



Moving HPCx Applications to Future Systems

X.Guo, A.Gray, A.Simpson, A.Trew

EPCC, The University of Edinburgh, James Clerk Maxwell Building,
Mayfield Road, Edinburgh, EH9 3JZ, UK

Abstract

There is a diverse range of modern supercomputing architectures. In this report we discuss the porting of applications currently running (or similar to those currently running) on HPCx to IBM Blue Gene/L, NEC SX8, CRAY XD1 and STI Cell architectures. Described are experiences from such porting, and corresponding performance results. Where possible, direct comparisons to HPCx are presented. We show that the effort involved in porting, and the resulting application performance, are found to vary depending on the target architecture and the application.

This is a Technical Report from the HPCx Consortium.

Report available from
<http://www.hpcx.ac.uk/research/publications/HPCxTR0610.pdf>

© UoE HPCx Ltd 2006

Neither UoE HPCx Ltd nor its members separately accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.

1	<i>Introduction</i>	3
2	<i>Architecture</i>	3
2.1	HPCx (Phase 2a)	3
2.2	EPCC's Blue Gene/L Service – BlueSky	3
2.3	NEC SX8	4
2.4	Cray XD1	4
2.6	STI Cell	5
3	<i>Porting and Results</i>	5
3.1	AMBER 9 on Blue Gene/L	5
3.1.1	Introduction	5
3.1.2	Porting	6
3.1.3	Results and Discussions	6
3.1.4	Summary	7
3.2	Virgo's FLASH on the NEC SX8	7
3.2.1	Introduction	7
3.2.2	Porting	7
3.2.3	Results and Discussions	7
3.2.4	Summary	8
3.3	Benchmarking on the Cray XD1	9
3.3.1	Introduction	9
3.3.2	Porting	9
3.3.3	Results and Discussions	9
3.3.4	Summary	10
3.4	Benchmarking on the STI Cell	11
3.4.1	Introduction	11
3.4.2	Porting	11
3.4.3	Results and Discussions	11
3.4.4	Summary	12
4	<i>Conclusions</i>	12
5	<i>Acknowledgements</i>	13
6	<i>References</i>	13

1 Introduction

The supercomputing marketplace has traditionally offered a wide range of differing architectures. This range has, arguably, diversified further recently with the introduction of massively parallel machines such as the IBM Blue Gene/L and silicon chips which contain multiple processing cores, such as the STI Cell. There has also been a trend, pushed by economical factors, towards using commodity processors, as in seen in the Cray XD1. Furthermore, vector machines are still commonplace, one modern example being the NEC SX8.

It is of interest to evaluate the effort that will be required to port codes currently running on HPCx to such systems, and to predict how well these applications will perform. In doing so it should give an indication of the suitability of each system for the scientific computing currently being performed on HPCx.

For each of the above architectures, applications currently running (or similar to those currently running) on HPCx have been ported and performance has been investigated. Most of this work was not carried out by the authors, but has appeared in recent publications (see the body of the report for more details). This report summarises, and in certain places adds to the results of, these publications.

The architecture of each system is described in section 2. Porting experiences and performance results, along with a discussion of the potential issues involved with porting the range of applications currently running on HPCx, are given in section 3 for each architecture in turn. Where possible, direct performance comparisons are made to HPCx.

2 Architecture

2.1 HPCx (Phase 2a)

HPCx [1] is the current UK national high performance computing service, which is a typical shared memory cluster. The current HPCx (Phase 2a) system has 96 IBM eServer 575 compute nodes, each of which has 16 IBM POWER5 processors with a clock speed of 1.5GHz. Each chip is packaged into a Dual-Core Module (DCM) and one eServer node contains 8 DCMs.

Each processor has its own Level 1 cache, which is divided into a 32KB data cache with 128-byte lines and a 64KB instruction cache. The two processors on one chip share a 1.9MB Level 2 cache and they also share a 36MB Level 3 cache. Each node on HPCx has a main memory of 32Gbytes.

The inter node communication on HPCx is provided by an IBM's *High Performance Switch* (HPS) and intra node communication is via shared memory.

2.2 EPCC's Blue Gene/L Service – BlueSky

The IBM Blue Gene/L [2] machine is a re-incarnated classical distributed memory machine, which can provide high performance and good scaling with both low power and low cost.

EPCC's BlueSky [3] is a Blue Gene/L machine with 1024 compute chips (2048 processors) in a single rack. Such a high processor density is enabled by the fact that the two PowerPC 440 processors on each chip have a relatively low clock speed, 700MHz, compared to many current supercomputers.

