Performance Prediction and Procurement in Practice: Assessing the Suitability of Commodity Cluster Components for Wavefront Codes

S.D. Hammond, G.R. Mudalige, J.A. Smith, S.A. Jarvis,
High Performance Systems Group,
Department of Computer Science,
University of Warwick,
Coventry, CV4 7AL, UK.

{sdh, g.r.mudalige, jas, saj}@dcs.warwick.ac.uk

Abstract

The cost of state-of-the-art supercomputing resources makes each individual purchase an expensive and, in many cases, lengthy process. Often each candidate architecture will need to be benchmarked using a variety of tools to assess potential performance. However, benchmarking alone often provides only limited insight into the potential scalability and suitability of each architecture for procurement.

In this paper we present a case study applying two recently developed performance models to the Chimaera benchmarking code written by the United Kingdom Atomic Weapons Establishment (AWE) with a view to analysing how the code will perform and scale on a medium sized, commodity based InfiniBand cluster. Our models are validated with average accuracies of 90% against an existing InfiniBand machine and then used as the basis for predicting code performance on a variety of hardware configurations including changes in the underlying network, faster processors and high core density per processor.

The results of our experimentation with machine performance parameters demonstrate the compute-bound nature of Chimaera and its sensitivity to network latency at increased processor counts. By using these insights we are able to discuss potential strategies which may be employed during the procurement of future mid-range clusters for a wavefront-code rich workload.

1 Introduction

Modern supercomputing resources are constantly evolving. Where once a 'supercomputer' may have been a shared memory machine comprising of tens of processors housed in a single structure, today supercomputing resources commonly utilise multiple sub-structures such as cabinets, multiple-processor nodes and more recently multiple-core processors. When combined with the complex network interconnects found in modern systems, identifying and analysing the performance properties of the machine as a whole becomes a significant challenge. With the growing core counts

of modern machines and the ever increasing complexity of each system the task of procuring the 'right' computing machinery for purpose is fastly becoming a lengthy and intricate process. Pure benchmarking of applications on candidate architectures serves only limited purpose - the results will only highlight the performance of specific codes and often only for specific inputs. For organisations who want the very best machine performance, a deeper knowledge of code behaviour with respect to each prospective platform is needed.

Performance modelling has been used as a basis for machine comparison [8, 14] and post-installation performance verification [15], and has been shown in a number of examples to address many of the questions which may arise during procurement. Whilst serving as a showcase for many performance modelling techniques, the focus has been on very large emerging architectures and not the small to medium sized commodity or near-commodity clusters used in a number of research organisations. In these procurement activities similar issues must be addressed but with hardware which may have lower specification, be arranged differently or have alternative behaviour to the expensive components that are common place in supercomputing systems.

In this paper we utilise two recently developed performance models to explore the performance of the Chimaera neutron transport benchmark developed and maintained by the United Kingdom Atomic Weapons Establishment (AWE), targetting a processing element count of up to 4096 cores. The direct use and cross-comparison of predictions from two performance modelling techniques aids not only in elucidating specific code and machine behaviour but also in increasing the accuracies of our observations. This work is not intended to comment on the respective costs of each strategy but to provide some degree of quantitative exploration of various hardware and application configurations, which can in turn support the queries that may arise during the early stages of a procurement activity. The specific contributions of this work are:

- The presentation of a performance study for the AWE Chimaera benchmark on commodity or near-commodity hardware. This is the first such study for the Chimaera benchmark and is designed to support future procurement activities for mid-range supercomputing resources at AWE. We use two approaches in verifying our predictions: (1) based on analytic methods utilising the recently developed "plug and play" reusable wavefront model [18] and (2) using a new discrete even simulation toolkit. Both approaches show predictive accuracies of over 90% and provide higher confidence in the conclusions obtained from our performance engineering study.

- A quantitative exploration of the key parameters which affect the performance of wavefront codes on modern commodity HPC systems, supporting the exploration of prospective machine configurations for procurement.

- An exploration of the contention costs arising on a CMP-processor-based cluster when executing Chimaera and the implications for code runtime and machine procurement.

The remainder of this paper is organized as follows, Section 2 provides a brief overview of the two main approaches to application modelling - analytical studies and simulation. We introduce the Chimaera benchmark in Section 3 continuing our discussion in Section 4 by describing the development of two performance models using analytical techniques and a new simulation-based toolkit. Sections 5 and 6 contain our case study

in which we benchmark an existing 11.5 TFLOP/s InfiniBand system and project runtimes for a variety of alternative application and machine configurations. Our paper concludes in Section 7 with a summary of the results and a review of the implications for procuring a small to medium size cluster for sustained wavefront-dominated computations.

2 Performance Modelling

Application performance modelling is principally charged with the derivation of models by which code behaviour can be analysed and predicted. In the main, the interest in such models is in analysing how the computational and communication structures in a code will change with respect to an increased processor count or change in application problem size. By developing a deeper insight of the runtime fluctuations resulting from such changes, an understanding of code bottlenecks, software optimisation and in many cases optimal configuration can be developed.

Current techniques for developing application models predominantly fall into two distinct categories - those based on analytical studies and those based on simulation. Although some conceptual work on a binding of the two is discussed in the POEMS framework [1], there has been little practical demonstration reported in academic literature. Analytical studies [11, 13, 21] which seek to represent code behaviour by a series of mathematical formula, are often developed within some modelling framework or methodology (e.g. LogP[4], LogGP[2] and LoPC[5]). The use of rigid frameworks for modelling helps to alleviate some of the complexity involved in modeling and provide a generic basis upon which code behaviour can be judged. The challenges of using an analytical approach are identifying the key application parameters which affect runtime behaviour and how best to represent each parameter mathematically. The analysis of code for modelling is often based on manual code inspection which, although time consuming, allows the performance modeller to develop a deeper understanding of specific code behaviour from which further behavioural insights may be garnered.

