work running on the cluster at a high level of efficiency and productivity	Chapter 1.3, 1.31 and 1.32
S3	Recognise budgetary, political, strategic, ethical and social issues when addressing issues within	Many areas related to both the upkeep and maintenance of the USCC as well as developing an almost zero-cost platform solution for teaching high performance	Chapter 2, 4 and 6 as well as in Section 6

	the workplace	computing within the curriculum.	
S4	Reflect on their own work, and on themselves, and thus operate as a truly reflective independent practitioner	The Professional Doctorate has given me the opportunity to work as a reflective practitioner both in my role as lecturer as well as part-time researcher. This appears in both the writing style of the dissertation as well as in the various taught modules	Report, all chapters
S5	Present and defend an original and coherent body of work which demonstrates, reflects upon, and evaluates the impact upon practice which they have personally made	Both the Dissertation and Portfolio I believe show a contribution to the profession primarily in relation to the work on VCNet as well as the collaborative research work carried out of the period of the Professional Doctorate. The research papers, journal articles as well as book chapter impact on the faculty/profession	Report & Portfolio Section 8

8.7 Research Excellence Framework 2014

In 2014, being a full-time member of staff at the University of Sunderland with recent publications, my work was submitted by the University as part of the Research Excellence Framework (REF2014) submittal. Although we did not get a breakdown of our individual results or outcomes, we were able to see the overall results for the department – see Figure 9, which clearly shows that all papers submitted achieved a classification of 1* or above.

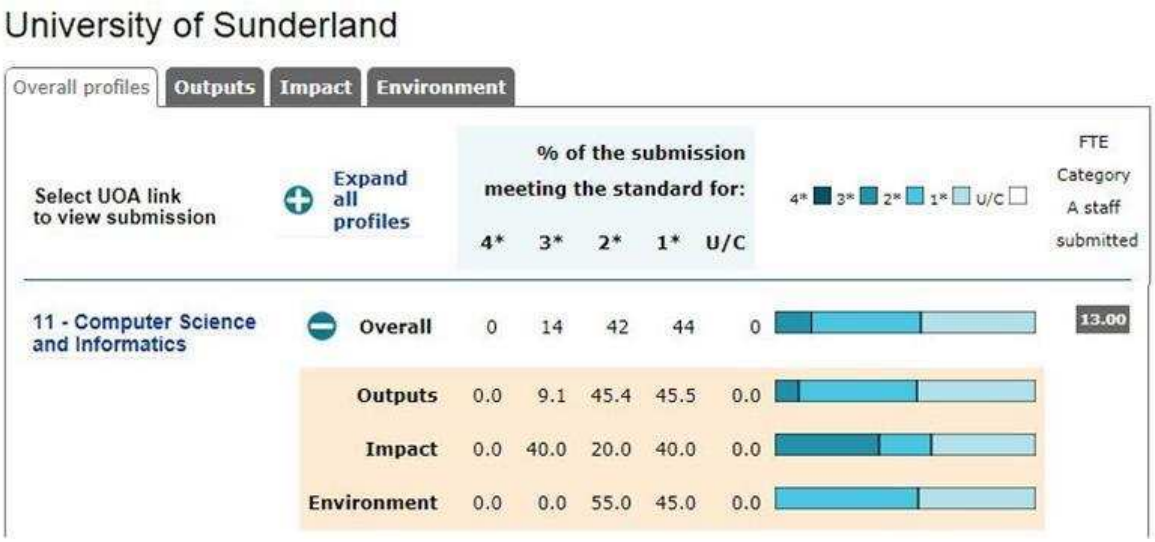

Figure 9 – Results for the UoS Computer Science and Informatics (REF2014)

Four of my papers were submitted as part of the REF2014 exercise. These were as follows:

1) Development of a Virtual Cluster – paper presented at the 2013 International Conference on High Performance Computing & Simulation, 1 – 5 Jul 2013, Helsinki, Finland. [Portfolio – Section 1.2].
2) Job Scheduling in a High Performance Computing Environment – paper presented at the 2013 International Conference on High Performance Computing & Simulation, 1 – 5 Jul 2013, Helsinki, Finland. [Portfolio – Section 1.3].
3) Application Framework for Computational Chemistry (AFCC) Applied to New Drug Discovery – paper published (2012) in the International Journal of Grid and High-Performance Computing, 4 (2). pp 46-62. ISBN 1938-0259. [Portfolio – Section 2.1].
4) Evaluating the Use of Virtual Machines in High Performance Clusters – paper published (2013) in International Journal of Advanced Computing Technology, 2 (5). pp 25-30. ISBN 2319-7900. [Portfolio – Section 2.2].

The REF classifies papers into five different research categories as follows:

1) World leading (4*)
2) Internationally excellent (3*)
3) Recognised internationally (2*)
4) Recognised nationally (1*)
5) Unclassified

As a new researcher to the area, it was therefore pleasing to see that all my papers submitted had achieved classification – indeed two papers were presented at an international conference in Finland and later included in IEEE Xplore Digital Library (R. L. Warrender *et al.*, 2013a; R. L. Warrender *et al.*, 2013c). The other two papers were published as journal papers in two respected international journals, the International Journal of Grid and High Performance Computing (Tindle *et al.*, 2012) as well as the International Journal of Advanced Computer Technology (R L Warrender *et al.*, 2013b). Out of a total of 13 full-time equivalents (FTEs) submitted as part of the University of Sunderland REF2014 submission, my papers and journal articles had made a significant contribution towards the University's success in this regard.

8.8 Final thoughts on the Professional Doctorate

The journey from a young graduate engineer to the present has been a journey very much of discovery and fulfilment over the course of what has been a lifetime's work. I was born at the start of the transistor revolution, studied through the early days of the integrated circuit era and was an early designer of microprocessor based systems which have fuelled the computational and embedded electronic markets for many years. I have worked in different parts of the UK as well as working for a substantial period of time as a consultant in the Middle East on a major water infrastructure project. Gaining an MSc while living abroad, I returned to the UK to take up an academic post at the University of Sunderland.

As a chartered engineer, I believe I have always been a 'closet' reflective practitioner but the experience gained particularly in the various taught modules during the early part of the Professional Doctorate has reinforced this aspect and provided some logical insight to this process. The most difficult aspect still remains the encouragement to use 'I … I' in my writing having lived through an age of engineering writing (including my time as a consultant) where personal statements were always frowned on and all views expressed had to be on behalf of the company.

The Professional Doctorate has provided a means to combine my desire for research within a teaching environment. These works are by no means finished and hope to carry on with my research and publishing with several of my colleagues as well as collaborative groups, looking forward to the next Research Excellence Framework (REF2020).

References

Albrecht, J. R. (2009) 'Bringing big systems to small schools: distributed systems for undergraduates', *Proceedings of the 40th ACM technical symposium on Computer science education*. Chattanooga, TN, USA. 1508903: ACM, pp. 101-105.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B. and Mock, S. (2004) *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE.

Amdahl, G. M. (1967) 'Validity of the single processor approach to achieving large scale computing capabilities', *Proceedings of the April 18-20, 1967, spring joint computer conference*. Atlantic City, New Jersey. 1465560: ACM, pp. 483-485.

Apon, A., Mache, J., Buyya, R. and Hai, J. (2004) 'Cluster computing in the classroom and integration with computing curricula 2001', *Education, IEEE Transactions on*, 47(2), pp. 188-195.

Assuncao, M. D. d., Costanzo, A. d. and Buyya, R. (2009) 'Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters', *Proceedings of the 18th ACM international symposium on High performance distributed computing*. Garching, Germany. 1551635: ACM, pp. 141-150.

Auliansyah, N. and Kusniawati, N. (2014) *2014 International Conference on Advances in Education Technology (ICEAT-14)*. Atlantis Press.

