Vector vs. Scalar Processors: A Performance Comparison Using a Set of Computational Science Benchmarks

Mike Ashworth, Ian J. Bush and **Martyn F. Guest**,
Computational Science & Engineering Department,
CCLRC Daresbury Laboratory

ABSTRACT: *Despite a significant decline in their popularity in the last decade vector processors are still with us, and manufacturers such as Cray and NEC are bringing new products to market. We have carried out a performance comparison of three full-scale applications, the first, SBLI, a Direct Numerical Simulation code from Computational Fluid Dynamics, the second, DL_POLY, a molecular dynamics code and the third, POLCOMS, a coastal-ocean model. Comparing the performance of the Cray X1 vector system with two massively parallel (MPP) micro-processor-based systems we find three rather different results. The SBLI PCHAN benchmark performs excellently on the Cray X1 with no code modification, showing 100% vectorisation and significantly outperforming the MPP systems. The performance of DL_POLY was initially poor, but we were able to make significant improvements through a few simple optimisations. The POLCOMS code has been substantially restructured for cache-based MPP systems and now does not vectorise at all well on the Cray X1 leading to poor performance. We conclude that both vector and MPP systems can deliver high performance levels but that, depending on the algorithm, careful software design may be necessary if the same code is to achieve high performance on different architectures.*

KEYWORDS: vector processor, scalar processor, benchmarking, parallel computing, CFD, molecular dynamics, coastal ocean modelling

1. Introduction

Vector computers entered the scene at a very early stage in the history of high-performance computers. The first systems to earn the epithet supercomputers were the Cray-1 vector computers (although the term has with hindsight been applied to earlier scalar systems). The concept of vectorisation increases the computational performance by efficiently pipelining identical calculations on large streams of data. Pipelined arithmetic and load/store units generate a peak or asymptotic calculation rate of one operation per clock cycle. This may be increased further by the provision of multiple, parallel units. The provision of vector instructions avoids the performance being limited by the instruction issue rate and distinguishes the genuine vector processor from pipelined scalar architectures. One of the major advantages of the vector processor architecture is the ability to keep a large number of memory operations in flight at any one time. Optimisation of programs for such computers generally requires restructuring of loop nests to generate long unit-strides inner loops.

All of the key computational science groups in the UK made use of vector supercomputers during their halcyon days of the 1970s, 1980s and into the early 1990s [1]-[3]. Their codes were optimised for these systems, notably the Cray 1, Cyber 205, Amdahl VP, Cray X-MP, Cray Y-MP, Fujitsu VPP and Cray J90. In the past decade there has been a decline in the popularity of vector processor based systems in favour of massively-parallel distributed-memory systems (MPP) built from cache-based microprocessors. This decline is shown dramatically in Figure 1. Here we have plotted the fraction of the aggregate *rmax* performance in the top 500 high-end systems [4] delivered by different processor architectures. This has been primarily a matter of cost, the driver being the financial benefit of utilising commodity chips in high-end systems. The high-end market has been, and will continue to be, a small fraction of the total server market, and there are considerable economies to be gained from a common architecture which will scale from the desk-side server, through departmental-scale machines and beyond to the high-end.

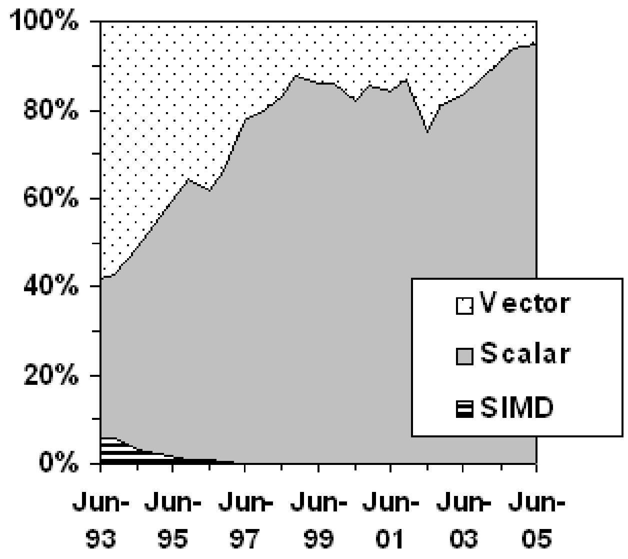

Figure 1. The fraction of the total *rmax* performance in the top 500 high-end systems delivered by different processor architectures.