A brief overview of the recently developed “plug and play” reusable wavefront model [18], which serves as the basis for our analytical exploration of Chimaera, is presented in Section 4.1. Note that the development of a reusable model serves to reduce the time required to model future wavefront codes, since a flexible framework can now be applied to any wavefront application; this approach also permits cross-application comparisons to be made within a highly algorithm-specific framework.

Simulation-based performance systems (*e.g.* Wisconsin Wind Tunnel [19], PROTEUS [3] and the PACE toolkit [7, 12]) were originally envisaged as a mechanism to lower the burden of performance modelling by eliminating the need to manually inspect application source code. The automated replay of applications either in source or binary form allowed developers and performance modellers alike to experiment with performance by making direct changes to the application and simulating execution without requiring direct access to the specific machine in question. In practice, the simulation environments developed to date have attempted to directly simulate individual application instructions making the simulation of large industrial codes infeasible in realistic time frames. When the increase of modern application complexity is compounded with increasing core counts of emerging cluster platforms, the use of simulation quickly becomes intractable as a source of fast and efficient performance evaluation. In Section 4.2 we present the development of a prototype simulation toolkit which seeks to overcome some of the problems discussed, in particular the use of coarser grained

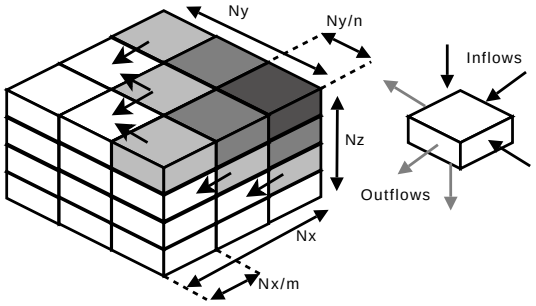


Figure 1: Sweep execution through the data array in Chimaera.

computational timings (as opposed to individual instructions timings) and a 'layered' network modelling system, significantly reduce simulation times, whilst providing prediction accuracies commensurate with leading analytical models.

3 The Chimaera Benchmark

The Chimaera benchmark is a three-dimensional neutron transport code developed and maintained by the United Kingdom Atomic Weapons Establishment (AWE). On first inspection the code shares a similar structure with the now ubiquitous Sweep3D application described in numerous analytical performance studies [13, 14, 17]. Unlike Sweep3D, however, the code employs a different internal sweep ordering and utilises a complex convergence criteria to decide when execution is complete. In this section of the paper we present a concise description of the wavefront algorithm employed by both Sweep3D and Chimaera. Our discussion is purposefully brief as a number of existing works describe the behaviour of the wavefront algorithm [16] and a short overview is sufficient to enable an understanding of the key application behaviours.

3.1 The Generic Wavefront Algorithm

The generic three-dimensional wavefront algorithm operates over a data array of size $N_x \times N_y \times N_z$. The data array is decomposed over a two-dimensional processor array sized $m \times n$. Each processor receives a 'column' of data sized $N_x/m \times N_y/n \times N_z$. For the purposes of our discussion it helps to consider this column as a stack of N_z tiles each $N_x/m \times N_y/n \times 1$ in size. The algorithm proceeds by executing *sweeps* through the data which pass from one vertex to its opposite. For Chimaera and Sweep3D eight sweeps are used - one for each vertex of the three-dimensional space.

A sweep originates at a vertex of the processor array (the origins of each sweep for Chimaera are shown in Figure 2). The computation required to solve the first tile in the originating processor's stack is completed and boundary information is exchanged with the two neighbouring processors. Once exchanges are complete the two neighbouring processors solve the first tile in their stack whilst the originating processor solves its second tile. On completion, boundary information is again passed downstream to neighbouring processors. A sweep completes once all tiles in the last processor have been solved. Figure 1 shows a partially complete sweep with dark grey tiles having been solved in previous stages, light grey tiles are currently executing and white tiles are awaiting boundary information from upstream processors (arrows are used to show

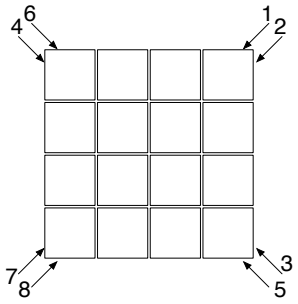


Figure 2: Starting locations for sweep within the two-dimensional processor array employed by Chimaera.

visible communications to downstream processors). A full 'iteration' of the wavefront algorithm in Chimaera requires all eight sweeps to have completed.

4 Modelling Chimaera

The modelling of Chimaera has been conducted using two approaches - analytical modelling based on the "plug and play" reusable model [18] and using the new WARwick Performance Prediction (WARPP) simulation toolkit developed by the University of Warwick.

4.1 Plug and Play Analytical Model

Model Parameter	Chimaera Value
N_x, N_y, N_z	Input size
W_g	$measured$
$W_{g,pre}$	0
$H_{tile}(cells)$	1
n_{sweeps}	8
n_{full}	4
n_{diag}	2
$Message$ $Size_{EW}$ (Bytes)	$8H_{tile}$ $\times \#angles$ $\times N_y/m$
$Message$ $Size_{NS}$ (Bytes)	$8H_{tile}$ $\times \#angles$ $\times N_x/n$

Table 1: Reusable Wavefront Model Application Parameters.