AutoDesk (2015) *3DS Max*. Available at: http://www.autodesk.co.uk/products/3ds-max/overview (Accessed: 8th Jan 2015).

AutoDock (2014) *AutoDock Vina*. Available at: http://vina.scripps.edu/ (Accessed: 1st July 2014).

Barga, R. S., Fay, D., Guo, D., Newhouse, S., Simmhan, Y. and Szalay, A. (2008) *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*. ACM.

Basili, V. R., Cruzes, D., Carver, J. C., Hochstein, L. M., Hollingsworth, J. K., Zelkowitz, M. V. and Shull, F. (2008) 'Understanding the high-performance-computing community: A Software Engineer's Perspective', *IEEE Softw.*, 25(4), p. 8.

Bates, A. T. (2003) *Managing technological change*. Jossey-Bass.

Bernstein, A. J. (1966) 'Analysis of Programs for Parallel Processing', *Electronic Computers, IEEE Transactions on*, EC-15(5), pp. 757-763.

Blender (2015) *Blender*. Available at: http://www.blender.org/ (Accessed: 8th Jan 2015).

Borkar, S. and Chien, A. A. (2011) 'The future of microprocessors', *Commun. ACM*, 54(5), pp. 67-77.

Checkland, P. (1981) *Systems thinking, systems practice*. J. Wiley.

Daniel, M. E. (1967) 'Development of mathematical models of semiconductor devices for computer-aided circuit analysis', *Proceedings of the IEEE*, 55(11), pp. 1913-1920.

Dean, C. R., Young, A. F., MericI, LeeC, WangL, SorgenfreiS, WatanabeK, TaniguchiT, KimP, Shepard, K. L. and HoneJ (2010) 'Boron nitride substrates for high-quality graphene electronics', *Nat Nano*, 5(10), pp. 722-726.

Dell (2008) *Eco-Friendly Super Computing*. Available at: http://www.cit.sunderland.ac.uk/downloads/files/Sunderland_University.pdf (Accessed: 22nd March 2015).

Dreher, P. and Vouk, M. (2013) 'Integration of high-performance computing into a VCL cloud', *Proceedings of the 8th Workshop on Virtualization in High-Performance Cloud Computing*. Denver, Colorado. 2535802: ACM, pp. 1-6.

Endo, P. T., Gonçalves, G. E., Kelner, J. and Sadok, D. (2010) 'A survey on open-source cloud computing solutions', *VIII Workshop em Clouds, Grids e Aplicações*. pp. 3-16.

Ertugrul, N. (2000) 'Towards virtual laboratories: a survey of LabVIEW-based teaching/learning tools and future trends', *International Journal of Engineering Education*, 16(3), pp. 171-180.

Faggin, F., Hoff, M. E., Mazor, S. and Shima, M. (1996) 'The history of the 4004', *Micro, IEEE*, 16(6), pp. 10-20.

Farian, H., Anne, K. M. and Haas, M. (2008) 'Teaching high-performance computing in the undergraduate college CS curriculum', *J. Comput. Sci. Coll.*, 23(3), pp. 135-142.

Fedorova, A., Blagodurov, S. and Zhuravlev, S. (2010) 'Managing contention for shared resources on multicore processors', *Communications of the ACM*, 53(2), pp. 49-57.

Feitelson, D., Rudolph, L. and Schwiegelshohn, U. (2005) 'Parallel job scheduling - a status report', *Job Scheduling Strategies for Parallel Processing*. Springer, pp. 1-16.

Fitz Gibbon, A., Joiner, D. A., Neeman, H., Peck, C. and Thompson, S. (2010) *Proceedings of the 2010 TeraGrid Conference*. ACM.

Fuller, S. H. and Millett, L. I. (2011) 'Computing Performance: Game Over or Next Level?', *Computer*, 44(1), pp. 31-38.

Fulton, J., Kuit, J., Sanders, G. and Smith, P. (2013) *The Professional Doctorate*. Palgrave MacMillan.

Gal-On, S. and Levy, M. (2008) 'Measuring multicore performance', *Computer*, 41(11), pp. 99-102.

