Application Framework for Computational Chemistry (AFCC) applied to New Drug Discovery

J. Tindle, M. Gray, R.L. Warrender, K. Ginty, P.K.D. Dawson**

Department of Computing, Engineering and Technology
*Department of Pharmacy, Health and Well-being **
Faculty of Applied Sciences,
University of Sunderland, St. Peter's Campus,
Sunderland, SR6 0DD, U.K.
(e-mail: robert.warrender@sunderland.ac.uk)

ABSTRACT

This paper describes the performance of a compute cluster applied to solve three dimensional (3D) molecular modelling problems. The primary goal of this work is to identify new potential drugs. The paper focuses upon the following issues: computational chemistry, computational efficiency, task scheduling and the analysis of system performance. The philosophy of design for an application framework for computational chemistry (AFCC) is described. Various experiments have been carried out to optimise the performance of a cluster computer, the results analysed and the statistics produced are discussed in the paper.

Keywords: High performance computing HPC, new drug discovery, molecular modelling, computational chemistry, parallel computing, compute cluster

1. INTRODUCTION

Computational methods have been developed that allow researchers to carry out comprehensive investigation of molecules and their reactions. Molecular modelling allows the researcher to observe biological reactions and the associated dynamics providing highly accurate descriptions of the relevant interatomic forces. In most cases it is computationally expensive to solve the high level quantum chemistry models.

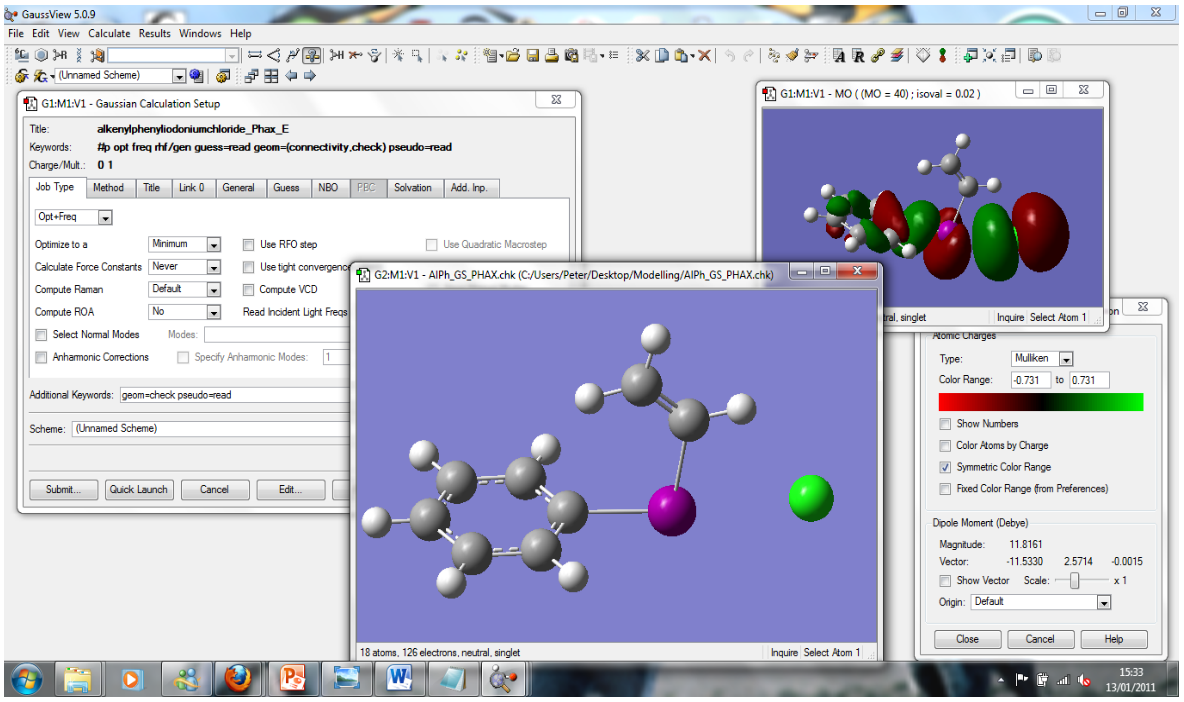
Chemists are now able to investigate the properties of chemical structures at the molecular level. The aim of the research described in this paper is to discover new drugs that targets particular cells and cures diseases at the cellular or genetic level.

This paper describes the performance of a compute cluster applied to solve a large number of molecular models. The paper considers factors such as the computational chemistry and molecular modelling, computational efficiency, task scheduling and the analysis of cluster system performance. The design of an application framework for computational chemistry (AFCC) research is discussed.

The University of Sunderland have recently installed and commissioned a general purpose high performance compute cluster in the Computing department. This development is based upon Microsoft High Performance Computing HPC Servers and the Compute Cluster Pack [1][2]. The design of the cluster computer was completed by University of Sunderland (UoS) and Dell engineers [3][4], refer to Appendix 1. A more detailed description of the cluster system hardware and software configuration may be found in the paper by Ginty [3]. The UoS cluster computer has recently been used to complete some 3D computer graphics (CG) modelling projects [5]. Staff employed within the departments of Computing and Pharmacy collaborated upon this research.

2. MOLECULAR MODELLING

Molecular modelling software allows the user to select atoms from the periodic table and to place them in a three dimensional workspace. In most modelling systems it is possible to build a three dimensional molecular structure by using a colour graphics user interface (GUI), refer to Figure1. The initial position of the atoms is normally determined by the user calling upon common sense and experience. In all cases the actual position that the atoms assume in the real world is determined by the Laws of Physics. Computational chemistry is a general name for computer based algorithms that may be used to solve this type of problem. There are numerous algorithms that may be deployed and normally this involves computing the minimum value of an energy function to find the optimum solution.