On each chip, there is a main memory of 512MB together with three levels of cache. Each processor has a 32KB Level 1 data cache and 32 KB Level 1 instruction cache. The L1 caches are not coherent between the two processors on a chip. The Level 2 cache is only 2KB and is used for prefetch. There is also one Level 3 cache of 4 MB, which is made from embedded DRAM, shared between the two processors on a chip.

The Blue Gene/L machine has two operation modes: *Co-processor mode* (CO), in which one processor of each chip takes charge of the computations and the other processor is utilised for communications, and *Virtual-Node mode* (VN), in which each processor handles both computations and communications. In VN mode, the resources on each chip are shared between the two processing cores.

There are five networks on BlueSky with different functionalities. The *3D TORUS network* is utilised for the point to point communication and there are two specific tree networks: *Global Tree network* for the collective communication with global operations such as broadcast and allreduce, as well as *Global Barrier and Interrupt Tree network* for fast barrier synchronisations.

2.3 NEC SX8

The NEC SX8 [4, 5], housed at the High Performance Computing Centre Stuttgart (HLRS) of the University of Stuttgart in Germany, is one of the few vector supercomputers which are available for academic use in Europe. The NEC SX8 is actually a SMP cluster consisting of 72 nodes, each of which has 8 SX8 vector processors with a clock speed of 2GHz. The 8 vector processors on each node share a main memory of 128GB, which is up to 9.2 TB in total for the whole machine. There is also a 160TB shared disk space and 140 GB disk space per node. The nodes on the NEC SX8 are connected with a central crossbar switch, the IXS interconnect.

The NEC SX8 also has a frontend, a scalar NEC TX7 called "Asama" [6], for the cross-compilation as well as pre- or post- processing data for the SX nodes. The frontend consists of 16 Itanium2 processors shared with 256 GB memory. The frontend and the backend share a file system.

2.4 Cray XD1

The Cray XD1 [7] is a purpose built system for High Performance Computing (HPC) owned by the CCLRC laboratory at Computational Science and Engineering Department. It consists of 6 chassis, each of which contains 6 nodes, i.e. 6 dual AMD Opteron 250 processors with a clock speed of 2.4 GHz. The system is configured as 1 master node and 35 compute nodes.

The Level 1 cache on the Cray XD1 includes 64KB instruction cache and 64KB data cache. There is also a 1MB Level 2 cache with 16-way associativity and Error Correcting Code. Its main memory is up to 96GB per chassis [8, 9].

The Cray XD1 system is based on the Direct Connected Processor (DCP) architecture [10]. The Cray XD1 uses the RapidArray™ interconnect to connect all the processors directly, which can eliminate PCI bus bottlenecks. The RapidArray™ interconnect uses 12 custom communications processors and 96 GB/s non blocking switching fabric per chassis to deliver 8GB/s bandwidth between SMPs with 1.8 microsecond MPI latency. Each chassis presents 24 external RapidArray™ links with a total 48GB/s bandwidth between chassis.

2.6 STI Cell

The STI Cell processor [11,12], which is developed collaboratively by Sony, Toshiba and IBM (STI), is a new approach to provide high performance as well as high power efficiency. It employs a multi-core design: the Cell processor consists of one *POWER™ processing element* (PPE), which is a 64bit PowerPC, and 8 *synergistic processing elements* (SPEs), which are simplified SIMD processors, and act as the computation workhorses for the main program on PPE.

The memory hierarchy of Cell architecture must be managed explicitly in software. The SPEs can not access the main memory directly, but instead each SPE can use its local direct memory access (DMA) engine to populate a 256kB fast local store (LS) with main memory data (or data from another SPE's LS) which must travel via an element interconnect bus (EIB), which has a limited bandwidth and is shared between all SPEs.

There are 4 *single precision* (SP) units and only one *double precision* (DP) unit in each SPE. The length of pipeline is optimised for the SP unit. Communication mechanisms on the Cell architecture, in addition to DMA, include signals and mailboxes, which enable quick transfer of small amount of data between processing elements without the overhead of DMA transfer.

Provided is a Cell software development kit (SDK) which contains the library *libspe*, to provide functions for the hand-managing SPE threads from PPE. It also provides an *Interface Description Language* (IDL) compiler, for remote procedure calls without any manual handling of the communications between PPE and SPEs.

3 Porting and Results

This section describes, for each new architecture in turn, the issues experienced and results gained from porting benchmark codes.

3.1 AMBER 9 on Blue Gene/L

3.1.1 Introduction

The porting and performance of several typical HPCx applications on the Blue Gene/L architecture was recently discussed in an HPCx technical report [13], where it was found that in general the performance was better than expected compared to HPCx (Phase 2), although the architecture is only suited to those applications which scale well and do not require large amounts of memory. This section, which summarises the work presented in [14], extends these investigations with a comparison of the AMBER molecular dynamics (MD) code on Blue Gene/L to HPCx (Phase 2a).

