

To Upgrade or not to Upgrade? Catamount vs. Cray Linux Environment

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

J. Holt

Tessella PLC, Abingdon Science Park, Berkshire, OX14 3YS, UK

I. Miller, J.A. Herdman, A. Vadgama

Atomic Weapons Establishment, Aldermaston, Reading, RG7 4PR, UK

Abstract

Modern supercomputers are growing in diversity and complexity – the arrival of technologies such as multi-core processors, general purpose-GPUs and specialised compute accelerators has increased the potential scientific delivery possible from such machines. This is not however without some cost, including significant increases in the sophistication and complexity of supporting operating systems and software libraries. This paper documents the development and application of methods to assess the potential performance of selecting one hardware, operating system (OS) and software stack combination against another. This is of particular interest to supercomputing centres, which routinely examine prospective software/architecture combinations and possible machine upgrades. A case study is presented that assesses the potential performance of a particle transport code on AWE's Cray XT3 8,000-core supercomputer running images of the Catamount and the Cray Linux Environment (CLE) operating systems. This work demonstrates that by running a number of small benchmarks on a test machine and network, and observing factors such as operating system noise, it is possible to speculate as to the performance impact of upgrading from one operating system to another on the system as a whole. This use of performance modelling represents an inexpensive method of examining the likely behaviour of a large supercomputer before and after an operating system upgrade; this method is also attractive if it is desirable to minimise system downtime while exploring software-system upgrades. The results show that benchmark tests run on less than 256 cores would suggest that the impact (overhead) of upgrading the operating system to CLE was less than 10%; model projections suggest that this is not the case at scale.

1 Introduction

Assessing the performance of individual hardware and software-stack configurations for supercomputers is a difficult and complex task, but with potentially high levels of reward. While the ultimate benefit of such system (re-)configuration and tuning studies is to improve application performance, potential improvements may also include more effective scheduling, kernel parameter refinement, pointers to application redesign and an assessment of system component upgrades. With the growing complexity of modern systems, the effort required to discover the elusive combination of hardware and software-stack settings that improve performance across a range of applications is, in itself, becoming an *HPC grand challenge*. The complexity of supercomputers is also set to grow, the introduction of general purpose GPUs (GPGPUs), dense many-core processors and advanced networking technologies only highlights the need to develop effective and scalable techniques that allow collections of components to be easily and cheaply compared.

Application Performance Modelling (APM) is such a technique. Different from traditional machine benchmarking and statistical analysis, APM has been shown to provide some of the deepest levels of insight into the performance of commodity clusters [6], supercomputers [9, 13] and their associated applications; these studies include efficient application configuration [15], machine selection [8] and more recently ahead-of-implementation algorithm optimisation [14]. This study concerns the use of these techniques in assessing the choice of operating system. Although Linux-based OSs are dominant in the supercomputing industry, proprietary lighter-weight kernels and more traditional environments such as Solaris, AIX and Windows HPC provide viable alternatives. Each operating system provides a unique blend of programming fea-

tures, reliability and performance. Selecting the appropriate operating system for a machine is therefore a complex choice, that must balance each of these requirements so that the machine provides an efficient runtime environment that is acceptable to system administrators and applications programmers alike. In this paper we use a recently developed simulation-based application performance model [7] of an AWE particle transport benchmark code to assess the impact of upgrading the operating system on AWE's 8,000-core Cray XT3 supercomputer. This paper provides a number of novel contributions:

- This is the first documented account of the use of a simulation-based application performance model to assess the impact of operating system selection or upgrades;
- This is the first independent comparison of the Cray Catamount and CLE operating systems for a 3,500+ node Cray XT3 machine;
- These are the first published results from a simulation study that demonstrate the scaling of the AWE particle transport benchmark code on a system of 32,768 cores - four times the current system configuration

The remainder of this paper is organised as follows: Section 2 describes related work; the main case study included in this paper, namely the modelling of AWE Chimaera on Catamount- and CLE-based machines, is described in Section 3; the paper concludes in Section 4.