GaussView 5.0.9
File Edit View Calculate Results Windows Help
(Unnamed Scheme)
G1:M1:V1 - Gaussian Calculation Setup
Job Type
Method
Title
Link 0
General
Guess
NBO
PBC
Solvation
Add. Inp.
Submit...
Quick Launch
Cancel
Edit...
G2:M1:V1 - AIPh_GS_PHAX.chk (C:/Users/Peter/Desktop/Modelling/AIPh_GS_PHAX.chk)
G1:M1:V1 - MO ((MO = 40) ; isoval = 0.02)
Inquire
Select Atom 1
Close
Cancel
Help
15:33
13/01/2011

Title: alkenylphenyliodoniumchloride_Phax_E
Keywords: #p opt freq rhf/gen guess=read geom=(connectivity,check) pseudo=read
Charge/Mult.: 0 1

Opt+Freq

Optimize to a: Minimum
Calculate Force Constants: Never
Compute Raman: Default
Compute ROA: No
Use RFO step
Use tight convergence
Compute VCD
Read Incident Light Freqs
Use Quadratic Macrostep
Select Normal Modes
Modes:
Anharmonic Corrections
Specify Anharmonic Modes: 1

Additional Keywords: geom=check pseudo=read
Scheme: (Unnamed Scheme)

18 atoms, 126 electrons, neutral, singlet

Atomic Charges
Type: Mulliken
Color Range: -0.731 to 0.731
Show Numbers
Color Atoms by Charge
[✓ Symmetric Color Range]
Fixed Color Range (from Preferences)

Dipole Moment (Debye)
Magnitude: 11.8161
Vector: -11.5330 2.5714 -0.0015
Show Vector Scale: x 1
Origin: Default

Figure1 Molecular Modelling Workspace

For complex structures in many cases the rate of convergence is relatively slow and it is therefore often necessary to employ high performance computing methods to produce solutions in a reasonable period of time.

2.1 Model Convergence Time

There are three principle factors that influence the time required to produce an acceptable solution.

- The initial position of the atoms selected by the user. This initial set of atomic positions is the seed for the numerical solver algorithm embedded in Gaussian_09.
- The number of heavy atoms in the model of the molecular structure.
- The basis set and the method deployed, for example, the molecular mechanics protocol which is intrinsically fast or the ab initio method MP4 that is relatively slow [13,14].

A system that accurately models the Schrodinger Equation will normally require a long time to produce a good solution whereas more approximate methods produce solutions more rapidly.

T is proportional to $K * A^n$

Where

T is the time to produce a solution

K is a constant that is associated with method in use

A is the number of heavy (non-hydrogen) atoms

n is a scalar value of 4

When large numbers of atoms are involved in a model the time taken to generate a solution can be very long.

3. NEW DRUG DISCOVERY

By using HPC methods and computational chemistry it is possible for the user to create a set of new potential drugs. The output data produced by the solver algorithms allows the user to evaluate each potential candidate and select the best for production and further testing. In this scenario it is only necessary to manufacture and test the best candidate molecular structures and the data associated with poor models may be discarded or archived. By evaluating a large number of the most promising models and selecting only the most promising cases it is possible to minimise production and testing costs.

The cost of developing a new drug can run into hundreds of millions or even several billion dollars. The traditional process will usually take 10 or more years and result in the synthesis, purification and biological evaluation of ten thousand or more new compounds in order to produce a single viable clinical candidate.

In order to reduce the costs implicit within the above, the pharmaceutical industry has invested heavily in computational approaches to drug discovery. These approaches often rationalise existing biological data to make informed choices as to the next series of compounds to make, so called Pharmacophore or Quantitative Structure Activity Relationship (QSAR) approaches. Alternatively, where good quality structural information is available regarding the intended biological target molecule, usually a particular enzyme, receptor protein or strand of DNA/RNA, a research team can embark upon a programme of so called Rational Drug Design.

Rational Drug Design comes in several subtypes, the details of which are beyond the scope of this paper. However, they all operate by defining a region of space within the target biomolecule within which the drug candidate must fit. The usual analogy for this is that we are trying to find a specific key (the drug candidate) that fits a particular lock (the drug target). Thus the drug candidate must be the correct size and shape to snugly fit within the available space in order to maximise the potential for interactions between drug and target without being too big to fit at all.

Along with the relatively crude considerations that must be made with regards to a complementary shape between lock and key we must also consider the complementarity or otherwise between the surface electrostatic potentials of the two molecules. All molecules whether natural or synthetic are made from atoms and these atoms are made in turn from three components: protons, neutrons and electrons. The protons which are positively charged and neutrons which are electrically neutral form a dense nucleus within the heart of each atom which is surrounded by a swarm of negatively charged electrons. However, when we put atoms together to make molecules certain atoms attract more of the electron density towards themselves on average than others. This leads to the surface of typical molecules being electron rich and electrostatically negative in some areas and electron deficient or electrostatically positive in others.

If we were to find molecules with the correct shape to fit to our binding site but would place regions of high electron density in contact with one another, Coulomb's Law dictates that the two molecules would repel and thus binding would not be able to take place. Similarly, repulsion would take place between molecules with a significant degree of electrostatically positive contact. Productive binding can thus only take place when the drug and drug target display regions of opposite charge on the electrostatic surfaces that would be in contact with one another when the drug docks into the target binding site.

Many programmes have been developed that are able to assign surface electrostatic potentials of molecules based upon a qualitative knowledge of the properties of their constituent atoms and then attempt to dock them together, and many new compounds in the clinic have been designed with the aid of these techniques. However, the ability of the computer to assign these charges and other properties of the molecules in question is entirely dictated by parameter sets stored within the computer.

In reality, it is known that when two molecules come into contact with one another they perturb the electron distribution within one another. This interplay between the electrostatic potentials of the two molecules in many instances is rather subtle but in some cases can lead to unexpected effects when the two molecules are allowed to bind to one another in real life. Moreover, many drugs that have long been on the market, such as aspirin and penicillin, operate by causing a chemical reaction between themselves and their target molecules alongside the initial binding event. These effects are much more difficult to anticipate and are often impossible to predict using the standard software in the industry based upon the traditional parameterised molecular mechanics (MM) approaches.