This effect has been no less dramatic in the UK where the flagship computing provision in the last ten years has been dominated by MPP systems. The Cray T3D, introduced in 1994 was followed by a Cray T3E, an SGI Origin 3000, SGI Altix and, most recently, an IBM p690 cluster.

Some of the former vendors of vector hardware, notably Fujitsu and Hitachi, have ceased production of vector machines in favour of commodity microprocessor systems. Others, notably Cray and NEC, are still bringing vector computers to market. Many workers have noted the decline in the percentage of peak performance attainable by real application codes (e.g. [5]). On vector systems many codes were capable of achieving around 50% of peak, whereas on current microprocessor systems only 10% or less is commonly experienced. This goes a long way to negating the cost advantage of the commodity-based systems.

When vector systems were dominant, many scientists became skilled in adapting and writing codes for efficient execution on vector processors and some important codes in use today still carry the legacy of this. The code is typically organised with long, unit-stride inner loops to allow the compiler to generate long vectors. Such is the benefit of long vectors that conditional execution is often implemented by computing everywhere and masking out unnecessary data with a vector mask. Intermediate results are often promoted from scalars to vectors increasing the amount of memory required. The priority is to generate unit-stride, vectorisable loops.

Different techniques are required for programming the cache-based microprocessors that are to be found in most of today's high performance PCs, workstations and massively parallel processors (MPP). Processor speeds have out-paced memory access speeds to such an extent that the major factor limiting performance is now the memory access pattern. Unit-stride access is still important but for a different reason; that is, to ensure optimum use of cache, rather than to optimise processor performance. Inner loops should be small, so that their data fits within cache; intermediate results should be scalars so that they remain in cache or in registers; and vector masking should be discarded in favour of computing only where necessary. Blocking loop nests to increase data locality often helps performance, whereas on vector systems it is totally inappropriate.

Which is superior out of vector and scalar high-end systems remains an open question in many quarters. This report is aimed at addressing this question by comparing the performance of full application codes on both types of architecture using systems currently available and in service. Section 2 describes the main features of the systems we have used, and section 3 describes each code and reports the performance results. Section 4 discusses some detailed performance analysis for two of the codes and we present conclusions in section 5.

2. Description of systems

Cray X1

Oak Ridge National Laboratory took delivery of a Cray X1 system in 2003 and have carried out an extensive evaluation of the system [6]. The Cray X1 is a scalable parallel computer with symmetric multiprocessor (SMP) nodes where the processors are very good vector processors with weak scalar performance. The basic building block of a Cray X1 system is the SSP. An SSP consists of a vector processor that has 32 vector registers of 64 elements each, implemented in two vector pipelines and operating at 800 MHz. An SSP also has a 400-MHz scalar processor. The peak performance of an SSP is 3.2 gigaflops. The two vector units in an SSP have an 800-MHz clock and can move a column (vector) of numbers from memory into high-speed registers or initiate an operation with a string of resultants obtained two per clock cycle after initial vector setup. These operations are accomplished much more efficiently than typical microprocessor operations and generate many more resultants per second and higher sustained computation rates.

The Cray X1's compiler supports two strategies for exploiting the hierarchical architecture. In MSP or multistreaming mode a single long vectorised loop or an unvectorised outer loop is divided and spread across the eight vector units of a single MSP. The compiler is also able to vectorise a long outer loop and multistream a shorter inner loop if the dependency analysis allows this.

The compiler also supports SSP mode in which each SSP is treated as a separate processor.

IBM p690+

HPCx is the UK's leading and most recent National High Performance Computing Service. It is a large IBM p690+ cluster consisting of 1600 1.7 GHz POWER4+ processors with 1.60 TB of memory and IBM's High performance Switch (HPS), formerly known as "Federation". Note that all reported timings for the p690+ system were carried out after the upgrade of the HPS microcode contained in Service Pack 7 (SP7) which yielded a significant improvement in the performance of the switch.

The IBM p690 frame is a 32-way shared-memory system with a complex cache architecture. There are two POWER4+ processors per chip each with its own Level 1 data and instruction caches and with a shared on-chip Level2 cache. Eight processors (four chips) are integrated into a Multi-Chip Module (MCM) together with 128 MB of Level 3 cache and 8GB of main memory. Four MCMs make up the p690 frame. The total Level 3 cache of 512 MB per frame and the total main memory of 32 GB per frame is shared between the 32 processors of the frame.