2 Related Work

The subject of operating system selection is not new. From their inception in the early 1960s, through UNIX systems on the PDP-7 (and later PDP-11) in the early 1970s, to the MS-DOS and PC-DOS environments of more recent years, modern operating systems represent a collage of techniques which have been gradually refined through each successive hardware generation. Many of the early operating systems were tailored for particular machines (e.g. IBM's OS/360 for the System/360 mainframe), and as such there is a wealth of academic and commercial literature describing many of the design choices which are present in each system. Whilst there is no shortage of operating systems to select from, no one operating system has gained complete dominance. High performance computing has its own requirements: it may, for example, prove advantageous to select a light-weight, low-noise kernel for a *compute* node, as although it may be restricted in terms of its

user support, it will likely provide better performance than a full OS image. Tsafirir *et al.* present a good discussion of the potential dangers of operating system noise during computation [19]; this builds on work by Petrini *et al.* at the Los Alamos National Laboratory on diagnosing the poor performance of the SAGE hydrodynamics code when using a full allocation of processors per node [17]. The outcome of Tsafirir's work is the development of several methods for analytically assessing the likely impact of noise on application execution.

In [3] the authors use machine simulation to investigate the impact of two versions of UNIX on the performance of memory subsystems for a variety of benchmarks and full applications. The results obtained are compared with several commonly held views of memory performance, with several cases differing from the expected behaviour. [18] also includes an investigation of the impact of architectural trends on operating system performance for the IRIX 5.3 operating system used on large SGI supercomputers. These studies differ from our own in that they present a very low level analysis of performance when applications are executed serially or in very low processor count configurations. Conducting such an analysis *at scale* presents several challenges including long simulation times and considerable increases in the complexity of modelling contemporary machine components.

The authors of [2] use trace-driven simulation as a method for evaluating the impact of operating system noise (jitter) in the execution of several benchmarks. While the results of this work are promising, the methods developed required modification of the Windows NT kernel to permit necessary runtime instrumentation. This presents obvious difficulties in production environments. The work presented in this paper requires no kernel-level modification, instead a user-space benchmarking process is used to obtain a representative profile of the jitter present during execution which is then incorporated during simulation. The scale of our simulations are also considerably higher, reflecting the growth in machine sizes since [2] was published.

In more recent work, [1] benchmarks several petascale capable architectures including the Cray XT series and IBM's BlueGene/L platform. As well as pure operating system benchmarking, a series of experiments are presented in which injected noise is used to examine the likely effect of unsynchronized background operations on the performance of collective communications. The conclusion of the authors is that unless significant de-synchronisation of nodes occurs, operating system noise provided a small but limited degradation in runtime. The work in this paper builds upon

this benchmarking approach, integrating the output of a commonly used noise benchmark in simulations of a large parallel scientific code.

3 Case Study: Operating Systems and the Chimaera Benchmark

The Chimaera benchmark is a three-dimensional particle transport code developed and maintained by the United Kingdom Atomic Weapons Establishment (AWE). It employs the wavefront design pattern, a description of which can be found in [7, 15]. The purpose of the benchmark is the replication of the operational behaviour of larger internal codes which occupy a considerable proportion of parallel runtime on the supercomputing facilities at AWE. The code shares many similarities with the ubiquitous Sweep3D application [8, 12] developed by the Los Alamos National Laboratory (LANL) in the United States; it is however larger in code size and more complex in its control flow. Unlike Sweep3D, Chimaera employs alternative sweep orderings within the data array, a convergence criteria to halt simulation (standard configurations of Sweep3D always execute precisely 12 iterations) and extended mathematics. The performance analysis/tuning and modelling of both the Chimaera and Sweep3D benchmarks continues to be of interest to and AWE and LANL because of the proportion of execution time consumed by such processes (estimated to be in excess of 50% at LANL). Because of this, relatively small insights into the execution of the codes can provide substantial savings both in terms of cost and time simply because of the large numbers of CPU hours involved.

In this paper we extend a recently published simulation model [7] of Chimaera. The accuracy of this model has been shown to exceed 80% when representing execution on an SGI Altix 3700 and a 960-core Intel-Xeon InfiniBand cluster. The simulation is made possible using the WARPP toolkit [7], which allows the flexible modelling of supercomputing components in the form of separate code, compute timing and network models which can be interchanged to analyse alternative machine configurations. This research is extended in this paper to include the migration of the Chimaera application model to the Cray XT3 supercomputer operated by AWE, the addition of models of the Catamount and the Cray Linux Environment (CLE) operating systems and, further scaling studies and analysis.