Problems such as reactivity and mutual polarisability that occur when two molecules come into intimate contact are however able to be tackled using Quantum Mechanics (QM). Quantum mechanical programmes such as Gaussian operate by finding approximate solutions to the Schrodinger Equation for collections of atoms and molecules using only physical constants such as the masses of subatomic particles and the speed of light. This means that there are no limitations or biases in these calculations from pre-determined parameter sets and the true nature of the interactions and reactions between molecules becomes realisable.

However QM methods are highly computationally resource intensive to the extent that while programmes for carrying out such calculations have been available for many years their implementation in biologically relevant problems such as drug design has not been feasible until recently due to the size and complexity of these molecules. Even now full QM approaches are not deemed to be cost effective for medium or large biological molecules and thus hybrid, or so called QM/MM approaches, have been developed. In a typical QM/MM calculation the drug or natural ligand to be studied is treated quantum mechanically, as are the parts of the target molecule which make up the binding site. The rest of the target molecule is treated using more traditional MM methods.

3.1 Parallel Processing

To identify a new potential drug a large number of jobs must be evaluated. This results in many hundreds of or even thousands of jobs being presented to the compute cluster for processing. Each job takes a finite time from perhaps a few hours up to many days

3.1.1 Compute Cluster Multiprocessing CCM.

If all the resources of the compute cluster are focused upon single job individual solutions are usually produced very rapidly. This method is termed compute cluster multiprocessing CCM. However, the operation of splitting the task between nodes results in an increased communication overhead and normally a reduced level of efficiency.

3.1.2 Symmetric multiprocessing SMP

An alternative approach is to suffer the longer time delay incurred and allocate a single job to each Compute Node for processing. The authors selected symmetric multiprocessing SMP for this work. The SMP approach works in conjunction with a job scheduler so that many jobs can run in parallel. The use of the scheduler ensures that there is no delay between the allocation of new jobs to Compute Nodes. SMP techniques that optimise the use of CPU cores and available RAM have been employed to ensure that CPU usage of 100% is normally achieved at each Compute Node.

4. COMPUTATIONAL CHEMISTRY AND GAUSSIAN

Professor John Pople designed the first version of the Gaussian computer program to make his computational techniques easily accessible to researchers. The use of Gaussian makes possible the theoretical study of molecules, their properties, and how they act together in chemical reactions. The Gaussian program has been continuously developed by many researchers and today it is a software product that is used by thousands of chemists. Professor John Pople was awarded the Nobel Prize in chemistry in 1998 for his pioneering contributions in developing computational methods. Researchers can use Gaussian_09 to investigate the following topics [8].

- Comprehensive investigation of molecules and their reactions
- Predicting and interpreting spectra
- Predicting optical spectra including hyperfine spectra
- Investigate thermo chemistry and excited state processes

- Gaussian_09 allows solvent effects to be taken into account when optimising structures and predicting most molecular properties
- Modelling NMR

In the past chemists used standard laboratory methods to undertake research, for example by using chemical compounds, solvents, test tubes and heat sources. It is now possible to complete much of this practical research work by using powerful computers and computational chemistry software. The application of computation chemistry has many advantages mainly by helping to reduce the need for access to expensive laboratory time, test equipment and chemicals.

From experience the authors have found that it takes about five minutes to manually prepare a single Gaussian_09 job to be sent to the scheduler prior to execution on the cluster computer. The manually preparation of jobs for execution is a standard approach often adopted by many research groups. This approach is most appropriate where a relatively small number of jobs are being processed by the scheduler, for example less than one hundred jobs.

The authors of this paper are working with a group of six research chemists working upon drug development. These researchers have produced large number of jobs in batches of various sizes for execution on the cluster computer. A single batch contains typically between one hundred and five hundred Gaussian_09 jobs. It is anticipated that the total number of jobs that will eventually be processed on the cluster will be of the order ten thousand for any given development. To date more than six thousand jobs have been completed. In this paper the initial two hundred and seventy six jobs have been analysed to determine the performance characteristics of the cluster computer. The development of the AFCC allows a large batch of jobs to be easily submitted to the cluster normally within a few minutes thereby saving valuable time.

By using the computational chemistry techniques researchers are able to carry out a series of experiments on a particular molecular structure (compound or solvent) in a manner similar to that employed in a conventional laboratory facility. AFCC employs a scheme where ‘the human is in the loop’.

In a typical scenario the researcher inspects the output results obtained for chemical structure of interest and determines the direction of future experimentation. A range of factors are taken into account. The most important factor is usually the molecular structure which can be inspected and manipulated in three dimensions by using GaussView GUI. By inspecting the results it is also possible to inspect the bonds between atoms, electron transition states, spectra and photo luminescence effects when a compound is mixed in a solution.

In AFCC there is no limit to the number of experimental steps that may be deployed. To date the number of successive experiments that have been made on a single compound is between two and six. A naming and numbering system has been devised to ensure that data sets are clearly grouped together to assist in the organisation and analysis of results.

At present it has proven to be impossible to embed an intelligent search algorithm within the AFCC to direct a series of experiments. The authors have been informed that for current type of research work being undertaken at the UoS the intermediate results of the last experiment must be evaluated by a researcher to determine the direction of the next experiment.

A group based in Australia [6] have developed a framework that enables a genetic algorithm to direct a series of experiments using Gaussian_09. The authors have investigated the possible application of an intelligent search algorithm embedded (PSO) within the AFCC [7]. However it has not been possible to identify an appropriate fitness function for the reason outlined above. In addition a typical GA requires a large number of iterations and associated potential solution evaluations. As the average Gaussian_09 job run time is about fifteen hours at this time the authors do not consider it feasible to apply an intelligent search algorithm for the range of experiments currently being investigated.

A team based in Illinois (United States) have employed multi-objective genetic algorithms to optimise semi empirical chemistry models based on excited state photo dynamics for material applications (e.g., LCD and LED), pharmaceuticals and chemical manufacturing [14].