Gardner, D. (2006) 'Microsoft Launches Windows Compute Cluster Server 2003', *Information Week*, 9th June 2006.

Gaussian (2012a) *Gaussian 09*. Available at: http://www.gaussian.com/g_prod/g09.htm (Accessed: 22nd Nov 2012).

Gaussian (2012b) *Gaussian 09 Links*. Available at: http://www.gaussian.com/g_tech/g_ur/m_linklist.htm (Accessed: 22nd Nov 2012).

Gaussian (2012c) *Get Your Gaussian Results Sooner*. Available at: http://www.gaussian.com/g_prod/parallel.htm (Accessed: 22nd Nov 2012).

Gaussian (2012d) *NVIDIA GPUs to Accelerate World-Leading Quantum Chemistry Application* Available at: http://www.gaussian.com/g_press/nvidia_press.htm (Accessed: 22nd Nov 2012).

Ginty, K., Tindle, J. and Tindle, S. (2009a) 'Cluster Systems - An Open Access Design Solution', *20th International Conference on Systems Engineering: ICSE 2009*. Coventry, UK, 8-10 Sep 2009.

Ginty, K., Tindle, J. and Tindle, S. (2009b) 'Rendering 3D Computer Graphics on a Parallel Computer', *20th International Conference on Systems Engineering: ICSE 2009*. Coventry, UK, 8-10th Sep 2009.

Gupta, A., Kal, L. V., #233, Milojicic, D. S., Faraboschi, P., Kaufmann, R., March, V., Gioachin, F., Suen, C. H. and Lee, B.-S. (2012) 'Exploring the performance and mapping of HPC applications to platforms in the cloud', *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*. Delft, The Netherlands. 2287093: ACM, pp. 121-122.

Gupta, V., Gavrilovska, A., Schwan, K., Kharche, H., Tolia, N., Talwar, V. and Ranganathan, P. (2009) 'GViM: GPU-accelerated virtual machines', *Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing*. Nuremburg, Germany. 1519141: ACM, pp. 17-24.

Gustafson, J. L. (1988) 'Reevaluating Amdahl's law', *Commun. ACM*, 31(5), pp. 532-533.

Hang-Ting, L., Tzu-Hsuan, H., Yi-Hsuan, H., Hong, S. P., Wu, M. T., Hsu, F. H., Lien, N. Z., Szu-Yu, W., Jung-Yu, H., Ling-Wu, Y., Tahone, Y., Kuang-Chao, C., Kuang-Yeu, H. and Chih-Yuan, L. (2010) 'A highly scalable 8-layer 3D vertical-gate (VG) TFT NAND Flash using junction-free buried channel BE-SONOS device', *VLSI Technology (VLSIT), 2010 Symposium on*. 15-17 June 2010. pp. 131-132.

He, Q., Zhou, S., Kobler, B., Duffy, D. and McGlynn, T. (2010) 'Case study for running HPC applications in public clouds', *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. Chicago, Illinois. 1851535: ACM, pp. 395-401.

Hoisie, A. and Getov, V. (2009) 'Extreme-scale computing–where'just more of the same'does not work', *Computer*, 42(11), pp. 24-26.

Hood, R., Jin, H., Mehrotra, P., Chang, J., Djomehri, J., Gavali, S., Jespersen, D., Taylor, K. and Biswas, R. (2010) 'Performance impact of resource contention in multicore systems', *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. IEEE, pp. 1-12.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P. and Oinn, T. (2006) 'Taverna: a tool for building and running workflows of services', *Nucleic acids research*, 34(suppl 2), pp. W729-W732.

Hwu, W.-m., Keutzer, K. and Mattson, T. G. (2008) 'The concurrency challenge', *IEEE Design and Test of Computers*, 25(4), pp. 312-320.

Joiner, D. A., Gray, P., Murphy, T. and Peck, C. (2006) *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM.

Lee, N.-J. (2009) *Achieving your professional doctorate*. McGraw-Hill Education.