AMBER (Assisted Model Building with Energy Refinement) is a biochemistry simulation system which is actually a MD suite that uses analytic potential energy functions, derived from experimental and ab initio data, to refine macromolecular conformations [15]. The AMBER 9 Joint AMBER-CHARMM benchmark (known as JAC) with a total of 23558 atoms and the Factor IX Protein benchmark (known as Big) with 90906 atoms in the system were each tested using two code implementations: Sander and PMEMD, the optimised re-implementation of Sander [16].

3.1.2 Porting

Blue Gene/L is a supported architecture for AMBER code, so porting AMBER to BlueSky was trivial.

3.1.3 Results and Discussions

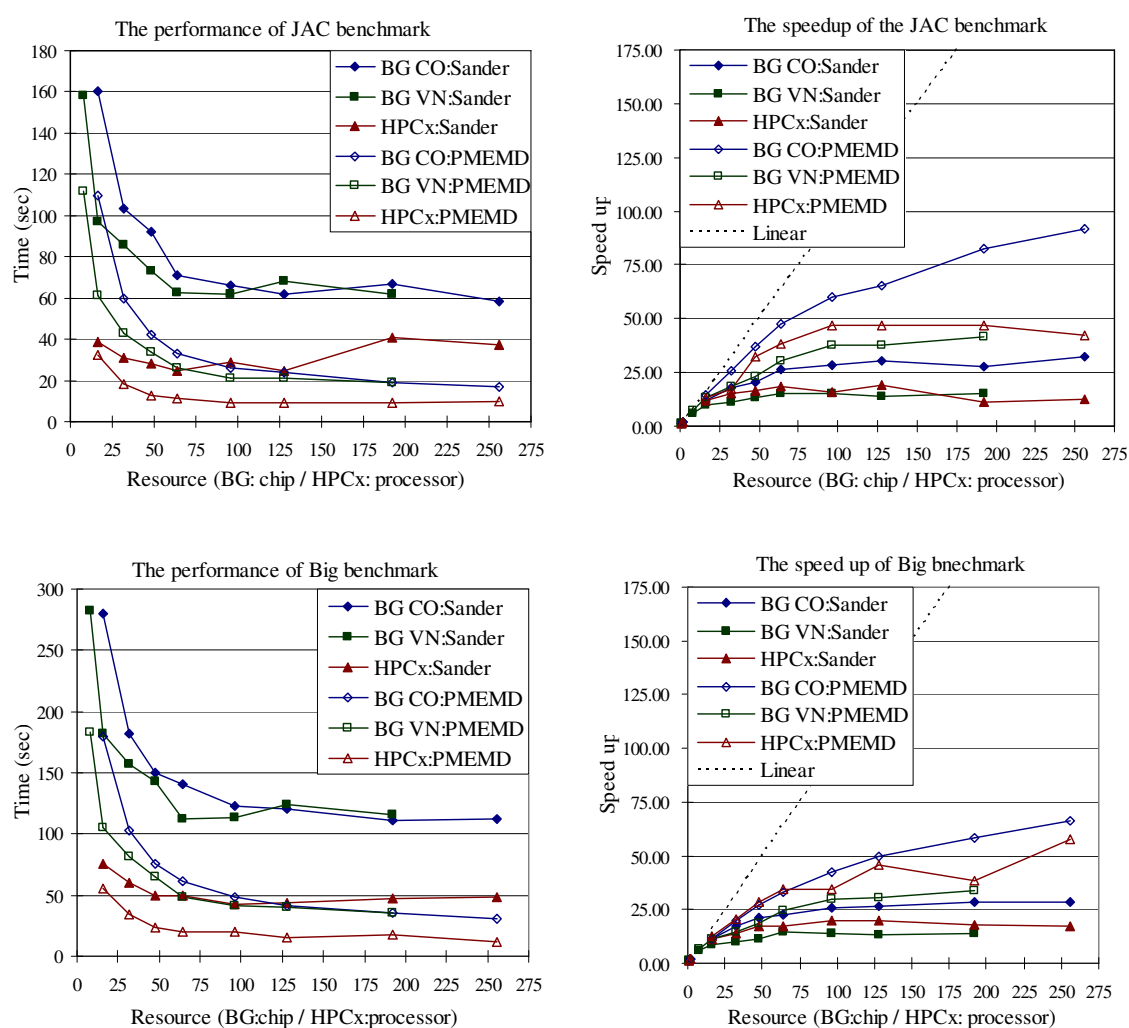


Figure 1 Reproduced from [14]. The left hand graphs show the execution time of the AMBER 9 Joint AMBER-CHARMM (JAC) (top) and Factor IX Protein (Big) (bottom) benchmarks on BlueSky and HPCx using either Sander or PMEMD code implementations. The right hand figures show the corresponding speed up. In VN mode on BlueSky, there were two processes per chip. The number of chips on BlueSky is directly compared with the number of processors on HPCx.