4.1 GaussView

GaussView provides features for studying large molecular systems, such as importing molecules from input files, modifying structural features and setting up calculations in Gaussian_09, viewing and plotting the final results. In addition GaussView can also import many other structure exchange formats.

GaussView allows the user to build a 3D molecular structure by using a colour graphics user interface. A file (.gjf) is produced by GaussView that may be sent to the input of the Gaussian solver program [9].

4.2 Gaussian_09

Gaussian_09 is designed to model a broad range of molecular systems under a variety of conditions. All computations performed start from the basic laws of quantum mechanics. Theoretical chemists can use Gaussian_09 to carry out basic research in established and emerging areas of interest.

Experimental chemists can use it to study molecules and reactions of potential interest, including both stable structures and those compounds which are difficult or impossible to observe experimentally such as short-lived transition structures.

Gaussian_09 can also predict energies, molecular structures, vibrational frequencies and numerous molecular properties for systems in solution and the gas phase. In addition it can model structures in both their ground state and excited states. Furthermore research chemists can apply these fundamental results to their own investigations, to explore chemical phenomena such as reaction mechanisms and electronic transitions.

The Gaussian package is a suite of programs and the 32bit multicore version of Gaussian_09 has been installed on all of the Cluster Compute Nodes [8]. The package includes a number of utility programs. These programs may be deployed to convert binary data in plain readable text or into a 3D visualisation of the molecular structures being investigated.

Gaussian_09 takes as an input the (.gjf) file produced by GaussView and runs the analyser to produce a solution to the problem. A typical command line for the solver is given below.

```
G09.exe input_file.gjf output_file.out
Checkpoint file: output_file.chk
```

An output text file (.out) is created that can be that can be read and inspected by a human. In addition, a checkpoint file (.chk) is also produced that may be processed by a computer to produce further detailed information.

5. APPLICATION FRAMEWORK FOR COMPUTATIONAL CHEMISTRY (AFCC)

During initial testing jobs were manually submitted to the scheduler and the time taken to load each job was typically between five and nine minutes. However it rapidly became apparent that for a very large number of jobs it would be necessary to develop a more efficient system to complete six main tasks.

- (i) to prepare batches of Gaussian files (.gjf) so they could run efficiently on the cluster computer
- (ii) to submit batches of (.gjf) files to the scheduler under program control
- (iii) to run the G09 analysis program on a Compute Node, also to setup the local environment and transfer I/O files
- (iv) to archive the set of files associated with each job so that they can be indexed, dated and searched
- (v) to analyse and extract specific textual data from the output files (.chk) under program control
- (vi) to prepare a batch of input files so that they may be resubmitted to the scheduler program

A set of C Sharp (C#) programs have been developed to satisfy these goals. The system that has been developed has been given the name an Application Framework for Computational Chemistry (AFCC), refer to Figure2. It is anticipated that many thousands of new jobs will be processed in the near future using the AFCC.

Sudholta and Buyya1 have also developed an application frameworks for molecular modelling [10][11][12] and both systems are based upon grid computing methods.

G09prep (i) this program is used to modify the input file so that it is compatible with the compute cluster requirements by defining various input/output data paths and the size of RAM for program execution.

G09submit (ii) this program is used to submit a job to the job scheduler. The parameters passed to this module determine where the input file is stored and the final location of the output file. This C# program code constructs the various UNC paths required by the system.

G09local (iii) this program is used to set up environmental variables on a Compute Node and to invoke the g09 executable. The program g09.exe is installed on all Compute Nodes. All input and output files reside in a storage area associated with the cluster HeadNode. During a compute run all input files are pulled down from the HeadNode to the Compute Nodes and the output results files are pushed back up to the HeadNode filestore. An advantage of this scheme is that there is no need to permanently store any Gaussian input or output files on the Compute Nodes.

G09archive (iv) this program takes batches of Gaussian job files and moves them into a central archive created on the main direct attached disk storage unit (DAS). The work space on the Head Node is also purged at the end of a batch run.

G09analyse (v) this program uses a text based search to extract relevant information from the output file. As these files are all very large this program significantly helps to reduce the time taken to extract relevant information from the large set of files, for example, hundred files each of size about 20MB.

G09analyse has virtually eliminated the need to manually load a large file into a text editor and search for specific information. Experience has shown that manual searching can be a very difficult and time consuming process.

At the end of a batch run all jobs are archived in an indexed central storage area and work areas on the cluster purged to release space on disk drives. An FTP server has also been installed so that remote users may download the results produced by the Gaussian program.

G09rerun (vi) this module is used to prepare and resubmit a job to the cluster. The module G09rerun is a modified version of G09prep. Normally there are two reasons why it is necessary to resubmit a job to the cluster.

- Case 1 - A job may run for a long time and then fail to complete because a parameter value is not within an allowed range. For example, a typical job may require five or more processing stages to be completed. A job may run for four days and then fail at the last stage. In this case it is possible to rerun the job starting from stage four rather than stage one. This approach can often avoid many days of unnecessary duplicated processing.

- Case 2 - In this case a job may have run to successful completion. After inspection of the results the researcher may wish to determine how the candidate drug reacts with another agent.

In both of the above cases the new information required to restart and control the analysis process is embedded in a modified version of the (gjf) input file. In addition the (chk) check file (created during the first run) must also be sent to the G09 scheduler to enable the analysis to restart from an intermediate stage. The role of module G09rerun in the (AFCC) framework is shown in Figure 2.

It is possible to use the rerun process a number of times to investigate the properties of the candidate drug. This approach helps to facilitate incremental drug development and minimise the time required to run a series of interrelated jobs.

In the AFCC the initial path taken by a typical G09 job is shown by grey arrows whereas a job that is resubmitted follows a path described by the white arrows, refer to Figure2.

FTP Server - this server has been installed on the cluster to enable researchers working remotely to upload jobs and to download the results obtained from use of the AFCC.