Leiserson, C. E. and Mirman, I. B. (2008) 'How to survive the multicore software revolution (or at least survive the hype)', *Cilk Arts, Cambridge*.

Leverich, J. and Kozyrakis, C. (2010) 'On the energy (in)efficiency of Hadoop clusters', *SIGOPS Oper. Syst. Rev.*, 44(1), pp. 61-65.

Li, H., Wang, L., Liu, Q., Zheng, J., Mei, W.-N., Gao, Z., Shi, J. and Lu, J. (2012) 'High performance silicene nanoribbon field effect transistors with current saturation', *The European Physical Journal B*, 85(8), pp. 1-6.

Li, P., Toderick, L. W. and Lunsford, P. J. (2009) 'Experiencing virtual computing lab in information technology education', *Proceedings of the 10th ACM conference on SIG-information technology education*. Fairfax, Virginia, USA. 1631747: ACM, pp. 55-59.

LSTC (2015) *LS-DYNA*. Available at: http://www.lstc.com/products/ls-dyna.

March, S. T. and Smith, G. F. (1995) 'Design and natural science research on information technology', *Decis. Support Syst.*, 15(4), pp. 251-266.

MathWorks (2014) *Matlab*. Available at: http://uk.mathworks.com/index.html.

Maxon (2015) *Cinema 4D*. Available at: http://www.maxon.net/ (Accessed: 8th Jan 2015).

McIntosh-Smith, S. (2011) 'The gpu computing revolution'.

Mellor, C. (2014) *Intel offers ingenious piece of 10TB 3D NAND chippery*. Available at: http://www.theregister.co.uk/2014/11/21/intel_offering_an_ingenious_piece_of_10tb_3d_nand_chippery/ (Accessed: 22nd Feb 2015).

Microsoft (2012a) *Windows HPC Server 2008 R2*. Available at: http://technet.microsoft.com/en-us/library/ee783547(WS.10).aspx.

Microsoft (2012b) *Technical Overview of Windows HPC 2008 Server R2*. Available at: http://technet.microsoft.com/en-us/library/ee783547(WS.10).aspx.

Microsoft (2012c) *Scheduling Jobs with Windows HPC Server 2008 R2*. Available at: https://cmg.vlabcenter.com/default.aspx?moduleid=f41f9b1b-416b-4ca2-839e-aa3bbbb410af.

Microsoft (2012d) *Managing and Monitoring HPC Clusters using HPC Server 2008 R2*. Available at: https://cmg.vlabcenter.com/default.aspx?moduleid=1ee1-aeb0-4aa0-9cdd-ca78ba00845d.

Murshed, M. and Buyya, R. (2002) 'Using the GridSim toolkit for enabling grid computing education', *International Conference on Communication Networks and Distributed Systems Modeling and Simulation (CNDS 2002)*, 31.

Nagel, L. W. and Enterprises, O. (1996) 'The life of SPICE', *1996 Bipolar Circuits and Technology Meeting*.

Nelson, D., Rossiter, N., Stirk, S. and Warrender, R. (2004) 'A Comparison of the Product Model and the Third Manifesto', *8th World Multiconference on Systemics, Cybernetics and Informatics*. Orlando, Florida, July 2004.

NewTek (2015) *LightWave3D*. Available at: https://www.lightwave3d.com/ (Accessed: 8th Jan 2015).

Oates, B. J. (2006) *Researching Information Systems and Computing*. London: Sage Publications Ltd.

Oxford English Dictionary (2015) Oxford University Press [Online]. Available at: http://www.oed.com/view/Entry/74161?redirectedFrom=framework.

Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T. and Epema, D. (2010) 'A performance analysis of EC2 cloud computing services for scientific computing', in *Cloud computing*. Springer, pp. 115-131.

Parkhurst, J., Darringer, J. and Grundmann, B. (2006) *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. ACM.

Pilot, P. (2011) 'Version 8.5. Accelrys', *Inc.: San Diego, CA*, 92121.