The "Plug-and-play" reusable wavefront model developed in [18] represents the culmination of three individual application performance studies for the Sweep3D, Chimaera and NAS-LU benchmarks. By using the insights obtained in modelling these three wavefront codes, Mudalige, Vernon and Jarvis have extracted and abstracted the common parameters (shown in Table 4.1) which affect application runtime into a generic

$W_{pre} = W_{g,pre} \times H_{tile} \times N_x/n \times N_y/m$	(r1a)
$W = W_g \times H_{tile} \times N_x/n \times N_y/m$	(r1b)
$StartP_{1,1} = W_{pre}$	(r2a)
$StartP_{i,j} = max(StartP_{i-1,j} + W_{i-1,j} + Total\_comm_E + Receive_N,$ $StartP_{i,j-1} + W_{i,j-1} + Send_E + Total\_Comm_S)$	(r2b)
$T_{diagfill} = StartP_{1,m}$	(r3a)
$T_{fullfill} = StartP_{n,m}$	(r3b)
$T_{stack} = (Receive_W + Receive_N + W + Send_E + Send_S$	(r4)
$+W_{pre})N_z/H_{tile} - W_{pre}$	(r4)
$Time\ per\ iteration = n_{diag}T_{diagfill} + n_{full}T_{fullfill}$	(r5)
$+n_{sweeps}T_{stack} + T_{nonwavefront}$	(r5)

Table 2: Plug-and-play LogGP Model: Single Core Per Node.

model. The computational time required, W_g, and the computational time per cell prior to the algorithm kernel, $W_{g,pre}$, are the only machine specific values for which benchmarking of the application is required.. For our study this was obtained by using a manually instrumented version of the benchmark which times the core computational kernel of the wavefront algorithm. $W_{g,pre}$ is unused in Chimaera since there are no computational sections in the sweep algorithm prior to the main kernel.

The sweep ordering parameters, n_{sweeps}, n_{full} and n_{diag} represent the total number of sweeps per iteration, the number of full sweeps and the number of half sweeps respectively. The concept of 'full' and 'half' sweeps relates to the ability of sweeps within the application to overlap. Recall the sweep ordering presented in Figure 2. Sweep 2 originates on the processor located in the top right corner of the processor array. Once this sweep has successfully passed through the bottom right (the starting location for sweep 3) the next sweep can begin. If this starts prior to sweep 2 finishing on the bottom left processor, overlapping occurs which serves to increase the efficiency of the code. Overlapping can only occur if sweep i finishes at the starting location for sweep $i+1$ whilst other downstream processors are still processing sweep i. This occurs twice in Chimaera (sweep pairs 2,3 and 6,7) giving an n_{diag} value of 2. The full reusable model is presented in Table 2 with the complete equation for runtime given in (r5). Explanations of each sub-equation are given in [18]. Note that in the original paper describing the reusable wavefront model, the authors develop a complex LogGP communications model for the Cray XT4 architecture. In this work we develop a simpler regionalised least squares regression model to obtain times for MPI send and receive operations (these are presented in Section 5.1.1).

4.2 Simulation using the WARPP Toolkit

The WARwick Performance Prediction (WARPP) toolkit presented in this paper is a prototype performance prediction toolkit and evaluation engine, which has been designed to support performance prediction and code analysis on machines containing thousands of processors. More specifically, we intend for our toolkit to provide accurate simulations for modern Massive Parallel Processor (MPP) machines which might consist of multi-core, multi-processor cabinet structures each having their own complex interconnect or protocol. As the sizes of future machine architectures to continue to grow, we expect that additional sub-structures will be required to support increasing core counts, again each is likely to have its own performance properties adding further

complexity to modelling activities. With this in mind, the structure of a machine is relayed to the simulator by a series of 'profiles.' Each profile has unique performance properties such as network latency, outbound bandwidth etc. When developing a simulation the user is required to specify the respective values for each performance property and a mapping of MPI processes to profiles for the specific machine configuration being analysed. By providing a generic basis for the description of a machine, arbitrarily complex hardware models can be developed enabling the exploration not only of modern machine structures but also future multi-structured computing resources.

Simulations developed using the WARPP toolkit build upon the observation that parallel codes are ordered executions of basic blocks separated by control flow, calls to network transmissions or I/O operations. Like previous simulators we recreate application behaviour by replaying the code's control flow, pausing during execution to directly simulate computation, communication and I/O events. Communication between processes are simulated fully ensuring that transmissions between nodes block when the transmissive partner is otherwise engaged. Computation is, however, modelled quite differently to existing work in that it does not simulate each application instruction directly. Instead, the toolkit jumps over whole basic blocks within the control-flow recording the time that the block requires for execution on the target platform. The switch to coarser grained computational timings significantly reduces the time required for individual simulations aiding in improving the scalability of the simulator to considerably higher processor counts than previous toolkits. The issue which arises in moving to coarser grained computational timings is precisely how the time for the block is extracted from the application. To alleviate the manual instrumentation of code to obtain such timings the toolkit includes an automated code analyser which injects timing routines into the application source code directly creating an instrumented benchmark version of the code. The analyser also generates a control flow representation of the code detailing where each block can be found and how to identify it's associated execution time from the instrumented application output.

4.2.1 Developing a Simulation in WARPP

Developing a WARPP simulation involves three stages. In the first the application source code is analysed using automated code analysis tools - these are responsible for diagnosing the 'basic blocks' of the application and extracting a control flow graph for each process in the parallel application. Basic blocks are considered to be separated by either a change in the address counter (as would be caused by a branching statement or loop) or a communication (such as an `MPI_Send` or `MPI_Recv`). Once the basic blocks have been found, each is instrumented with timing routines to record the wall time that is required for execution. Two outputs are produced at this stage of simulation - an instrumented version of the application's source code and a basic performance model which describes the control flow of the application, the arrangement of basic blocks within this control flow and the points at which communication and I/O occurs.

The second stage of simulation requires the user to benchmark the machine using the instrumented version of the code and some reliable MPI benchmarking utility (such as MPPTest [22] or the Intel MPI Benchmark [9]). The output of these benchmarks, which takes the form of a 'work time' for each sequential block and a set of network latencies and bandwidths, is then fed into the third stage of simulation where the control flow is replayed using the wall clock times of each block to calculate the compute resources required and the communication points in the application directly simulated to obtain a communication model.