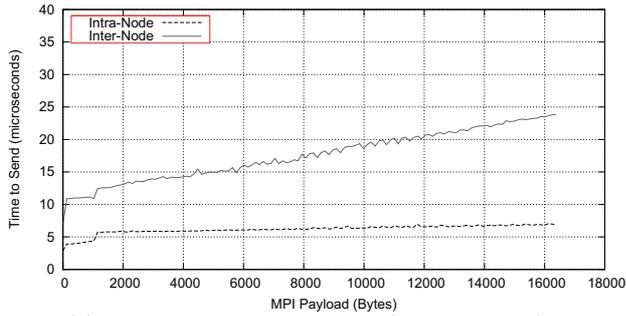
Note that our interest is in developing an insight into the behaviour of the Chimaera application when used at scale under CLE on the same hardware that is currently available. In order that the production en-

vironment is not disrupted, a small 128-processor pilot machine has been used to obtain performance profiles for the networking and compute behaviour of Chimaera under both CLE and Catamount. The hardware within the Pilot test machine is identical in specification to that of the production system. The extensions to the existing Chimaera model therefore require: (1) a new compute timing model for the XT3's AMD Opteron processor under Catamount and CLE, (2) network models in the context of both the Catamount and Cray Linux Environment (CLE) operating systems and (3) the benchmarking of noise profiles for each operating system, which can be integrated during simulation to analyse the effect of system jitter at scale.

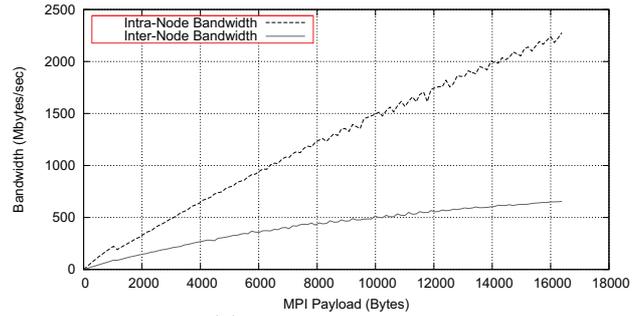
The case study is presented as follows: validations of the Chimaera simulation model on the existing Catamount-based XT3 system are shown, demonstrating the portability of the model to new architectures; we then use the benchmarking results of several small processor configuration runs of Chimaera on the pilot machine to predict the runtime behaviour of Chimaera at larger scale; these results are validated against benchmarked runtimes of Chimaera on the pilot machine; finally we analyse the likely implications for the performance of the full 8000-core XT3 machine if the operating system were to be upgraded to CLE.

3.1 AWE Cray XT3

The Cray XT3 [4] was Cray's first generation massively parallel processing (MPP) system to be built around commodity AMD microprocessors. The first prototype of such a system was the RedStorm machine developed in collaboration with the Sandia National Laboratory in the United States. The XT3 model has since been extended with higher bandwidth routers and increased core-density processors in subsequent revisions (XT4 and XT5). The compute engines utilised in the XT series are the dual-core (Opteron) or quad-core (Barcelona) AMD processors, connected to a high bandwidth three-dimensional torus interconnect via a specialised PowerPC 440 derived ASIC named the 'SeaStar'. The purpose of the SeaStar chips is to act as a separate network co-processor, allowing the main Opteron to offload much of the data processing which is required during network transmission. The XT3 system owned by AWE comprises of 3,944 compute-allocated dual-core AMD Opteron processors which run at 2.6Ghz. Each has exclusive access to 4GB of RAM.