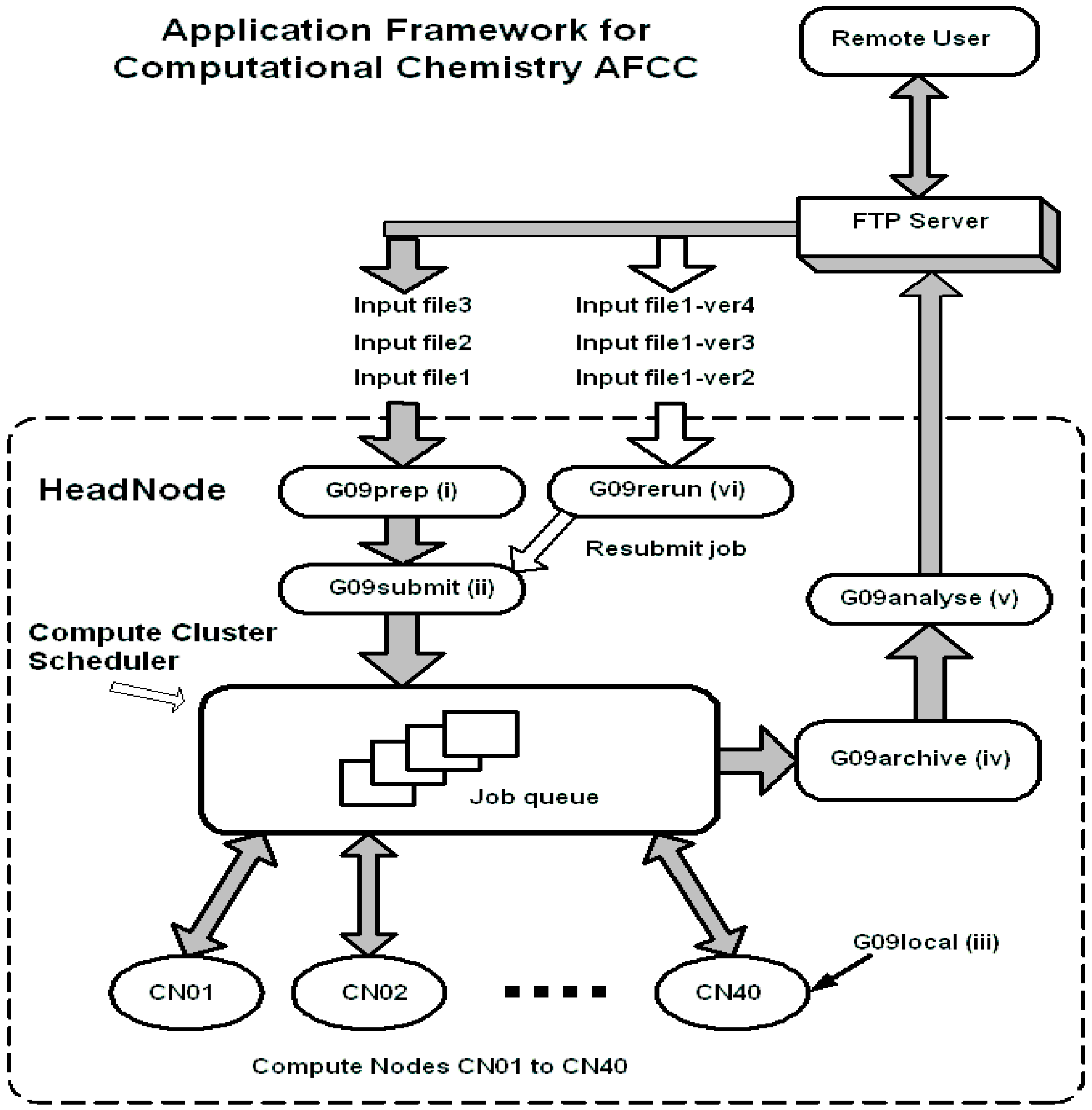


Figure 2. Application Framework for Computational Chemistry (AFCC)

6. RESULTS

Jobs were submitted to the cluster computer and various different batch sizes have been processed. The overall efficiency of the cluster computer and the batch process being employed is given in Table 1. The data set (Series1) was analysed to produce the information shown in Figures 3 and 4. A graph showing the execution time against the job number is shown in Figure 3. By inspecting the graph it can be seen that the maximum job time is greater than 60 hours. The average time has been calculated to be 15.6 hours. Figure 4 shows a plot of job time frequency versus job time with a bin size of 2 hours.

Comment	
Total number of jobs submitted	276
Total number of jobs successfully completed	247
Efficiency = job completed/ jobs submitted * 100	89%
Jobs that failed	11%
Average job time	15.6 hours
Maximum job time	>60 hours