Network Profile	Message Size) (Bytes)	n_l (microsec)	B (GBytes/s)
on-chip (core to core)	≥ 0	0.655	2.70
off-processor (processor to processor)	< 2048	0.69	2.80
	≥ 2048	0.91	3.83
off-node (node to node)	< 2048	2.64	0.46
	≥ 2048	3.63	0.73

Table 3: Benchmarked Network Performance for the CSC-Francesca Machine (measurements taken using the Intel MPI benchmarking utility version 3.0 [9]. The Intel C compiler version 10 was used with default system MPI libraries.)

Core Count	Problem Size	Actual Runtime (sec)	Analytical Pred. (sec)	Sim. Pred. (sec)	Analytical Error (%)	Sim. Error (%)
32	120^3	107.18	88.76	89.58	-17.19	-16.42
64	120^3	56.72	47.59	48.75	-16.09	-14.04
128	120^3	32.56	28.20	28.98	-13.40	-11.01
81	240^3	342.33	326.45	330.46	-4.64	-3.47
96	240^3	297.03	268.71	277.56	-9.54	-6.55
100	240^3	278.37	243.36	248.32	-12.58	-10.79
128	240^3	225.65	205.50	207.18	-8.93	-8.18
169	240^3	174.35	174.35	177.09	-0.88	1.57
256	240^3	129.65	115.58	117.98	-10.85	-9.01

Table 4: Model Validations on the CSC-IBM Francesca Machine - (Compiler - Intel Fortran 10.0 with `-O2` optimisation setting, OpenMPI 1.2.5, All runtimes given are wall time for sweeping components in seconds, Negative values indicate under-predictions)

During a simulation, data relating to the application's performance and machine utilisation is recorded enabling performance modellers to replay the simulated execution at a later date and analyse where execution time was spent (for example, time spent in communication, computation, idle etc). As our studies into the applications used at AWE deepen we intend to use the simulation data to direct potential improvements in code structure and resource allocation.

5 Modelling Code Performance on a Commodity High Performance Cluster

In this section we present the results of a benchmarking and modelling exercise conducted on the recently installed Centre for Scientific Computing (CSC) *Francesca* machine operated by the University of Warwick. The benchmarked values from this machine serve two purposes - firstly to allow us to verify our performance models against a set of known runtimes ensuring accuracy, and secondly to form the basis of projections

for alternative machine configurations that may be considered during a procurement exercise.

5.1 The University of Warwick Centre for Scientific Computing (Francesca) Machine

The recently installed 11.5 TFLOP/s Centre for Scientific Computing (University of Warwick) IBM supercomputer is typical of a large, sub-Million pound commodity cluster available today. The system comprises of 240 dual-Intel Xeon 5160 dual-core nodes each sharing 8GB of memory (giving 1.92TB in total). Nodes are connected via a QLogic InfiniPath 4X, SDR (raw 10Gb/s, 8Gb/s data) QLE7140 host channel adapters (HCAs) connected to a single 288-port Voltaire ISR 9288 switch. Processor core to HCA ratio is $4:1$. Each compute node runs the SUSE Linux Enterprise Server 10 operating system and has access to the IBM GPFS parallel file system [20]. For our study the Intel C/Fortran 10 compiler suite was used in conjunction with OpenMPI 1.2.5 [6] and the PBS Pro scheduler. By default, jobs launched under PBS are allocated 'freely' in the system - *i.e.* to any free core which meets the wall time or memory resources requested by the job. Nodes and processors are shared between jobs unless specifically requested during submission. Runtimes can therefore vary (by as much as 10-15%) between successive runs due to the 'free' placement of processes within the machine and the potential sharing of node resources.

5.1.1 Machine Network Benchmarks and Models

The results of machine benchmarking demonstrating raw MPI latency and bandwidths are shown in Table 3. Note that the network benchmarking is partitioned into two regions by message size. The point at which the split in network performance occurs is 2048 bytes, indicating that the InfiniBand management system may be configured for a maximum transmission unit (MTU) size of 2Kbytes (a maximum of 4K is supported by the HCA and switch).

For both performance studies we model the communication time for a message of length x bytes as $t_{send}(x) = (1/B)x + n_l$ with the bandwidth (B) and latency (n_l) associated with the appropriate region for x . The time for a receive is modelled by: $t_{recv}(x) = (1/B)x$ since the receiver does not experience the latency required to establish the connection but must spend at least the actual transmission time in a locked state accepting data from the network interconnect. Using these values we can calculate the point at which bandwidth will dominate network transmissions as: $(2.62 \times 10^{-6})/(1(/0.46 \times 1024^3)) = 1304$ bytes (small messages) and $(3.63 \times 10^{-6})/(1/(0.73 \times 1024^3)) = 2846$ bytes for large messages. In the context of Chimaera these values, where each cell contains 10 angles, each of which is a double floating point value, equate to message sizes of 17 and 36 cells respectively. These values indicate the "see-saw" point at which the network operates, giving some indication of whether bandwidth or latency is dominant for each MPI operation.

5.2 Performance Model Validation

Table 4 presents validations of both performance models for the CSC-Francesca machine. For the results presented, the average prediction error is 10.46% for the analytical model and 9.03% for the simulation demonstrating the high degree of accuracy in the models and the strong correlation between both studies.