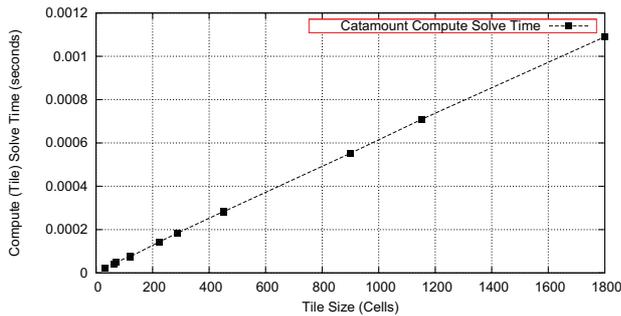


(a) Time to Perform MPI Send (Point-to-Point)

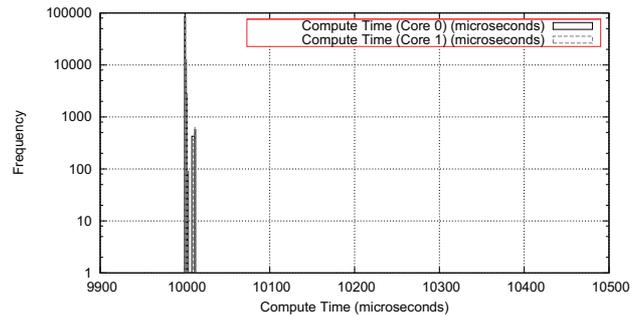


(b) MPI Bandwidth

Figure 1. Benchmarked MPI Performance of Catamount (Intel MPI Benchmarking Utility 3.1, Cray MPI)



(a) Per-Core Compute (Tile) Solve Time on Catamount



(b) Catamount P-SNAP[16] Operating System Noise Profile (10,000 μ s quantum)

Figure 2. Computation Benchmarks for Catamount

3.2 Catamount

Catamount has been designed to provide a lightweight, low-overhead operating system primarily for execution on compute nodes in large MPP clustering environments [10, 11]. The system has a long heritage with much of the conceptual framework for lightweight kernels having been developed in the SUNMOS, PUMA and Cougar operating systems featured on machines such as the Intel Paragon and ASCI Red. At the heart of the Catamount design lies the belief that general purpose operating systems, which are usually developed to support a wide range of requirements, are often loaded with so many components that their efficiency is reduced in compute-heavy environments. Catamount is therefore designed to reduce the number, and impact, of these background processes by stripping a Linux-like operating system to the minimal components required to run massively parallel computations. The design choice has therefore been one of efficiency over rich functionality. Specifically, Catamount has removed the following: (1) threading support, (2) off-node communications using UNIX pipes, sockets or RPC, (3) UNIX process forking, (4) dynamically loading of binaries, (5) Linux style ‘proc’ filesystem and (6) UNIX functions which require daemon support.

Figures 1(a) and 1(b) present the benchmarked MPI performance for the existing Catamount-based XT3. These graphs demonstrate discernable protocol differences at the 1024-byte payload where a separate protocol is used to process short and large messages. The compute benchmark values are presented in Figures 2(a) and 2(b). Operating system noise is recorded using the P-SNAP benchmark [16] which measures the variance in computation time of a predetermined calculation loop. Background processes and operating system interference delay this execution and present as increased periods of calculation time. Note the low degree of variation in the P-SNAP noise benchmark, which demonstrates the effectiveness of the stripped down, lightweight operating system.

Table 1 contains a set of predictions and actual run times for small processor configurations of Chimaera when executing on the Catamount XT3 pilot system compiled with the PGI compiler toolkit. Three predictions are provided - a lower, average and high estimate based on the respective values obtained during benchmarking. The model error is calculated for the average prediction. For each configuration the predictive accuracy exceeds 90%, demonstrating a high correlation between the simulation model and the code’s behaviour on the XT system. Note that these predictions are ob-

Problem Size	Processor Cores	Actual (secs)	Pred. (secs, low)	Pred. (secs, average)	Pred. (secs, high)	Error (Avr. Pred., %)
60×60×60	4	128.21	124.70	126.50	128.29	-1.34
60×60×60	8	68.72	65.76	66.72	67.68	-2.91
60×60×60	16	39.18	35.53	36.10	36.66	-7.87
60×60×60	32	23.84	21.64	22.01	22.37	-7.68
120×120×120	8	514.37	495.13	502.31	509.49	-2.35
120×120×120	16	271.93	254.10	257.86	261.62	-5.17
120×120×120	32	147.49	134.33	136.34	138.35	-7.56
120×120×120	64	78.97	71.80	72.93	74.06	-7.65
240×240×240	8	4026.59	3872.70	3929.11	3985.52	-2.42
240×240×240	16	2078.24	1954.55	1983.10	2011.64	-4.58
240×240×240	32	1085.21	999.18	1013.71	1028.24	-6.59
240×240×240	64	554.03	510.83	518.35	525.87	-6.44

Table 1. Validation of the Chimaera WARPP Simulation Model on Catamount

tained from the Pilot system to ensure fair comparison, model predictions for core counts up to 4,096-cores on production XT systems are presented in [7].