Configuration		**Cray X1**	**IBM p690+**	**SGI Altix 3700**
Processor			POWER4+	Itanium2
Processor clock (GHz)		0.8	1.7	1.3
Processor peak (Gflop/s)		3.2	6.8	5.2
Processors/node		32	32	256
Memory/node (GB)		16	32	512
Memory/ processor (GB)		0.5	1	2
Caches	L1	0.5 MB	64/32 kB per CPU	16/16 kB per CPU
	L2	-	1.5 MB per 2-CPU chip	256 kB per CPU
	L3	-	128 MB per 8-way MCM	3MB (1.3GHz) 6MB (1.5GHz)
OS		Unicos/mp 3.0	AIX 5.2	SGI Linux
FORTRAN compiler		Cray Fortran 5.3	xlf 8.1	Intel 7.1

Table 1. Hardware and software characteristics of the systems under comparison in the paper. The Cray X1 "processor" is an SSP as all codes in this paper were run in SSP mode.

SGI Altix 3700

CSAR at the University of Manchester operate a flagship 512 Itanium2 processor SGI Altix 3700 system known as newton. Of the 512 processors, 384 have a clock speed of 1.3 GHz and 128 are 1.5 GHz. By selecting different batch queues one can select which processors are used. The system has 1.5 GB of memory per processor and uses the NUMAflex interconnect. The node size (the size of a single system image) is 256 processors and, although MPI jobs can span nodes, the system is currently configured with a maximum job size of 250 processors.

The primary characteristics of the three systems are summarised in Table 1.

3. Performance of Applications

DL_POLY

DL_POLY [7] is a general-purpose molecular dynamics simulation package designed to cater for a wide range of possible scientific applications and computer platforms, especially parallel hardware. DL_POLY supports a wide range of application areas, including [8] ionic solids, solutions, metals, zeolites, surfaces and interfaces, complex systems (e.g. liquid crystals), minerals, bio-systems, and those in spectroscopy. Significant enhancements to the code's capabilities have arisen from the recent release of the distributed data (or domain decomposition) version (DL_POLY 3) [9].

The port to the Cray X1 at ORNL was very straightforward. The only problems encountered were some very minor C pre-processing issues, but once these were solved the code compiled using more or less the standard options in the makefile for Cray machines. Once compiled the tests all run successfully.

To optimise the code initially a simulation of Sodium Chloride was used. This was chosen partially because it is a very well understood case, being part of the acceptance tests for the HPCx machine at Daresbury, and partially because it is a fairly simple example. The actual simulation is of 216,000 ions run for 200 timesteps.

The performance was initially measured on 8 multi-streaming processors (MSP), and was found to be poor taking 916 seconds. This compares with 146 seconds on 32 processors of the HPCx machine. However running on 32 single streaming processors (SSP), which should be equivalent to 8 MSPs, the time roughly halved to 454 seconds, indicating that the code does not multi-stream well, at least as currently written. However this is still somewhat worse than HPCx.

The code was then profiled with the Cray Performance Analysis Tool (PAT). This indicated that two routines were dominating, PARLINK and EWALD_SPME. These routines were known not to consume much time on HPCx, and so they were investigated further. Examining a compiler generated listing of PARLINK indicated that the routine was not being vectorised. This was not surprising; the algorithm generates link lists to hold which atoms a given atom interacts with and this is known to vectorise poorly. Rewriting the routine to avoid do-while loops and backward jumps allowed the compiler to vectorise the code, and the elapsed time on 32 SSPs dropped to 292 seconds.

For EWALD_SPME a compiler listing indicated that, although the compiler was vectorising the inner loop, the loop length was very short, typically of length 6 to 10. The use of loop jamming techniques, that is rewriting a number of nested loops as a single loop, increased the length of the loop body that could be vectorised, and this further reduced the time on 32 SSPs to 171 seconds, a factor of 2.7 faster than the original SSP time.

Though still slower than HPCx the code can still further be improved, though the gains are becoming harder to achieve since at this stage no particular routine in the code was standing out as being expensive.