Before comparing the benchmark performance on HPCx and BlueSky, there are two factors to be noted. Firstly, the clock speed of HPCx is approximately 2.1 times as much as that of

BlueSky. Considering the two independent floating point units of each processor on HPCx, the performance on HPCx was expected to be around 4.2 times faster than that of BlueSky. Another point is that the processors of BlueSky are much less expensive, in term of monetary cost, power and floorspace than those on HPCx.

Figure 1 shows that the benchmark performance is in all cases better on HPCx. However, the factor in all cases is less than the expected 4.2, especially at large chip/processor counts. The benchmarks are seen to scale poorly on both systems, likely due to their small size. However, scaling is seen to be better on BlueSky than HPCx, particularly for the optimised version of the code, indicating that the BlueSky network has higher performance.

3.1.4 Summary

Porting AMBER 9 to Blue Gene/L was a straightforward procedure. This together with the results of [13] indicates that the porting of similar HPCx applications should be relatively straightforward, provided that they fit into the available memory. Good performance should be possible for codes which can scale to large number of processors, but the architecture of Blue Gene/L will not be suitable for those poorly scaling codes which rely on high clock speed.

3.2 *Virgo's FLASH on the NEC SX8*

3.2.1 Introduction

The Virgo Consortium [17] is a collaboration of scientists from UK, Germany, Canada, the USA and Japan. One of its main purposes is to perform Cosmological Supercomputer Simulations to investigate the early structure formation of the universe. FLASH [18] is a modular, Adaptive Mesh Refinement (AMR), parallel code used for cosmological simulations.

This section summarises the work presented in [19]. FLASH was ported to the NEC SX8 and the benchmark results, using a 128K (128^3 particles) Santa Barbara (StB) data set, were gained and compared with those from HPCx. Code optimisations were performed on the NEC SX8 system, and further benchmarking was done.

3.2.2 Porting

Porting this application to a vector supercomputer for the first time resulted in several problems, most of which due to the use of non-standard FORTRAN. A significant amount of time was spent in addressing these issues, but a successful port was achieved. For some routines, compiler optimisation had to be disabled or set to safe vectorisation to enable compilation.

3.2.3 Results and Discussions

FLASH was benchmarked on HPCx and the NEC SX8. Table 1 shows that the run times on the two machines are similar. Vector machines typically have fewer processors than their scalar counterparts, and rely on much higher single processor performance. Therefore, these results indicate that FLASH does not effectively take advantage of the vector architecture (which is not surprising since the code was ported, without modifications, from a scalar architecture). HPCx is seen to scale better, as is accentuated in Figure 2.

CPUs	Run time [min]	
	HPCx	NEC SX8
16	135	125
32	86	79
64	51	52
128	33	43

Table 1 Reproduced from [19]. The run times of Virgo’s FLASH on HPCx and the NEC SX8 for time step n=48 with 128K StB.

CPUs	Run time [min]	
	HPCx	NEC SX8
16	22	42
32	14	21
64	15	15
128	53	15

Table 2 Reproduced from [19]. The run times of the optimised Virgo’s FLASH using FFT static on HPCx and the NEC SX8 with 128K StB.

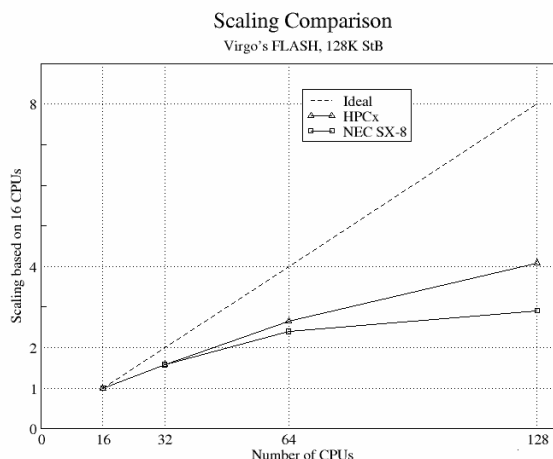


Figure 2 The scaling of Virgo’s FLASH on HPCx and the NEC SX8 with 128K StB [19].