3.3 Cray Linux Environment (CLE)

The Cray Linux Environment (CLE) [20], also referred to in earlier literature as Compute Node Linux (CNL), is Cray’s successor to Catamount which attempts to selectively restore some of the features removed during the development of Catamount in order to make the compute nodes of large-scale machines closer in resemblance to conventional Linux based workstations. In so doing, the programming environment available on the compute nodes is richer and more familiar to application programmers. The objective of Cray in creating a mezzanine style OS, which sits between an entirely reduced platform and a full heavyweight operating system, is to provide a more familiar Linux environment on the compute side with performance which is no worse than 10% above that of the same system running Catamount [5]. The restoration of some background Linux services in the operating system, as well as providing a more familiar programming environment, also provides support for multiple-socket processors, OpenMP-based applications as well as reduced time for application porting.

The network benchmarks presented in Figures 3(a) and 3(b) demonstrate the significantly reduced core-to-core (intra-node) latency in CLE which is attributable to the re-engineering of internal MPI calls so that internal transfers occur via DMA rather than through direct SeaStar routing requests. For small codes which run entirely on-node the reduction in latency represents a potentially large reduction in runtime. Note that the CLE intra-node bandwidths are higher than Cata-

mount for smaller message sizes but approximately 500MB/s less for message sizes greater than 14,000 bytes. Benchmarked inter-node latency and bandwidth is poorer under CLE and demonstrates increased variance.

With reference to Chimaera, which typically sends a large number of small messages, the increase in inter-node latency (which is between 3 and 5 μ seconds higher) is likely to lead to higher runtimes, particularly on large, strongly-scaled problems where the communications become an increasingly dominant proportion of the total runtime. Since Chimaera executes over a two-dimensional processor array, which communicates using a near-neighbour communication pattern, the reduced intra-node latency will be of little benefit, as the processors execute in a highly synchronised pattern. For all cases other than a single processor (two-core) run, the synchronisation will therefore cause the pipeline of the application to run at the speed of the slower inter-node link.

The compute times for CLE are presented as a percentage increase over Catamount (see Figure 4(a)). Note that for each tile size, execution on CLE causes an increase in solve time of between 1.5 and 4.5%. Likely candidate causes for this increase are the presence of higher operating system noise as well as altered underlying operating system libraries and compilation strategies which may induce small delays.

Benchmarked P-SNAP noise for the CLE system is shown in Figure 4(b). The profile is interesting for two reasons: (a) the results demonstrate variability between the cores indicating that single core jobs might benefit from execution on the quieter core (core 1); (b) the second peak indicates a potential daemon which causes an impact of approximately 60 μ seconds.

The simulation-derived predictions for the perfor-

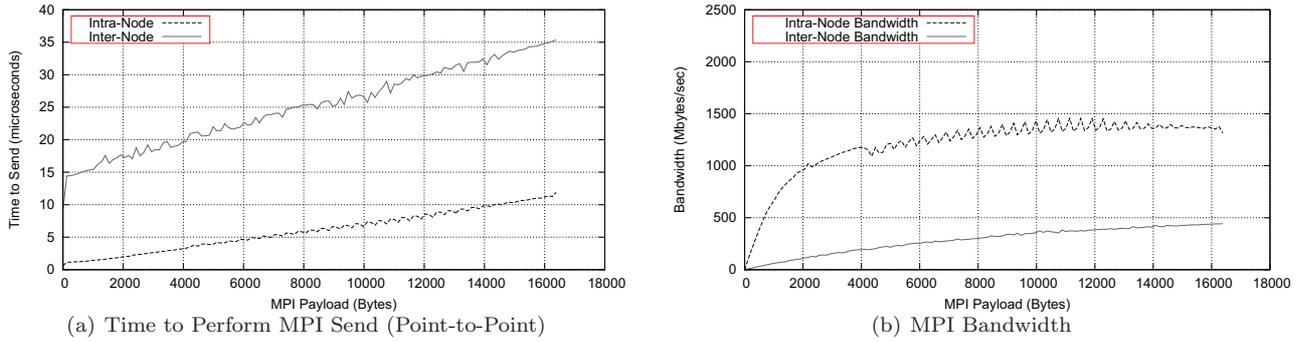


Figure 3. Benchmarked MPI Performance of CLE (Intel MPI Benchmarking Utility 3.1, Cray MPI)