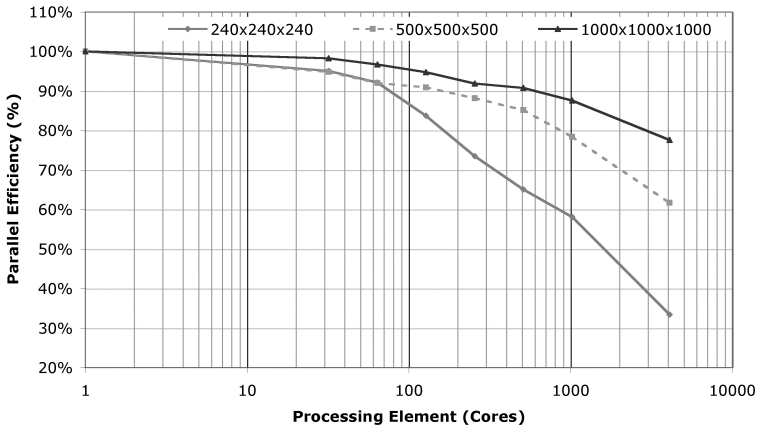


Figure 3: Parallel Efficiency of Large Problem Sizes using the InfiniBand Interconnect.

Note that the vast majority of the predicted runtimes are below the actual execution time - the principle reason being that both performance models assume as 'perfect' allocation of processor cores within the machine, assuming that neighbouring MPI ranks will be allocated as closely as physically possible. In practice, the free placement of processes causes some degree of increased execution time due to the higher network costs experienced. Similarly, the natural load and noise which occurs from shared resources helps to create variation in execution. Additionally, predictions are taken from averaged estimates of machine parameters for which rounding and measurements may also occur.

6 Procurement: Assessing the suitability of machine components

Following the benchmarking of the CSC-Francesca machine and validation of the performance models, we present several sub-studies exploring alternative machine or application configurations. In the following studies we analyse the effect on code runtime of a change in (1) an increase in problem size, (2) moving to a gigabit ethernet networking solution (3) the installation of InfiniBand resources with identical bandwidth but increased latency (4) a change in the performance of individual processor-cores and (5) a doubling of processor-core density.

6.1 Large Problem Sizes

New computing machinery is often purchased with the intention of not only running current codes but also future higher complexity problems or larger input sizes. The decision of which machine to purchase today may often be governed by expectations of how future users intend on using the system. Figure 3 presents the expected parallel efficiency of an increased input size with increasing processor count. Note that there is a significant decline in efficiency for each input size as the PE count rises. This effect is attributable to the increasing proportion of runtime accounted for by communication

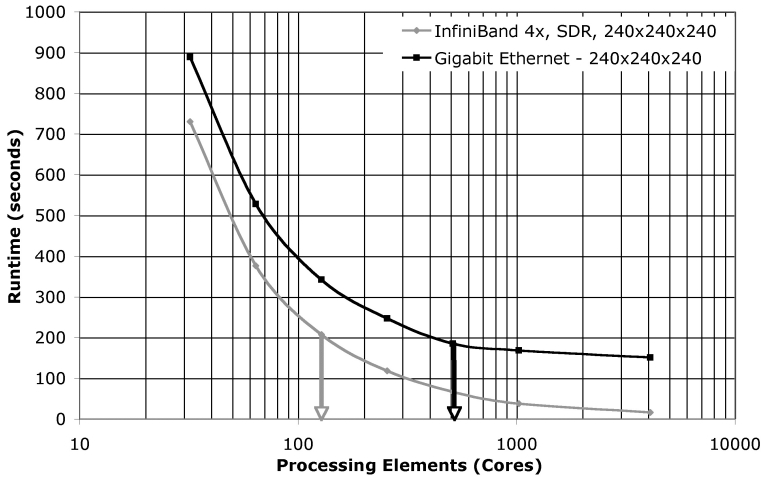


Figure 4: Chimaera Runtime using InfiniBand (4x, SDR) and Gigabit Ethernet Interconnects.

resulting from decreasing computation time per processor and an increase number of network transmissions in the system as a whole.

The measure of parallel efficiency is of particular interest to AWE since parallel jobs are mandated to be in higher than 50% configurations wherever possible with a number of users specifically choosing PE counts to target this value. For the 240^3 problem this point occurs between 1024 and 2048 cores indicating the approximate core count which may be required per job if targeted specifically for a 50% efficiency. Depending on how many simultaneous jobs the organisation wants to execute at this level of efficiency an approximate core count for procurement can be deduced. For larger problem sizes a similar form of analysis is also applicable, however, significantly more cores will be required before the 50% point is reached.

6.2 Choice of Networking Interconnect

For any machine intended to execute high performance parallel codes the choice of interconnect is particularly accute. The precise mix of latency, bandwidth capacity and cost must be balanced to support the compute resources in delivering smooth, consistent performance. At the time of procurement it is common to want to assess not only which interconnect will provide the best raw performance but also what the effect of changing the interconnect or choosing a slightly lower specification will have on overall runtime. We have modelled two such choices - (1) whether to select a Gigabit network over an InfiniBand interconnect and (2) the effect of purchasing an InfiniBand network with identical 4x, SDR bandwidths but 25%, 50% and 75% higher latencies.

Figure 4 presents the predicted runtimes for a hypothetical machine in which we have replaced the InfiniBand interconnect with a gigabit ethernet network. The gigabit runtime is consistently over 100 seconds slower than the InfiniBand system reflecting the impact of increased latency and a significant decrease in bandwidth. In analysing the results we propose that the reader considers the economics of purchasing either fewer processors and a more expensive InfiniBand network or a greater number of processors and a less expensive gigabit interconnect - a typical decision which may be