Number of Processors	Cray X1	IBM p690+
16	337	
32	171	146
64	91	78

Table 2. Times in seconds for the DL_POLY NaCl 216,000 ion benchmark case on the Cray X1 and the IBM p690+ cluster.

An initial examination of the scaling of the optimised code shows that, as on HPCx, it scales quite well. Table 2 shows the timing results for the optimised code, the number of processors being the number of SSPs on the Cray.

SBLI/PCHAN

The direct solution of the equations of motion for a fluid remains a formidable task and simulations are only possible for flows with small to modest Reynolds numbers. Within the UK the Turbulence Consortium (UKTC) has been at the forefront of simulating turbulent flows by direct numerical simulation (DNS). UKTC has developed a parallel version of a code to solve problems associated with shock/boundary-layer interaction [10].

The code (SBLI) was originally developed for the Cray T3E and is a sophisticated DNS code that incorporates a number of advanced features: namely high-order central differencing; a shock-preserving advection scheme from the total variation diminishing (TVD) family; entropy splitting of the Euler terms and the stable boundary scheme. The code has been written using standard Fortran 90 code together with MPI in order to be efficient, scalable and portable across a wide range of high-performance platforms. The PCHAN benchmark is a simple turbulent channel flow benchmark using the SBLI code.

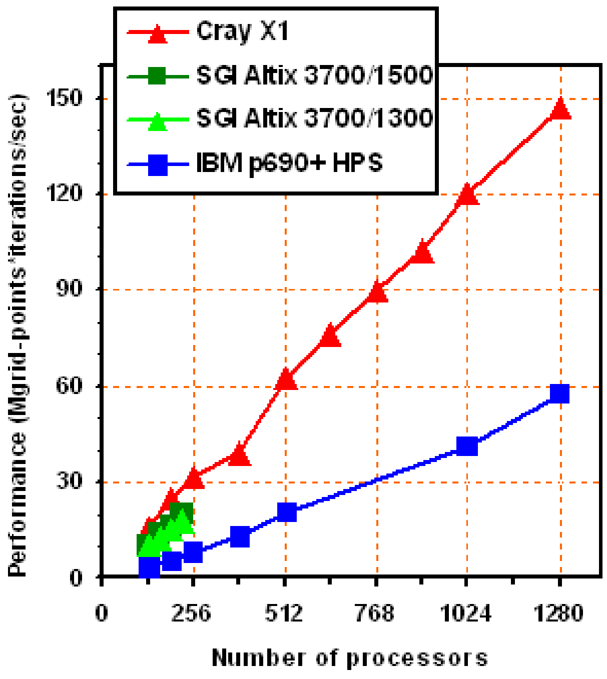

Figure 2 Performance of the PCHAN T3 benchmark on the Cray X1, the IBM p690+ cluster and the SGI Altix 3700 (1.3GHz and 1.5GHz processors).

The most important communications structure within PCHAN is a halo-exchange between adjacent sub-domains. Providing the problem size is large enough to give a small surface area to volume ratio for each sub-domain, the communications costs are small relative to computation and do not constitute a bottleneck.

Figure 2 shows performance results for the T3 data case, a grid of 360x360x360, from the Cray X1, the IBM cluster and the SGI Altix and shows ideal scaling on all systems. Hardware profiling studies of this code have shown that its performance is highly dependent on the cache utilisation and bandwidth to main memory [11].

It is clear that memory management for this code is taking place more efficiently on the Altix than on the p690+. The match to the streaming architecture of the Cray X1 is excellent. Timings for the Cray X1 and the IBM p690+ are shown in Table 3 together with the performance ratio. At 128 processors the ratio is 4.3, dropping to 2.5 at 1280 processors, possibly as the sub-domain size reduces and the vector length of the inner loops becomes smaller.

Number of processors	PCHAN		
	IBM p690+	Cray X1	Ratio Cray/IBM
128	1245	290	4.3
192	812	189	4.3
256	576	147	3.9
512	230	75	3.1
768	146	52	2.8
1024	112	39	2.9
1280	81	32	2.5

Table 3. Execution times in seconds for the PCHAN T3 benchmark on the Cray X1 and IBM p690+ including the performance ratio between the Cray and the IBM.