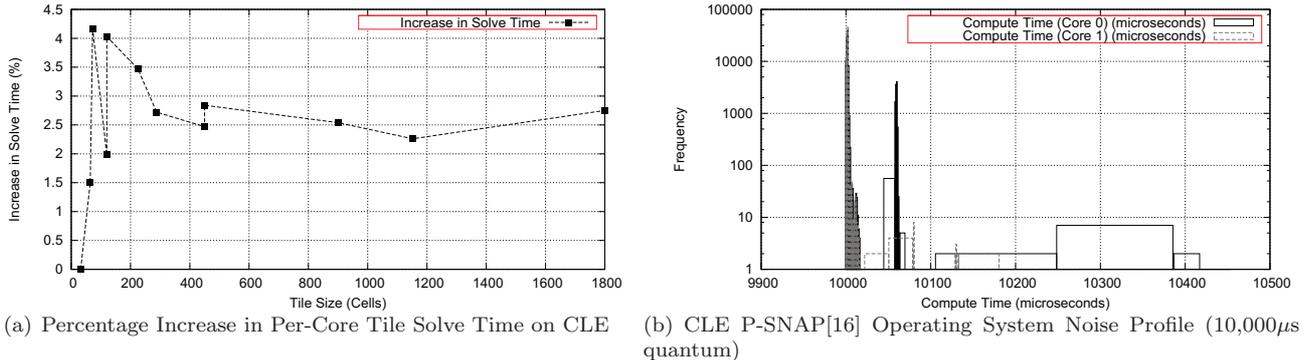


Figure 4. Computation Benchmarks for CLE

mance of Chimaera on CLE is shown in Table 2. Note that similar levels of accuracy are achieved by this model compared with those obtained from the Catamount system/model (greater than 90%) and that the runtimes follow a general pattern of being initially faster on CLE and then slower as the core count rises. We attribute the faster performance at very low core counts to the improved intra-node latency of CLE which becomes dominated at larger core counts by the more expensive inter-node latency. CLE runtimes are therefore poorer at larger scales because of the increased inter-node network latency.

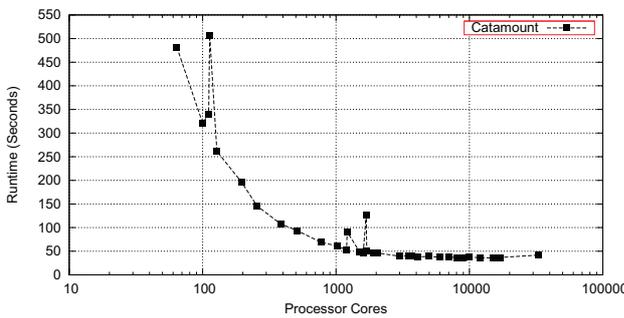
3.4 Assessing the Impact of CLE at Large Scale

Figures 5(a) and 5(b) show simulation-derived predictions for the runtime of Chimaera on Catamount and the percentage increase in runtime when running on CLE respectively. Note the spikes present in Figure 5(a) which arise from processor allocations that make the two-dimensional processor array less square and, hence, force downstream processors to wait for longer before receiving their initial data. The underlying simulation parameters for these projections are obtained through extrapolated values for the compute and networking costs by developing per cell compute