Popper, K. R. S. (1959) *The logic of scientific discovery / Karl Raimund Popper ; translation prepared by the author with the assistance of Julius Freed and Lar Freed*. London: Hutchinson.

Rees, C., Forbes, P. and Keble, B. (2006) 'Student employability profiles', *The Higher Education Academy, York*.

REF2014 (2014) *Research Excellence Framework 2014*. Available at: http://www.ref.ac.uk/ (Accessed: 18th December 2014).

Régin, J.-C., Rezgui, M. and Malapert, A. (2013) 'Embarrassingly Parallel Search', in Schulte, C. (ed.) *Principles and Practice of Constraint Programming*. Springer Berlin Heidelberg, pp. 596-610.

Rehr, J., Vila, F., Gardner, J., Svec, L. and Prange, M. (2011) 'Scientific Computing in the Cloud', *Computing in Science & Engineering*, PP(99), pp. 1-1.

Reineke, J. and Wilhelm, R. (2014) *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE.

Scalise, E., Houssa, M., Pourtois, G., Afanas'ev, V. and Stesmans, A. (2012) 'Strain-induced semiconductor to metal transition in the two-dimensional honeycomb structure of MoS2', *Nano Research*, 5(1), pp. 43-48.

Schaffer, H. E., Averitt, S. F., Hoit, M. I., Peeler, A., Sills, E. D. and Vouk, M. A. (2009) 'NCSU's virtual computing lab: a cloud computing solution', *Computer*, 42(7), pp. 94-97.

Scheckler, R. K. (2003) 'Virtual labs: a substitute for traditional labs?', *International journal of developmental biology*, 47(2/3), pp. 231-236.

Schlegel, H. B. (2008) *Typical Potential Energy Surface Diagram*. Available at: http://www.chem.wayne.edu/~hbs/chm6440/ (Accessed: 11th Aug 2014).

Schwierz, F. (2010) 'Graphene transistors', *Nat Nano*, 5(7), pp. 487-496.

Scientific Computing Associates (2015) *TCP Linda*. Available at: http://www.lindaspaces.com/products/linda.html (Accessed: 16th Mar 2015).

Shalf, J., Dosanjh, S. and Morrison, J. (2011) 'Exascale computing technology challenges', in *High Performance Computing for Computational Science–VECPAR 2010*. Springer, pp. 1-25.

Silberschatz, A., Galvin, P. B. and Gagne, G. (2001) *Operating System Concepts*. John Wiley & Sons, Inc.

Sinnott, R. O., Stell, A. J. and Watt, J. (2005) *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*. 9-12 May 2005.

Sodan, A. C., Machina, J., Deshmeh, A., Macnaughton, K. and Esbaugh, B. (2010) 'Parallelism via multithreaded and multicore CPUs', *Computer*, (3), pp. 24-32.

Spjuth, O., Helmus, T., Willighagen, E. L., Kuhn, S., Eklund, M., Wagener, J., Murray-Rust, P., Steinbeck, C. and Wikberg, J. E. (2007) 'Bioclipse: an open source workbench for chemo-and bioinformatics', *BMC bioinformatics*, 8(1), p. 59.

Sulistio, A., Schulz, A., Keller, R., Kritikakos, P., Kostas, M. and Varvarigou, T. (2012) *High Performance Computing and Simulation (HPCS), 2012 International Conference on*. 2-6 July 2012.

Tennant, M. (2004) 'Doctoring the knowledge worker', *Studies in Continuing Education*, 26(3), pp. 431-441.

Thiruvathukal, G. K., Hinsen, K., La, x, ufer, K. and Kaylor, J. (2010) 'Virtualization for Computational Scientists', *Computing in Science & Engineering*, 12(4), pp. 52-61.

Tichy, W. F. (1998) 'Should Computer Scientists Experiment More?', *Computer*, 31(5), pp. 32-40.