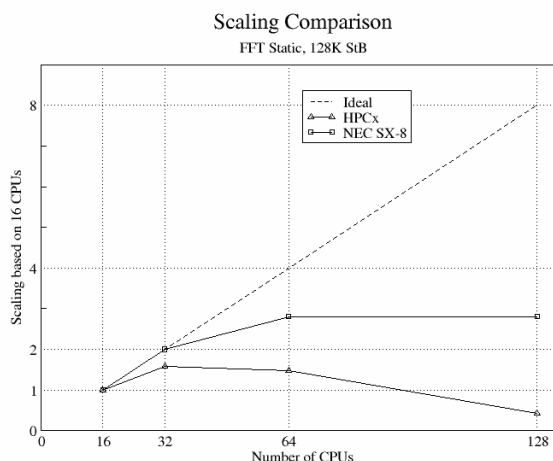


Figure 3 The scaling of optimised Virgo’s FLASH using new FFT static on HPCx and the NEC SX8 with 128K StB [19].

Optimisations were adopted based on profiling results on the NEC SX8, including the integration of a new algorithm which employs the Fast Fourier Transforms instead of the iterative method to solve Poisson’s equation. This resulted in a factor of 6 to 7 speed up.

Table 2 and Figure 3 show the performance and scaling respectively of FLASH with the new algorithm on the NEC SX8 and HPCx. The scaling is seen to be better on SX8, but poor on both systems, probably due to the small problem size. A larger problem would be needed to fairly compare the scalability of the systems. However, HPCx is seen to perform better than the SX8 at all processor counts.

3.2.4 Summary

FLASH was successfully ported to the vector NEC SX8 machine, albeit with problems due to the use of non standard FORTRAN code, demonstrating that the porting of such applications is possible. This code was seen, however, to not take advantage of the vector processors and is thus more suited to scalar architectures. It is expected that there may be a subset of HPCx

applications which are easily vectorisable and can potentially perform well on this system, but code modifications may be required at least in some cases. It is likely that applications exist which contain algorithms inherently more suited to scalar processors.

3.3 Benchmarking on the Cray XD1

3.3.1 Introduction

Two applications: DL_POLY [20] and PCHAN [21] were ported to and tested on the Cray XD1 machine. This summarises, and expands on, the work presented in [22].

DL_POLY is a general purpose molecular dynamics simulation package developed at CCLRC's Daresbury Laboratory by W. Smith and T. R. Forester. There are two versions of this application: DL_POLY 2.16 and DL_POLY 3.06. For this investigation, the Potassium disilicate glass (with 3-body forces) test case with 1080 atoms of DL_POLY 2.16 was tested.

PCHAN, which was developed by the UK turbulence consortium, uses direct numerical simulation techniques to solve the Navier-Stokes' equations for turbulent fluid flow. A small PCHAN benchmark, T1, which has a cubic grid size of 120x120x120 and a memory requirement of approximately 1.3GB was used.

3.3.2 Porting

3.3.2.1 Porting DL_POLY

Porting DL_POLY 2.16 to the Cray XD1 was straightforward. The architecture of Cray XD1 is supported by the DL_POLY code, and a Makefile with standard Cray XD1 optimisations is included with the distribution.

3.3.2.2 Porting PCHAN

Porting PCHAN to the Cray XD1 was fairly challenging. This system is not among those supported by PCHAN. The source code had to be modified, and overall a significant amount of time was spent in porting.

3.3.3 Results and Discussions

3.3.3.1 DL_POLY

Figure 4 shows the performance and scaling of DL_POLY on the Cray XD1 and HPCx. It can be seen that the Cray XD1 performs better, i.e. shows lower execution time, than HPCx, but this performance advantage of the Cray XD1 is not as high as expected if considering the XD1/HPCx processor clock frequency ratio of 1.6. This could indicate that the HPCx processors have a better performing memory architecture. The code is seen to scale fairly well on both HPCx and the Cray XD1, with the XD1 having a slight advantage.