Table 1 Overall Batching Processing Results

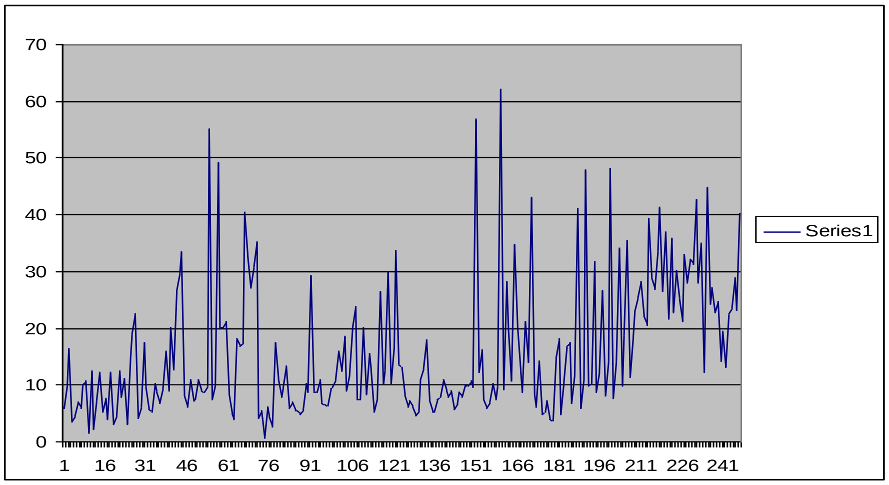


Figure 3 Job Time (Hours) versus Job Number

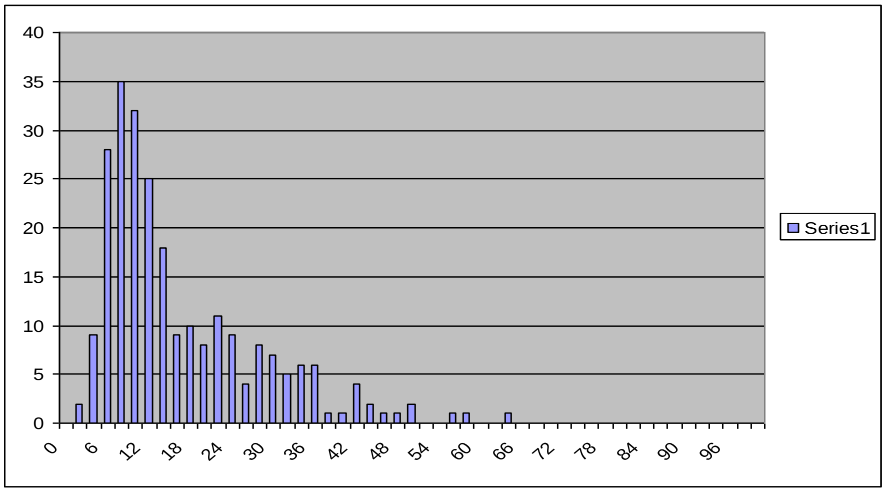


Figure 4 Job Time Frequency versus Job Time (Hours)

A total of 276 jobs were submitted for processing and 89% were successfully completed. A total of 11% failed to run successfully. Further analysis of the failed jobs produced the following results.

Comment	**Percentage**
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%

Table 2 Analysis of Failed Jobs

As time progresses the authors are gaining more experience of molecular modelling batch processing. Consequently as the number of batches processed slowly increases the number of failed jobs is gradually decreasing. This improvement can be attributed to input data files with a smaller number of errors and better initial seed values. In most cases it is possible to correct the error in the input data file and add the job to the next batch to be submitted to the scheduler.

To solve a molecular modelling problem a number of distinctly separate processing steps must be completed. In some cases it is possible to resubmit a failed job at an intermediate stage to reduce to minimise the time required to produce a solution.

Structures that have very similar levels of complexity can require very different execution times. Unfortunately, at the outset it is not normally possible to predict the time required to produce a solution. Furthermore during the execution of a job it is not normally possible to determine how much longer is required to produce a good solution. As a result it is difficult to decide upon the best criteria for termination of the solver. The authors often terminate a process if a solution is not found within five days.

6.1 Job Execution Time Variation

Further tests were carried out to determine if there was any variation in the time taken to solve a job by submitting three different jobs many times to the scheduler. The jobs selected for this test were of short duration, refer to Table 3. These test files were copied twenty times, renamed to avoid duplication and three different groups sent to the scheduler. The initial seed values were not modified for this test.

Test Name	Average Run Time hours	Number of runs	Standard Deviation %	Output File Size MByte approx.
Test01	0.67	20	<0.1	20
Test02	2.58	20	<0.1	20
Test03	3.00	20	<0.1	20

Table 3 Run Time Repeatability

In Table 3 the standard deviation is expressed as a percentage of the average runtime. The results presented show that there is no significant variation in the time taken for the cluster to process a particular job. Any variation in the time taken to complete a job could be attributed to network congestion at the time when the output files are written to the Head Node disk store. The authors concluded that the network congestion does not adversely influence the job run time to any great extent.

6.2 Job Execution Time using different Computer Systems

A series of tests were carried out to compare the performance of the Compute Nodes with other machines that could have been employed. The results obtained from this test are shown in Table 4.

Machine	CPU	CPUs	Cores	Clock GHz	Ram GB	R
Compute Node	Dual core Xeon	2	4	2.66	8	1
Workstation1	Dual core	1	2	2.66	4	2
Workstation2	Single core	1	1	1.0	1	4

Table 4. A Comparison of Run times for different Computers

The relative processing times for the machines list in Table 4 is given the column R. These values are rounded and normalised after taking the clock frequency into account. By inspecting Table 4 it can be seen that computational performance of a Compute Node is much better than that of Workstations 1 or 2. It was found that the time taken to run a job is roughly proportional to the number of CPU cores deployed. The results were not markedly influenced by the amount of RAM allocated for the job.

6.3 Job Execution Time using various RAM Sizes

Further tests were carried out on a Compute Node to determine how the number of CPU cores and the amount of RAM allocated for job execution influenced the run time. The authors carried out experiments to determine the parameter setting that would maximise the processing efficiency.

6.3.1 A Short Job (<1 hour)

The results of these tests are shown in Figure 5 showing the run time in hours versus the amount of allocated RAM. For this particular case where the run time was quite short (average time 0.67 hours) the selected the amount of RAM allocated does not have a significant effect on the overall job execution time.

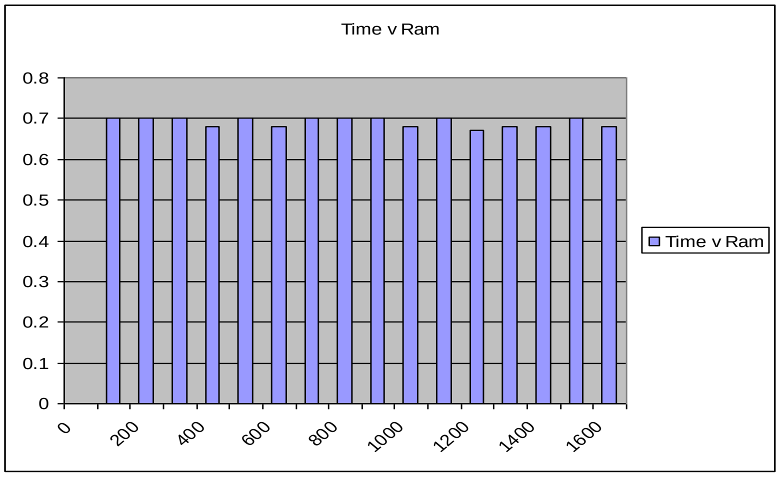


Figure 5. Job Run Time (Hours) versus Allocated Ram (MBytes)

6.3.2 A Long Job (>20 hours)

Figure 6 again shows how the job run time varies as the amount of allocated RAM is varied. In this case the run time was greater than 20 hours. The results show that the best performance is obtained when the amount of RAM is set to 1400MByte. In this state the total amount of RAM in use is about 550MB for operating systems plus 1400MB for Gaussian_09 which is just less than the recommended value of 2GB for a 32bit system.