Tindle, J., Gray, M., Warrender, R. L., Ginty, K. and Dawson, P. (2012) *Application Framework for Computational Chemistry (AFCC) Applied to New Drug Discovery*. IGI Global. [Online] Available at: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jghpc.2012040104.

Tindle, J., Gray, M., Warrender, R. L., Ginty, K. and Dawson, P. K. D. (2014) 'Further Development of an Application Framework for Computational Chemistry (AFCC) Applied to New Drug Discovery', in Mehdi, K.-P. (ed.) *Contemporary Advancements in Information Technology Development in Dynamic Environments*. Hershey, PA, USA: IGI Global, pp. 92-110.

TOP500 (2015) *TOP 500 The List*. Available at: http://www.top500.org/ (Accessed: 21st Mar 2015).

Trott, O. and Olson, A. J. (2010) 'AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading', *Journal of Computational Chemistry*, 31(2), pp. 455-461.

UKUUG (2004) *Do It Yourself - KNOPPIX Remastering the Easy way*. Available at: http://www.ukuug.org/events/linux2004/knoppix.shtml.

UoS (2009a) *AQH-L5 Regulations for Prof.Doc. & Associated Awards*. Available at: https://docushare.sunderland.ac.uk/docushare/dsweb/Get/Document-3771/AQH-L5 (Accessed: 22nd Dec 2014).

UoS (2009b) *AQH-L8 Professional Doctorate Programme Specification*. Available at: https://docushare.sunderland.ac.uk/docushare/dsweb/Get/Document-3771/AQH-L8 (Accessed: 22nd Dec 2014).

Van Houten, J. (2002) 'A Century of Chemical Dynamics Traced through the Nobel Prizes. 1998: Walter Kohn and John Pople', *Journal of Chemical Education*, 79(11), p. 1297.

Vile, A. and Liddle, J. (2009) *The Savvy Guide to HPC, Grid, Data Grid, Virtualisation and Cloud Computing*. Milton Keynes, UK: Lightning Source UK Ltd.

Walters, R., Millard, D., Bennett, P., Argels, D., Crouch, S., Gilbert, L. and Wills, G. (2006) 'Teaching the Grid: Learning Distributed Computing with the M grid Framework', *World Conference on Educational Multimedia, Hypermedia and Telecommunications 2006*. AACE, pp. 2234-2241. Available at: http://www.editlib.org/p/23317.

Warrender, R., Nelson, D. and Tindle, J. (2012) 'Virtual cluster to enhance student experience within a teaching environment', in *Proceedings of the HEA STEM Learning and Teaching Conference (2012)*. The Higher Education Academy.

Warrender, R. L. (2003) *Databases*. Exeter: Learning Matters Ltd.

Warrender, R. L., Tindle, J. and Naylor, I. (2004) 'UoSLinux–A Linux LiveCD distribution for use in Higher Education', *UKUUG 2004 Linux Technical Conference, Leeds*, pp. 5-8.

Warrender, R. L., Tindle, J. and Nelson, D. (2013a) 'Development of a virtual cluster', *International Conference on High Performance Computing and Simulation (HPCS), 2013* pp. 545-551.

Warrender, R. L., Tindle, J. and Nelson, D. (2013b) 'Evaluating the use of Virtual Machines in High Performance Clusters', *International Journal of Advanced Computer Technology (IJACT)*, Volume 2(5), pp. 25 - 30.

Warrender, R. L., Tindle, J. and Nelson, D. (2013c) 'Job scheduling in a high performance computing environment', *International Conference on High Performance Computing and Simulation (HPCS), 2013* pp. 592-598.

Wisker, G. (2007) *The Postgraduate Research Handbook: Suceeed with your MA, MPhil, EdD and PhD*. New York: Palgrave.

Zhang, Y., Franke, H., Moreira, J. and Sivasubramaniam, A. (2000) *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*. IEEE.

Zhuravlev, S., Blagodurov, S. and Fedorova, A. (2010) 'Addressing shared resource contention in multicore processors via scheduling', *SIGARCH Comput. Archit. News*, 38(1), pp. 129-142.