POLCOMS

The Proudman Oceanographic Laboratory Coastal Ocean Modelling System (POLCOMS) has been developed to tackle multi-disciplinary studies in coastal/shelf environments [11]. The central core is a sophisticated 3-dimensional hydrodynamic model that provides realistic physical forcing to interact with, and transport, environmental parameters. The hydrodynamic model is a 4-dimensional finite difference model based on a latitude-longitude Arakawa B-grid in the horizontal and S-coordinates in the vertical. Conservative monotonic PPM advection routines are used to ensure strong frontal gradients. Vertical mixing is through turbulence closure (Mellor-Yamada level 2.5).

In order to study the coastal marine ecosystem, the POLCOMS model has been coupled with the European Seas Regional Ecosystem Model (ERSEM) [12]. Studies have been carried out, with and without the ecosystem sub-model, using a shelf-wide grid at 12km resolution. This results in a grid size of approx. 200 x 200 x 34. In order to improve simulation of marine processes, we need accurate representation of eddies, fronts and other regions of steep gradients. The next generation of models will need to cover the shelf region at approximately 1km resolution.

In order to assess the suitability of the POLCOMS hydrodynamic code for scaling to these ultra-high resolutions we have designed a simulated 2km shelf-wide benchmark which runs (without the ecosystem model) at a grid size of 1200x1200x34. In order to keep benchmark run times manageable, the runs were kept short (100 timesteps) and the initialisation and finishing times were subtracted from the total run time. The performance is reported in Figure 3 as the amount of work (gridpoints $\times$ timesteps) divided by the time.

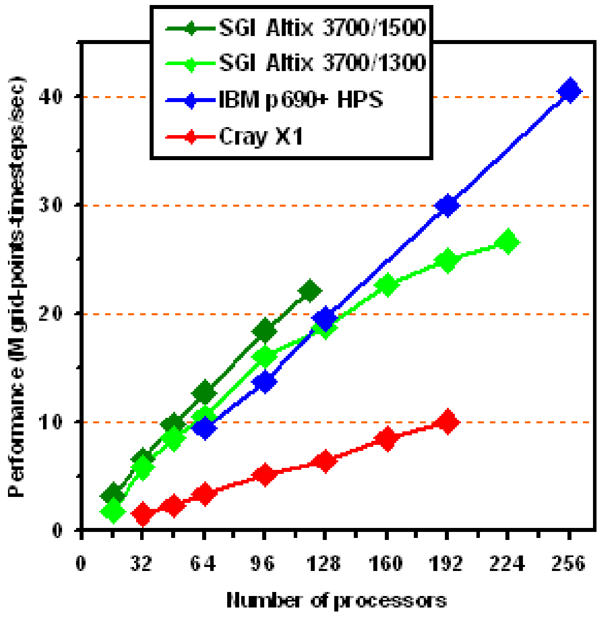

Figure 3. Performance of the POLCOMS 2km benchmark on the Cray X1, the IBM p690+ cluster and the SGI Altix 3700 (1.3GHz and 1.5GHz processors).

As with the SBLI code communications are limited almost entirely to nearest neighbour boundary exchange operations and we see almost perfect linear scaling on the p690+ HPS. However, the scaling on the Altix 3700/1300 is starting to run out above 128 processors. The absolute performance is highly dependent on cache and memory issues [11]. However the Cray X1 is not competitive. Timings for the Cray X1 and the IBM p690+ are shown in Table 4 together with the performance ratio, but note that in contrast with Table 3 this is the IBM/Cray ratio not the Cray/IBM ratio!

Number of processors	POLCOMS		
	IBM p690+	Cray X1	Ratio IBM/Cray
128	542	6613	12.2
192	352	4381	12.5
256	260	3096	11.9
512	125	1617	13.0
768	86	1040	12.2

Table 4. Execution times in seconds for the POLCOMS 2km benchmark on the Cray X1 and IBM p690+ including the performance ratio between the IBM and the Cray.

4. Performance Analysis

We used Cray's Performance Analysis Tool (PAT) to investigate further the performance of the SBLI and POLCOMS codes. PAT can be run in a number of different ways to measure performance of an application as a whole, based on resource usage, accounting, and hardware performance, (pat_hwpc) or to measure performance at the granularity of individual functions, source code line number, etc. (pat_build and pat_report). We ran 64 processor (SSP) jobs of the benchmarks using pat_hwpc. Some key hardware counter numbers from the output are shown in Figure 4.