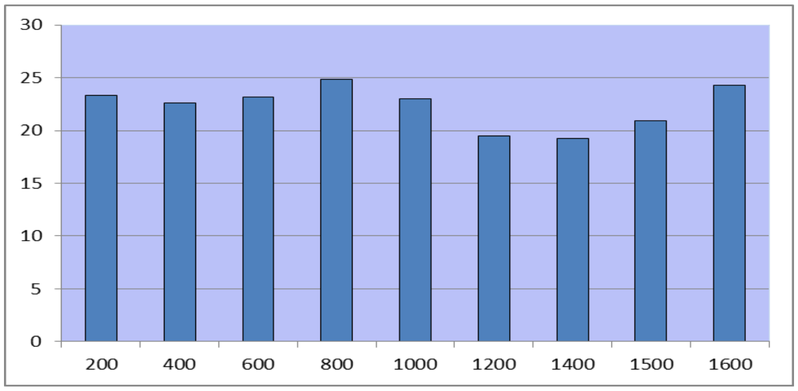


Figure 6 Run time (hours) versus Ram Size (MByte)

6.3.3 Varying the number of CPU Cores

Figure 7 show the job execution time in hours versus the number of CPU cores selected to execute the job in a Compute Node. The benefit of employing a multi-core processor with support for symmetric multiprocessing for molecular modelling is clearly demonstrated by this data.

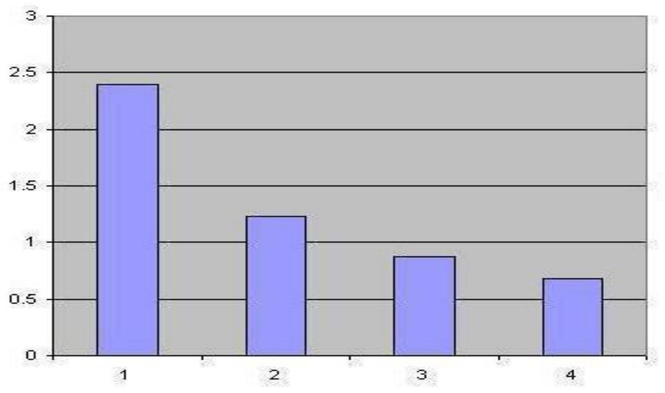


Figure 7 Job Execution Time (Hours) versus Number of CPU Cores

7. JOBS AND MANUAL PROCESSING

Consider the scenario when a large number of tasks are manually started during a normal working week without using a job scheduler. During the working day of eight hours jobs can be started however some will terminate during the day and others during the night. The probability of the termination is assumed to be evenly distributed across time. A large number of jobs will terminate during the night and the next job will not start execution until the following morning. If it is assumed that jobs are continuously monitored during the day and the next job started without any delay, it is possible to estimate the time wasted by the approach, refer to Table 4.

Working day	8 hours
Working week	7 days
Time jobs monitored	8 hours
Time jobs not monitored	16 hours
Number of jobs processed	700
Jobs ending during night	700*16/24
Average delay between jobs	8*16/24 = 5.3 hours
Wasted time per job	16/2 = 8
Average job time	15.6 hours
Time wasted	700*0.66*8 = 3733 hours = 155 days
Total processing time	700*15.6 hours = 10920 = 455 days
Total time days	610 days = 1.67 years

Table 4 Time Wasted using Manual Job Control

In this example of the total time required to complete all jobs 75% is spent processing data and 25% waiting. In reality the wasted time may be even greater because jobs are not normally continuously monitored and this analysis does not take the weekend break from work into account. It is very clear that a considerable period of time is saved by using the AFCC developed by the authors.

The average run time for a Gaussian_09 job is 15.6 hours using the USCC. During a job execution about 20MByte of output data is sent across the network to the Head Node in many short bursts. The time taken to transfer the output data is very short in comparison to the run time, typically measured in seconds compared to 15.6 hours. The authors have therefore concluded that the latency due to network traffic is not a significant factor with regard to the overall efficiency of the AFCC.

8. CONCLUSIONS

Prior to the use of AFCC on the cluster computer a single much less powerful machine was employed to process molecular modelling jobs. In all cases jobs were manually submitted for processing without using a task scheduler. The typical run time for a single job was between one and two weeks.

By using the cluster computer it is now possible to produce overnight (average run time 15.6 hours) results that would have previously required a run time of more than forty weeks or more.

The AFCC framework developed by the authors has been proven to be reliable providing support for the following operations within a molecular modelling job: prepare, submit, run, archive, analyse and rerun.

The massive computational speedup of the cluster computer (USCC) has now been realised via the AFCC enabling researchers to significantly reduce the cycle time required to design, model and evaluate molecular structures for drug development.

Symmetric multiprocessing SMP methods have been successfully applied using a cluster computer in the domain of Computational Chemistry to efficiently and successfully solve a large number of molecular modelling problems.

The use of the AFCC framework has proved to be successful and researchers are now able to prepare and schedule large numbers of molecular modelling jobs for the cluster within a short period of time (a few minutes). The total time required to process a large batch, take for example a batch of one thousand jobs will be typically around fifteen days. A batch of this size produces a large amount of output data in the form of plaintext as well as typically one thousand check files (.chk) with a total size of the order 20GByte. In the future it is planned that a large number of batches of this size will be submitted for processing.