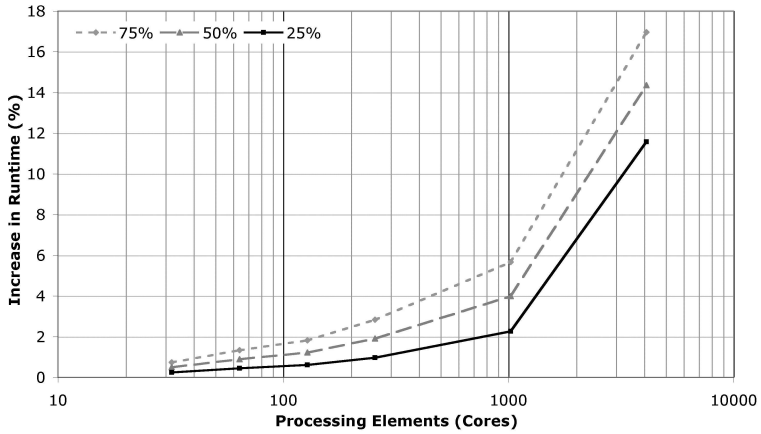


Figure 5: Increase in runtime from a 4x SDR InfiniBand network with varying increases in latency.

faced in any procurement activity. For the Chimaera benchmark at least, the results demonstrate that between two and four times as many processors will be required to offset the degradation of using a slower interconnect - a significant increase which will in turn make the machine more expensive to run and potentially more difficult to administer. The runtimes marked in Figure 4 present this point for the 128 and 512 core cases.

In Figure 5 we demonstrate predictions for the percentage increase in runtime resulting from the use of an 4x SDR InfiniBand interconnect with 25%, 50% and 75% higher latencies. For small processor counts (less than 1000) the increase in runtime is less than 6% in all cases. After this point - where communication begins to become a higher proportion of runtime - the runtime begins to increase rapidly with an increase of at least 10%. In this scenario the purchase of a lower specification system may be acceptable if the intention is to limit the maximum processor count of each job to 1024 cores or less.

We also believe that the use of machine configurations for node counts greater than 288 will cause increases in experienced wire latencies as tree based switch topologies will need to be employed in order to cope with the extra port count. These costs are not included in this work as benchmarked values to support a predictive model are not currently available and work completed in [10] provides some suggestion that contention within InfiniBand switches may be reduced in future systems through the use of advanced routing algorithms. Figure 5 does however help to give indication of how sensitive the structured communication pattern used in Chimaera is to even minor increases in network latency.

6.3 Machine Compute Performance

The compute resources of the machine are usually the feature which draws the most attention. Whilst only part of the picture for parallel systems, the computational aspects of a code are often better understood by domain experts and developers. With increasing variation in processors being offered in the form of increasing core counts and arrangements, considerable clock speed differences and in some cases, varying

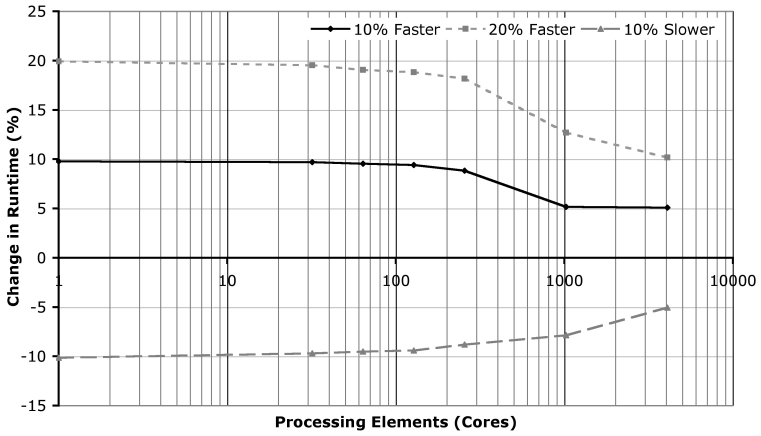


Figure 6: Increase in runtime from a varying changes in individual processor-core performance.

cache implementations, choosing the 'right' processor for an application can be difficult. We present several studies in this section of the paper which attempt to quantify the performance benefit of choosing either 10% or 20% faster processors, 10% slower processors or making the move from the existing dual-core Intel Xeon 5160 processors to quad-core chips with the same per-core performance but high core-density per processor.

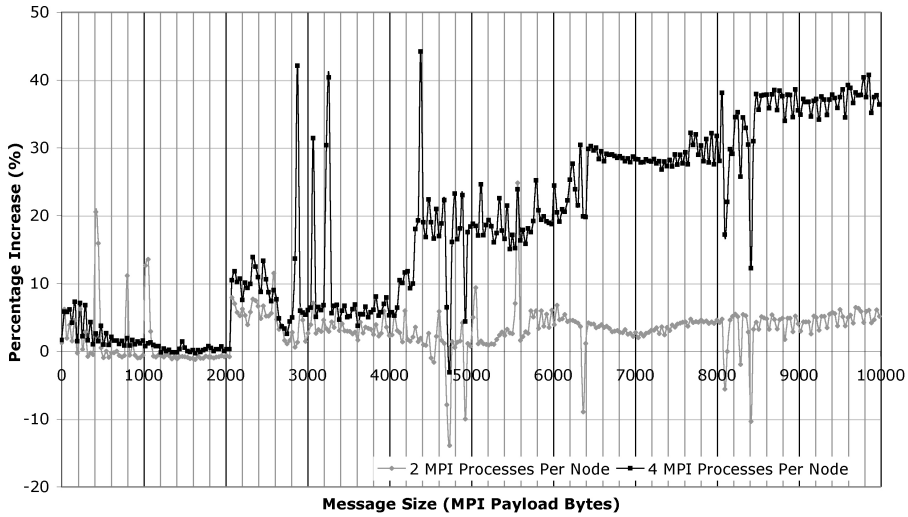
6.3.1 Increased Individual Core Performance

Figure 6 presents the predicted change in runtime from using dual core processors with individual core performances of +10%, +20% and -10%. The diminishing returns demonstrate the respective points at which communication begins to dominate runtime. In each case the change in runtime performance is approximately equal to the change in per core performance for small processor counts. As the processor count rises the impact on runtime is reduced due to the increase proportion of runtime accounted for by communication, reducing the contribution of faster computational resources to the runtime. Note that at increased processor counts the impact on runtime of using a slower processor is also reduced. The choice of core performance should therefore be considered in the context of job size - at small job sizes the runtime is improved best by using the fastest processors possible, as the core count in use rises there are diminishing returns from employing faster computational resources.