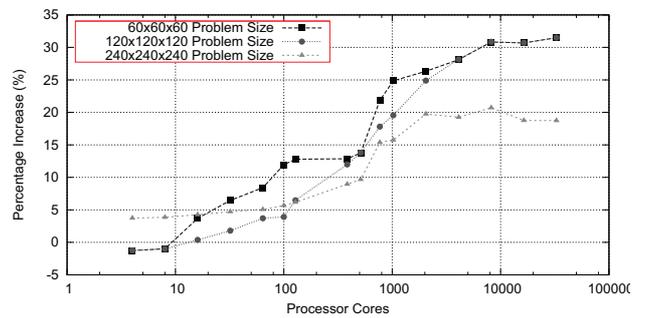
timings from the benchmarked values used in previous sections. Through using the known number of cells per processor at each configuration, appropriate compute and network times (based on simulated message sizes) can therefore be developed. The noise profile is replicated without extrapolation since it represents a problem-agnostic profile of jitter on the compute nodes. For runs of 256 processors or less, a common configuration for the benchmarking of Chimaera, the impact of the switch to CLE seems to suggest an increase in runtime within Cray’s target of 10% for the 240^3 problem and less than 15% for smaller cases. For configurations exceeding 1024 cores, the simulations show a runtime degradation of between 18 and 35%. Note that smaller problem sizes are sensitive to CLE at lower core-counts because the network is a more significant proportion of the parallel runtime. CLE’s 30% impact on runtime reflects the reduced compute size per core for these problem sizes, where the compute time is significantly smaller than the associated near neighbour message transmission. For CLE-based executions the disparity between the compute solve time and the message transmission increases at a faster rate due to CLE’s increased MPI latency.

Problem Size	Processor Cores	Actual (secs)	Pred. (secs, low)	Pred. (secs, average)	Pred. (secs, high)	Error (Avr. Pred., %)
60×60×60	4	129.47	119.00	126.02	133.04	-2.66
60×60×60	8	70.39	63.01	66.62	70.23	-5.36
60×60×60	16	40.52	35.88	37.76	39.64	-6.81
60×60×60	32	24.97	22.62	23.63	24.64	-5.37
120×120×120	8	515.37	490.17	497.18	504.20	-3.53
120×120×120	16	276.54	255.24	258.85	262.45	-6.40
120×120×120	32	150.00	136.92	138.79	140.67	-7.47
120×120×120	64	83.44	74.62	75.63	76.64	-9.36
240×240×240	8	4072.68	3831.16	3887.07	3942.98	-4.56
240×240×240	16	2083.92	1940.67	1968.64	1996.62	-5.53
240×240×240	32	1091.83	997.35	1011.48	1025.61	-7.36
240×240×240	64	558.22	513.13	520.34	527.55	-6.79

Table 2. Validation of the Chimaera WARPP Simulation Model on CLE



(a) Projected Parallel Runtime for a Catamount-based XT3



(b) Predicted % Increase in Runtime using CLE (240^3 Problem)

Figure 5. Predicted Chimaera Behaviour at Large Scale Processor Configurations

4 Conclusions

In this paper we present a case study investigating the likely impact on the performance of an industrial-strength benchmarking code when executing on (1) an existing light-weight, stripped-kernel OS (Catamount) and (2) a more feature rich OS upgrade (CLE). In each case we present benchmarked network, compute and operating-system noise. These data are used to prime a simulation-based application performance model, which is able to predict the runtime of the AWE particle transport benchmark code to within 90% of its actual execution time for a range of problem sizes and processor configurations. We believe that this paper provides a number of novel contributions: (1) This is the first documented account of the use of a simulation-based application performance model to assess the impact of operating system upgrades; (2) This is the first independent comparison of the Cray Catamount and CLE operating systems, which includes the simulation of operating system noise, for a 3500+ node Cray XT3 machine; (3) These are the first published results from a simulation study that demonstrate the scaling of the

AWE particle transport benchmark code on a system of 32,768 cores, four times the current system configuration, and (4) This is the first set of simulation results to demonstrate a predictive accuracy of greater than 90% on any AWE high-performance computing benchmark code.

Assessing the likely impact on the application runtime of an operating system upgrade from sample runs on a single small pilot machine is a difficult task. From the results shown in this paper, benchmarks on less than 256-cores may give the impression that the impact of switching to CLE would be less than 10-15% at scale; our projections show that that we do not believe this to be the case. The use of performance models in gaining insight into the impact of proposed operating system upgrades represents an inexpensive method of examining the likely resulting behaviour of the machine, and can be accurately conducted on small pilot machines leaving the production environment intact. The outputs from this study should therefore be seen in the context of providing a quantitative assessment of the change in runtime behaviour which results from an upgrade. When deciding whether to

proceed with any proposed kernel change, the impact on performance must be balanced against the desire for a richer programming environment or the improved reliability which may result from the new operating system – since these are often better understood, the techniques explored in this paper help to provide a clearer, more complete, picture into the likely benefits which any future upgrades may bring.