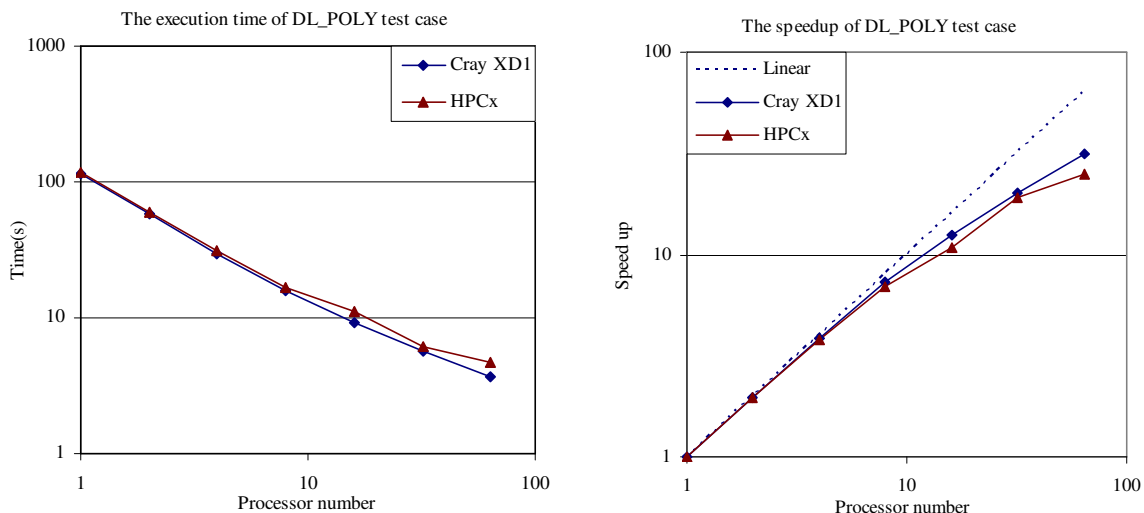


Figure 4 Reproduced from and expanded on [22]. The left figure shows the execution time of DL_POLY on the Cray XD1 and HPCx. The test case is the one to calculate Potassium disilicate glass (with 3-body forces) with 1080 atoms. The right figure shows the corresponding speedup.

3.3.3.2 PCHAN

The PCHAN T1 benchmark results are shown in Figure 5. T1 is seen to scale very well on HPCx, and even superscales at high processor counts. The scaling on the XD1 is only slightly poorer. HPCx has the performance advantage at all processor counts.

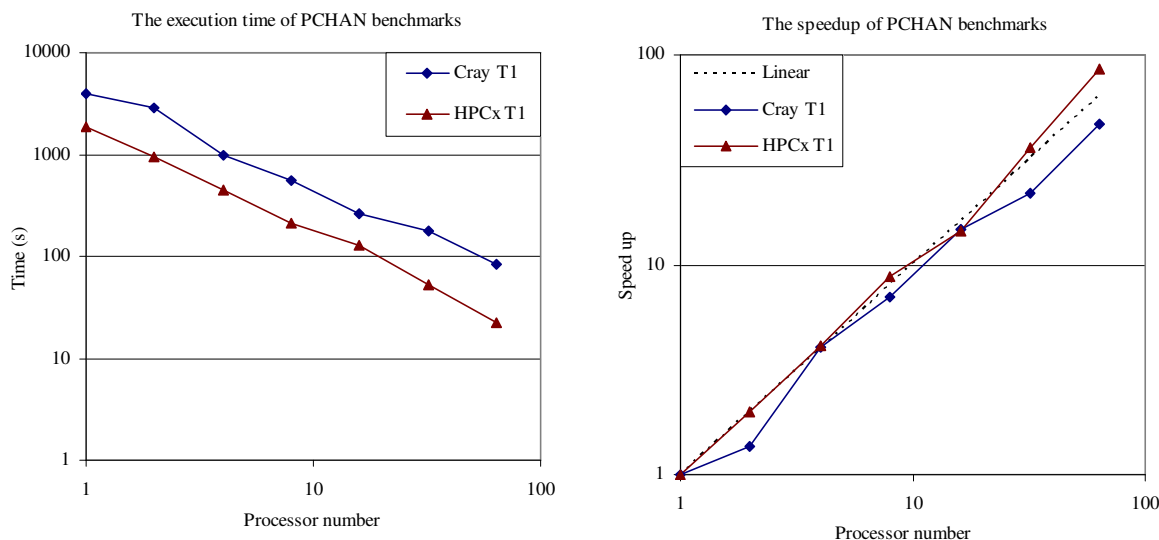


Figure 5 Reproduced from and expanded on [22]. The left figure shows the execution time of PCHAN T1 benchmarks on the Cray XD1 and HPCx. The right figure shows the corresponding speedup.

3.3.4 Summary

DL_POLY and PCHAN were successfully ported to the Cray XD1 architecture. The effort required was dependent on the application code. In general the XD1 was seen to perform comparably to HPCx, but perhaps not quite as well as one would expect from the clock frequency ratio.

3.4 Benchmarking on the STI Cell

3.4.1 Introduction

This section summarises the work presented in [12]. These results should be regarded as experimental: the physical Cell hardware was unavailable, so the tests were run on the Cell Simulator which is a part of the Cell Software Development Kit (SDK). Note also that all the tests were executed using only one processor and at this stage it is unclear how effectively codes could scale to multiple processors.

Two codes were benchmarked: a Monte Carlo simulation of a Heisenberg model ferromagnet and a MPI image processing program. Results were compared to those from an Intel Xeon processor. The simulated model of the first program is actually a simple 2D square grid of “spins”, and the system was implemented using the Metropolis algorithm. On the Cell, the lattice is decomposed to SPEs 1-dimensionally and the edge spins (halos) are swapped between the neighbouring SPEs during the iterations of the Monte Carlo procedure. The `libspe` version and IDL version of this program were both tested using both single precision (SP) and double precision (DP).