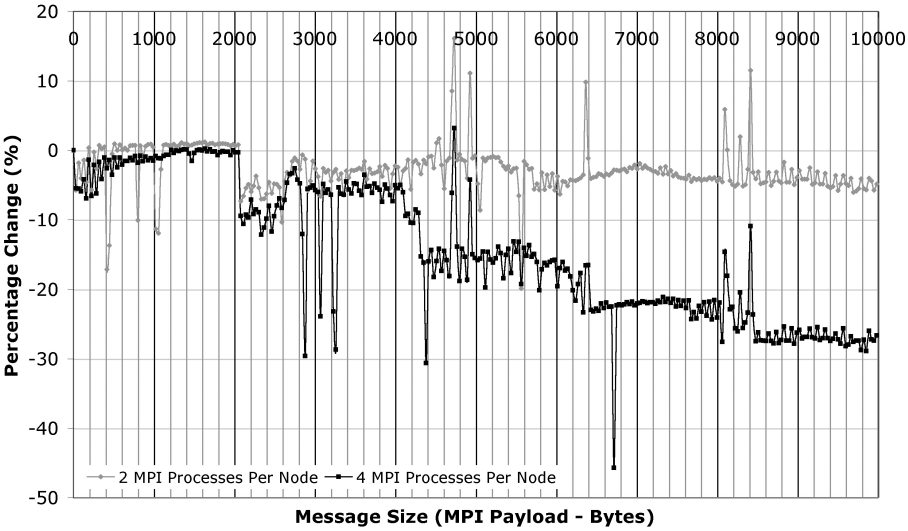
6.3.2 Increased Core Density - Dual versus Quad Core

With an increasing variety of multi-core processors becoming available including dual, quad and oct-core configurations, a common issue arising in procurement is which core density to select in designing the machine's compute architecture. On initial consideration the economic advantages of higher core densities are consolidation and reduced power or cooling demands per core, however, the increasing density often impacts on runtime performance.

In Figures 7(a) and 7(b) we show a set of results obtained from running the Intel



(a) Percentage change in time required to complete MPI send operation.



(b) Percentage change in per-core network bandwidth

Figure 7: Percentage change in time to send and per-core bandwidth when increasing the MPI process per node from one to two and four.

Total Core Count	Dual Core Runtime (s)	Quad Core Runtime (s)	Percentage Change (%)
32	729.97	726.19	-0.52
64	376.46	373.62	-0.75
128	207.18	207.92	0.36
256	117.98	118.50	0.44
1024	66.64	66.33	7.88
4096	37.29	40.45	8.46

Table 5: Predicted Quad versus Dual Core Performance (The quad-core configuration is modelled with an increased in time to send and reduced bandwidth to account for contention).

MPI benchmark in three configurations - one, two and four MPI processes per node respectively. The increasing number of processes per node (which is the effect of higher core densities) reduces the per-core network performance. The increased time to perform an MPI send, and the decreased per core bandwidth, result from high levels of contention for the single InfiniBand HCA per node. Each process must wait longer before having exclusive access to the machine network. If core densities continue to rise then this will continue to impact performance unless the issue of contention is addressed by increasing the number of networking channels per node - the economic effect of this may be a significant addition to procurement cost.

We have modelled the effect on runtimes of replacing each existing dual-core processor with a quad-core equivalent in which the per-core performance of the chip remains identical. The network latency for the InfiniBand network has been left the same for message sizes less than 2048 bytes, increased by 10% for message sizes ranging from 2048 to 4096 bytes and increased by 20% for larger messages. Network bandwidth has been changed by the same value but decreased. The changes in latency and bandwidth are drawn from the observed values shown in the figures above. Table 5 presents our predicted runtimes for the quad core machine compared with the existing dual core structure. Initially performance is improved since there are more cores utilising the fast core-to-core transmission speeds. Once core counts reach 1024 processors the increased latency and reduced bandwidth create up to an 8% increase in runtime.

7 Conclusions

In this paper we have presented a case study detailing the application of two performance models - one based on analytical techniques and the other based on simulation - in supporting the procurement of a large, sub-Million pound commodity cluster for a wavefront-code rich workload. The study explores the performance and scalability of the Chimaera benchmark code used by the United Kingdom Atomic Weapons Establishment.

We demonstrate average predictive accuracies of 90% for a variety of processor configurations and input sizes. The cross-correlation of predictions from two contrasting performance models serves to increase the confidence in our predictions and the insights obtained during our subsequent analysis.

More specifically, this paper shows:

- Quantitative estimates for the parallel efficiency of existing and future problem sizes that are of interest to AWE;
- That a system with a low performance network will require a greater processor count to offset the effect of higher latencies and lower bandwidth. We demonstrate this by projecting the performance of a Gigabit ethernet network in comparison to a faster InfiniBand system, showing approximately between two and four times as many processors are required by the ethernet system to achieve comparable levels of performance at core counts less than 1024;
- Improving/reducing the latency performance by a factor of 2, results in up to 10% change in overall runtime;
- For small processor counts the overall runtime varies by the factor of improvement in per-core performance, but as core counts increase, the contribution of faster per core performance provides diminishing returns;
- Increasing the core density per processor reduces the performance due to contention for memory and network resources. We estimate the quantitative degradation of overall runtime when doubling core-density from dual to quad core processors to be approximately 8% up to 4096 cores on the commodity InfiniBand system studied.