Acknowledgements

Access to Chimaera and the AWE XT3 and Pilot machines was supported by the United Kingdom Atomic Weapons Establishment (AWE) under grants CDK0660 (*The Production of Predictive Models for Future Computing Requirements*) and CDK0724 (*AWE Technical Outreach Programme*). Procurement and upgrade modelling programmes are also supported jointly by AWE and the TSB Knowledge Transfer Partnership grant number KTP006740.

References

- [1] P.H. Beckman, K. Iskra, K. Yoshii, S. Coghlan, and A. Nataraj. Benchmarking the Effects of Operating System Interference on Extreme-Scale Parallel Machines. *Cluster Computing*, 11(1):3–16, 2008.
- [2] J. Casmira, J. Fraser, D. Kaeli, and W. Meleis. Operating System Impact on Trace-Driven Simulation. In *Proc. of the 31st Annual Simulation Symposium*, pages 76–82, 1998.
- [3] J.B. Chen and B.N. Bershad. The Impact of Operating System Structure on Memory System Performance. In *SOSP*, pages 120–133, 1993.
- [4] Cray Inc. Cray XT3 Datasheet.
- [5] D. Wallace. Compute Node Linux: Overview, Progress to Date and Roadmap. In *Proceedings of the 2007 Annual Cray User Group*, 2007.
- [6] S.D. Hammond, G.R. Mudalige, J.A. Smith, J.A. Davis, A.B. Mills, S.A. Jarvis, J. Holt, I. Miller, J.A. Herdman, and A. Vadgama. Performance Prediction and Procurement in Practice: Assessing the Suitability of Commodity Cluster Components for Wavefront Codes. *IET Softw.*, pages 509–512, December 2009.
- [7] S.D. Hammond, G.R. Mudalige, J.A. Smith, S.A. Jarvis, J.A. Herdman, and A. Vadgama. WARPP: A Toolkit for Simulating High-Performance Parallel Scientific Codes. In *SIMUTools09*, March 2009.
- [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] 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.
- [10] S.M. Kelly and R. Brightwell. Software Architecture of the Light Weight Kernel Catamount. In *Cray User Group*, 2006.
- [11] S.M. Kelly, R. Brightwell, and J. Van Dyke. Catamount Software Architecture with Dual Core Extensions. In *Cray User Group*, 2006.
- [12] 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.
- [13] 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.
- [14] G.R. Mudalige, S.D. Hammond, J.A. Smith, and S.A. Jarvis. Predictive Analysis and Optimisation of Pipelined Wavefront Computations. In *11th Workshop on Advances in Parallel and Distributed Computational Models (APDCM)*. IEEE Computer Society, 2009.
- [15] G.R. Mudalige, M.K. Vernon, and S.A. Jarvis. A Plug-and-Play Model for Evaluating Wavefront Computations on Parallel Architectures. In *International Parallel and Distributed Processing Symposium 2008 (IPDPS'08)*, Miami, Florida, USA, 2008. IEEE Computer Society.
- [16] Los Alamos Performance and Architecture Laboratory. PAL System Noise Activity Program (LA-CC-06-025).
- [17] F. Petrini, D.J. Kerbyson, and S. Pakin. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In *SC '03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, page 55, Washington, DC, USA, 2003. IEEE Computer Society.
- [18] M. Rosenblum, E. Bugnion, S.A. Herrod, E. Witchel, and A. Gupta. The Impact of Architectural Trends on Operating System Performance. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, pages 285–298, 1995.
- [19] D. Tsafrir, Y. Etsion, D.G. Feitelson, and S. Kirkpatrick. System noise, OS clock ticks, and Fine-Grained Parallel Applications. In *ICS '05: Proceedings of the 19th Annual International Conference on Supercomputing*, pages 303–312, New York, NY, USA, 2005. ACM.
- [20] D. Wallace. Compute Node Linux: New Frontiers in Compute Node Operating Systems. In *Cray User Group*, 2007.