The image processing program, originally written using MPI, implements a simple algorithm for reverse edge-detection to reconstruct an image (where the input here is the output of an edge detection algorithm applied to an image). This operation may seem slightly artificial, but is actually similar to a large number of real scientific HPC applications that solve partial differential equations. For execution on the Cell, code using both `libspe` and IDL was derived from the original MPI code. In this test, the program operated on a small image (360x192 pixels) which was divided into 48 segments.

3.4.2 Porting

The Cell architecture does not support the commonly used message passing and shared memory parallelisation paradigms, so a code reimplementaion had to be performed in order to distribute tasks between the SPEs: each of the two programs used both the `libspe` and IDL approaches. A significant amount of time was spent in the porting, and the resulting code was verbose and Cell specific. Using the IDL approach is more succinct than using `libspe` approach but gives less efficient use of the Cell.

3.4.3 Results and Discussions

Since the Cell hardware was not available and all the results are achieved using the simulator, the accurateness of performance may be limited. Note that the simulator performance results were produced in cycle counts and the results are reported in units of time based on a 3.2GHz clock speed. Although the clock speed of Xeon is 2.4GHz, for a more fair comparison, the clock speed of Xeon has been scaled to 3.2GHz.

Table 3 compares the estimated calculation time of the Monte Carlo (MC) code on the Cell processor to the (scaled up) Xeon results. The Cell intra-processor communication time estimates are missing but are expected to be small for this code according to tests of the performance of the Cell communication network [23] as well as the analysis in [12]. The Cell is seen to achieve much higher performance than the conventional Xeon. However, it should be noted that this system size fits into the SPE LS space: if the system is too large, it will be more complicated to perform double buffering and this may severely detriment performance.

	Cell	Xeon
SP	41.0	394
DP	65.5	545

Unit: μ s per iteration

Table 3 The computation time of Monte Carlo (MC) code in each iteration on the Cell processor and the Intel Xeon processor [12]. The results are presented using either single precision (SP) or double precision (DP). The data of Xeon has been scaled up to the value based on 3.2GHz.

	Cell	Xeon
SP	19.8	832
DP	34.5	1558.8

Unit: μ s per iteration

Table 4 The computation time of the image processing code for 1/48 of the image (360x192) in each iteration on the Cell processor and the Intel Xeon processor [12]. The results are presented using either single precision (SP) or double precision (DP). The data of Xeon have been scaled up to the value based on 3.2GHz.

The Cell is again estimated to show a much higher performance for the image processing code, as is shown in Table 4. This code also fits into the SPE LS space. Since the image processing program is less computationally intensive than the MC code, the communication overheads (not accounted for in the results) could be higher but are not expected to dominate.

In both MC and image processing tests, the performance difference between using single precision (SP) and double precision (DP) on the Cell was not huge, but the tests of the peak performance on the Cell processor [12] show that the peak performance of SP on the Cell is much greater than that of DP, so one explanation is that these two programs tested did not approach the peak SP performance.

3.4.4 Summary

Two codes were successfully ported to the Cell architecture, but significant effort was needed in reprogramming communications, and the resulting code was verbose and Cell specific. The memory limit of Cell should be considered in porting general applications. For computation-intensive programs which fit into the SPE LS space, the Cell processor is seen to provide a much higher performance than a conventional (Intel Xeon) processor. Single Precision codes are expected to perform better than Double Precision, but that was not observed for these codes. Scaling to more than one Cell processor was not investigated, and whether the Cell processor can be useful as a supercomputer architecture remains to be seen.

4 Conclusions

There is currently a diverse range of modern supercomputing architectures. We showed that porting existing HPCx codes to other modern systems is possible, but the effort involved is dependent on the architecture of each system and the application code.

The Blue Gene/L is a classical MPP machine aimed at massive scaling by exploiting low power and low cost processors. Indications are that porting to Blue Gene should be straightforward, but only codes that scale well and do not have large memory requirements are suitable. For those codes which are suitable, the utilisation of many processors should be a cost effective way of achieving high performance.

The NEC SX8 is a SMP cluster with vector processors. A code was successfully ported to this system directly from HPCx, but it was found not to take advantage of the vector architecture. Some codes on HPCx may be more suited to vector processing. In general, it is

expected that effort will be required in adapting code to utilise vector processors, and there likely exist algorithms which are inherently more suited to scalar processors.

The Cray XD1 is a supercomputer based on the Direct Connected Processor (DCP) architecture. The effort required for porting was observed to be dependent on applications. Similar performance and scaling to HPCx was observed.