Our results demonstrate that the selection of machine configuration and processor count should be directed by the average size of jobs the machine is intended to execute. For multiple small jobs, individually faster processors should be prioritised over a faster interconnect, since the code is predominantly compute bound at these points. For larger jobs, the interconnect plays a more significantly role in performance indicating that a more expensive, low latency network should be targetted during procurement.

The predictive models used in this study demonstrate efficient, low-cost and rapid methods to gather quantitative and qualitative insights into questions which arise during procurement for both currently available and future systems. In contrast, traditional approaches such as direct benchmarking require significant and expensive machine execution time and more effort to arrive at a subset of conclusions limited solely to currently available machine configurations.

Acknowledgements

Access to the Chimaera benchmark was provided under grants CDK0660 (*The Production of Predictive Models for Future Computing Requirements*) and CDK0724 (*AWE Technical Outreach Programme*) from the United Kingdom Atomic Weapons Establishment. Access to the CSC-Francesca machine was provided by the Centre for Scientific Computing at the University of Warwick with support from the Science Research Investment Fund.

References

[1] V.S. Adve, R. Bagrodia, J.C. Browne, E. Deelman, A. Dubeb, E.N. Houstis, J.R. Rice, R. Sakellariou, D. Sundaram-Stukel, P.T. Teller, and M.K. Vernon. POEMS:

End-to-end performance design of large parallel adaptive computational systems. *Software Engineering*, 26(11):1027–1048, 2000.

[2] A. Alexandrov, M.F. Ionescu, K.E. Schauser, and C. Scheiman. LogGP: Incorporating long messages into the LogP model for parallel computation. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1997.

[3] Eric A. Brewer, Chrysanthos Dellarocas, Adrian Colbrook, and William E. Weihl. PROTEUS: A High-Performance Parallel-Architecture Simulator. In *Measurement and Modeling of Computer Systems*, pages 247–248, 1992.

[4] D.E. Culler, R.M. Karp, D.A. Patterson, A.Sahay, K.E. Schauser, E. Santos, R. Subramonian, and Thorsten von Eicken. LogP: Towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.

[5] Matthew Frank, Anant Agarwal, and Mary K. Vernon. LoPC: Modeling Contention in Parallel Algorithms. In *Principles Practice of Parallel Programming*, pages 276–287, 1997.

[6] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[7] G.R. Nudd, D.J. Kerbyson, E.Papaefstathiou, J.S. Harper, S.C. Perry and D.V. Wilcox. PACE: A Toolset for the Performance Prediction of Parallel and Distributed Systems. *The International Journal of High Performance Computing*, 4:228–251, 1999.

[8] A. Hoisie, G. Johnson, D.J. Kerbyson, M. Lang, and S. Pakin. A Performance Comparison through Benchmarking and Modeling of Three Leading Supercomputers: Blue Gene/L, Red Storm, and Purple. In *Proc. IEEE/ACM SuperComputing*, Tampa, FL, October 2006.

[9] Intel Corp. MPI Benchmark Utility. 2008.

[10] G. Johnson, D.J. Kerbyson, and M. Lang. Optimization of InfiniBand for Scientific Applications. In *International Parallel and Distributed Processing Symposium 2008 (IPDPS'08)*, Miami, Florida, USA, April 2008.

[11] Richard M. Karp, Abhijit Sahay, Eunice E. Santos, and Klaus E. Schauser. Optimal broadcast and summation in the logp model. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 142–153, 1993.

[12] D. J. Kerbyson, J. S. Harper, A. Craig, and G. R. Nudd. PACE: A Toolset to Investigate and Predict Performance in Parallel Systems. In *European Parallel Tools Meeting, ONERA, Paris*, October 1996.

[13] D.J. Kerbyson, A. Hoisie, and S.D. Pautz. Performance Modeling of Deterministic Transport Computations. In *Performance Analysis and Grid Computing, Kluwer*, October 2003.

[14] D.J. Kerbyson, A. Hoisie, and H.J. Wasserman. A Comparison between the Earth Simulator and AlphaServer Systems using Predictive Application Performance Models. *Computer Architecture News (ACM)*, December 2002.

[15] D.J. Kerbyson, A. Hoisie, and H.J. Wasserman. Use of Predictive Performance Modeling During Large-Scale System Installation. In *Proceedings of PACT-SPDSEC02*, Charlottesville, VA., August 2002.

[16] L. Lamport. The Parallel Execution of DO Loops. *Commun. ACM*, 17(2):83–93,

1974.

[17] G.R. Mudalige, S.A. Jarvis, D.P. Spooner, and G.R. Nudd. Predictive Performance Analysis of a Parallel Pipelined Synchronous Wavefront Application for Commodity Processor Cluster Systems. *Cluster 2006*, 2006.

[18] G.R. Mudalige, M.K. Vernon, and S.A. Jarvis. A Plug and Play Model for Wavefront Computation. In *International Parallel and Distributed Processing Symposium 2008 (IPDPS'08)*, Miami, Florida, USA, April 2008.

[19] S.K. Reinhardt, M.D. Hill, J.R. Larus, A.R. Lebeck, J.C. Lewis, and D.A. Wood. The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers. In *Measurement and Modeling of Computer Systems*, pages 48–60, 1993.

[20] Frank Schmuck and Roger Haskin. GPFS: A shared-disk file system for large computing clusters. In *Proc. of the First Conference on File and Storage Technologies (FAST)*, pages 231–244, January 2002.

[21] D. Sundaram-Stukel and M.K. Vernon. Predictive Analysis of a Wavefront Application Using LogGP. In *PPoPP '99: Proceedings of the seventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 141–150, 1999.

[22] W.Gropp and E.L. Lusk. Reproducible measurements of MPI performance characteristics. In *PVM/MPI*, pages 11–18, 1999.