The authors anticipate that it may be necessary to further develop the G09analysis and G09rerun modules to optimise performance and thereby reduce the time required to analyse and identify additional important features in the data sets. Providing a high level of feedback from the analysis of large process batches will also enable researchers to gain a better insight into why jobs take so much time to process. Applying this knowledge to new designs will help to maintain a high success rate while helping to lower future job processing times. In addition, the G09rerun module helps to facilitate incremental drug development by minimising the need for duplicate analysis.

The authors argue that the AFCC framework may be adapted and applied successfully particularly in other domains where the number of jobs is relatively large (>100) and the analysis time is greater than the time associated with network communication.

Chemists are now able to investigate the properties of chemical structures at the molecular level. The primary aim of this research is to discover new drugs that targets particular cells and cures diseases at the cellular or genetic level, such as multiple sclerosis and cancer, without the serious side effects associated with some standard treatments. The research described in this paper makes a valuable contribution towards this important goal.

9. ACKNOWLEDGEMENTS

The Science Research Investment Fund (SRIF) is a joint initiative by the UK Office of Science and Technology (OST) and the Department for Education and Skills (DfES). The purpose of SRIF is to contribute to higher education institutions' (HEIs) long-term sustainable research strategies and address past under-investment in research infrastructure. The University of Sunderland Cluster Computer was purchased with the support of the SRIF III fund – see Appendix 1.

10. REFERENCES

1. Microsoft High Performance Computing HPC Server, http://www.microsoft.com/hpc/en/us/product/cluster-computing.aspx
2. Microsoft Compute Cluster Pack, http://msdn.microsoft.com/en-us/library/cc136762%28v=vs.85%29.aspx
3. K. Ginty, J. Tindle, S.J. Tindle, Cluster Systems – An Open-Access Design Solution, International Conference on Systems Engineering (ICSE) 8-10 September 2009, Venue: Coventry University
4. Eco_Friendly Super Computing, Dell Case Study, http://www.dell.com/downloads/global/casestudies/798_Sunderland%20University.pdf
5. J. Tindle, K. Ginty, S.J. Tindle, Rendering 3D Computer Graphics on a Parallel Computer, International Conference on Systems Engineering (ICSE) 8-10 September 2009, Venue: Coventry University
6. Brain Z and Addicoat M, Using Meta-Genetic algorithms to tune parameters of Genetic Algorithms to find lowest energy Molecular Conformers, Proc. of the Alife XII Conference, Odense, Denmark, 2010 pp 378-385
7. Vigneshwaran Namasivayam, Robert Günther, A Fast Flexible Molecular Docking Program Based on Swarm Intelligence, Journal of Chemical Biology and Drug Design. 2007 Dec;70(6):475-84. Epub 2007 Nov 6.
8. Gaussian_09 Reference Manual, http://www.gaussian.com/g_tech/g09w_ref.htm
9. GaussView 5 Reference Table Contents, http://www.gaussian.com/g_tech/gv5ref/gv5ref_toc.htm
10. Wibke Sudholta, Kim K. Baldridgea, David Abramson, Colin Enticott, Slavisa Garic, Chris Kondric, Duy Nguyen, Application of grid computing to parameter sweeps and optimizations in molecular modelling, Future Generation Computer Systems, Volume 21, Issue 1, 1 January 2005, Pages 27-35
11. Rajkumar Buyya1 and Manzur Murshed, GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing, Concurrency and Computation: Practice and Experience, Concurrency Computat: Pract. Exper. 2002; 14:1175–1220 (DOI: 10.1002/cpe.710)

12. Rajkumar Buyya1, KimBranson, Jon Giddy and David Abramson, The Virtual Laboratory: a toolset to enable distributed molecular modelling for drug design on the World-Wide Grid, Concurrency and Computation: Practice and Experience, Concurrency Computat.: Pract. Exper. 2003; 15:1–25 (DOI: 10.1002/cpe.704)
13. J. G. Vinter, A. Davis and M. R. Saunders, Strategic approaches to drug design. I. An integrated software framework for molecular modelling, Research Papers, Journal of Computer-Aided Molecular Design, Volume 1, Number 1, 31-51, DOI: 10.1007/BF01680556
14. Kumara Sastry1, D. D. Johnson, Alexis L. Thompson, David E. Goldberg, Todd J. Martinez, Jeff Leiding, and Jane Owens, Optimization of Semiempirical Quantum Chemistry Methods viaMultiobjective Genetic Algorithms: Accurate Photodynamics for Larger Molecules and Longer Time Scales, Materials and Manufacturing Processes, 22: 553–561, 2007, ISSN: 1042-6914 print/1532-2475 online, DOI: 10.1080/10426910701319506

APPENDIX 1 UOS CLUSTER COMPUTER OUTLINE SPECIFICATION

UoS Cluster Computer Head and Compute Nodes

- Number of compute nodes: 40
- Head nodes (Linux/Windows): 2
- Compute nodes based upon Dell Server type 2950
- Head node boots Win2003 Server HPC version (64bit)
- Central processor unit Xeon 5100 64bit, 2.66GHz,
- Number of CPU processor per node: 2
- Number of CPU cores per node: 4
- Total number of compute cores in the cluster: 160
- RAM per node: 8Gbyte, per CPU core: 2Gbyte
- Network bandwidth Gigabit Ethernet 1Gbps
- Networks - Data, IPC and IPMI

Main Switch Unit

- Main switch type Cisco 6509
- Number of line cards: 3
- Line card type 48 port 10/100/1000Mbps
- Total number of 1Gbps ports: 144

Central Data Storage

- Central storage 15 * 500Gbyte disk, 8 Linux and 7 Windows
- Total central storage on Dell MD1000 DAS: 7.5Tbyte
- Bandwidth 2 * 12Gbps

Operating Systems

- All compute nodes support dual boot
- Scali Manage provides support for managing the cluster, monitoring performance and control of services
- Control via Linux Parallel Shell or Windows Grid Administrator