The STI Cell processor employs a novel architecture with a multi-core design to provide high performance as well as high power efficiency. Significant effort is needed for using this type of processor as it requires (non MPI) programming of communications specific to the architecture. For the applications whose size fit into the SPE LS space, the Cell was observed to substantially outperform a conventional processor. It is not clear at this stage how well applications could scale beyond one Cell chip.

5 Acknowledgements

We wish to thank Radoslaw Ostrowski, John Spray, Adrian Jackson, Jon Hill, and George Kyriacou, for providing results for this paper. We also would like to thank Kenton D'Mellow and Dyane Goodchild for their assistance.

6 References

- [1] "User's Guide to the HPCx Service (Version 2.02)", HPCx service website, <http://www.hpcx.ac.uk/support/documentation/UserGuide/HPCxuser/HPCxuser.html>
- [2] "An Overview of the BlueGene/L Supercomputer", IBM and Lawrence Livermore National Laboratory, Proceedings of the SC 2002 conference, Baltimore, 2002.
- [3] "User Guide to EPCC's BlueGene/L Service (Version 1.0)", Blue Gene User Information webpage, <http://www.epcc.ed.ac.uk/~bgapps/UserGuide/BGuser/BGuser.html>
- [4] High Performance Computing Center Stuttgart, <http://www.hlrs.de>
- [5] HLRS Platforms – NEC SX-8, <http://www.hlrs.de/hw-access/platforms/sx8>
- [6] HLRS NEC SX-8 cluster Documentation, http://www.hlrs.de/hw-access/platforms/sx8/user_doc
- [7] CSEXD1 Cluster Specifications, The Distributed Computing Group at CCLRC Daresbory Laboratory, <http://www.cse.clrc.ac.uk/disco/csexd1/csexd1/shtml>
- [8] AMD Opteron™ Product Data Sheet, June 2004
- [9] Cray XD1 Technical Specifications – Release 1.4, 2005
- [10] Direct Connected Processor (DCP) Architecture, Cray Inc website, <http://www.cray.com/products/xd1/architecture.html>

- [11] Introduction to the Cell multiprocessor, J.A. Kahle *et al.* IBM Journal of Research and Development, <http://researchweb.watson.ibm.com/journal/rd/494/kahle.html>
- [12] J.Spray, A.Jackson, J.Hill, "Computational science on the Cell processor", Technical report of EPCC, 2006.
- [13] A. Gray, L. Smith, J. Hein, J.M. Bull, F. Reid, O. Kenway, B. Dobrzeleck, A. Trew, "An Application Performance Comparison of HPCx to EPCC's Blue Gene/L Service", HPCx technical report, HPCxTR0511,2005, http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0511.pdf
- [14] X. Guo, "Application Performance on Future High Performance Computing Architectures ---- Blue Gene", MSc dissertation of EPCC, 2006, <http://www.epcc.ed.ac.uk/msc/dissertations-0506.htm>
- [15] AMBER, the introduction webpage of AMBER on HPCx website, <http://www.hpcx.ac.uk/research/chemistry/amber.html>
- [16] MD Benchmarks for Amber, CHARMM and NAMD, the introduction webpage of AMBER benchmark, <http://amber.scripps.edu/amber8.bench2.html> (Updated: April, 2005)
- [17] The Virgo Consortium, <http://www.virgo.dur.ac.uk>
- [18] ASC FLASH Centre University of Chicago, FLASH User's Guide Version 2.5, February 2005.
- [19] R.Ostrowski, "Porting, Profiling and Optimising an AMR Cosmology Simulation", MSc dissertation of EPCC, 2006, <http://www.epcc.ed.ac.uk/msc/dissertations-0506.html>
- [20] Smith, W. & Forester, T. 1996, DL POLY: A General Purpose Parallel Molecular Dynamics Simulation Package. *J. Molec. Graphics* **14**, 136; Smith, W., Yong, C. & Rodger, M. 2002, DL POLY: Applications to Molecular Simulation. *Molecular Simulation* **28**, 385; Todorov, I. & Smith, W. 2004, DL POLY 3: The CCP5 National UK Code for Molecular Dynamics Simulations. *Phil. Trans. R. Soc. Lond. A* **362**, 1835; http://www.cse.clrc.ac.uk/msi/software/DL_POLY/
- [21] PCHAN webpage, <http://www.cse.clrc.ac.uk/arc/pchan.shtml>; M. Ashworth, PCHAN package, CCLRC.
- [22] G.Kyriacou, "Application Performance on Future High Performance Computing Architectures", MSc dissertation of EPCC, 2006, <http://www.epcc.ed.ac.uk/msc/dissertations-0506.html>
- [23] M.Kistler, M.Perrone, and F.Petrini, "Cell multiprocessor communication network: build for speed", IEEE Micro, 26(3), 2006