The values are represented as percentages e.g. the vector operations as a percentage of all operations, the vector memory references as a percentage of all memory references etc. For the average vector length this is given as a percentage of 64, the maximum vector length of the SSP.

It is clear that the PCHAN benchmark vectorises very well, as we suspected from the performance figures. Vector floating point operations and vector memory references are at 100%. On 64 processors the grid of 360x360x360 is partitioned into 90x90x90 sub-domains leading to an inner loop length of 90. The average vector length is very close to 45.

The POLCOMS code was originally a vector code with long inner loops. The data arrays were organised with the two horizontal dimensions first (contiguous in memory) and in some cases the horizontal dimensions were collapsed into a single index in order to increase the length of the inner loops. However the code has been significantly restructured to suit cache-based systems [13]. The major three-dimensional arrays now have the vertical dimension first. This helps cache re-use by allowing whole water columns to remain in cache e.g. for one-dimensional processes such as convection and vertical diffusion, or for use by a subsequent outer-loop iteration in the advection code.

The effect of this code structure on the Cray X1 is clear from Figure 4. Not all important loops have vectorised so that the proportion of vector floating point operations is down at 90% and vector memory references at 67%. Given the rather slow speed of the scalar processor on this system, this is quite enough dramatically to restrict the performance. Furthermore the average vector length is low at about 11.

5. Conclusions

Our experience is that the porting of codes from MPPs to the Cray X1 presents no great difficulty. The performance of the scalar unit is quite poor. To achieve satisfactory performance it is essential that the code should vectorise fully, if necessary by investing effort to resolve vectorisation problems. To this end the use of PAT and compiler listings was very useful. We found that SSP mode gave better performance than MSP mode, though we made no effort to improve the multi-streaming performance of the code.

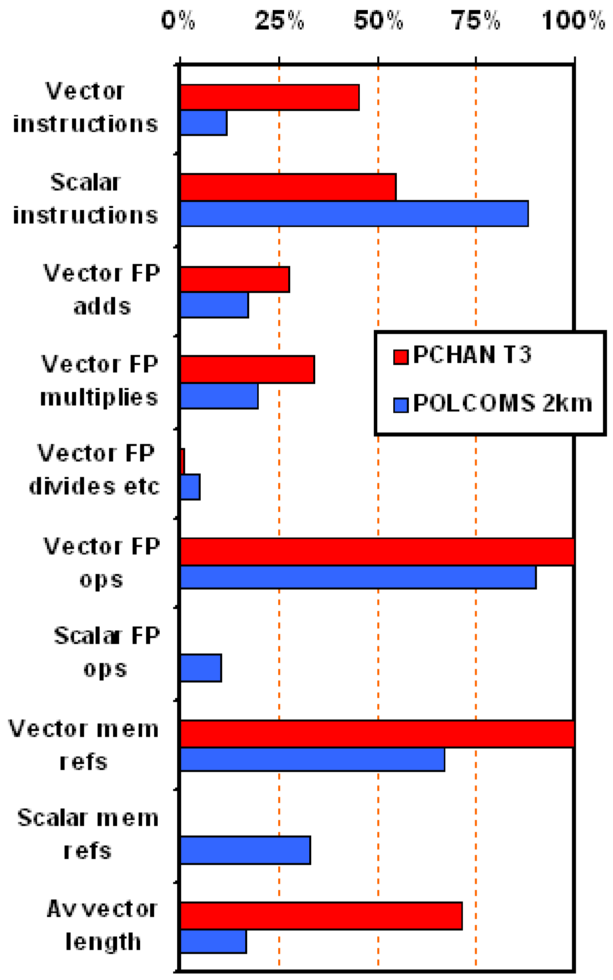

Figure 4. Comparison of the percentage of total operations, memory references, etc. from hardware performance counters for the PCHAN T3 and POLCOMS 2km benchmarks.

We have compared the performance of three different codes, one an explicit finite-difference CFD code, one a molecular dynamics code and one a coastal-ocean code, between the Cray X1 vector system and two massively parallel (MPP) micro-processor-based systems. We find three rather different results.

The SBLI PCHAN benchmark performs excellently on the Cray X1 with no code modification, showing

100% vectorisation and significantly outperforming the MPP systems. The performance of DL_POLY was initially poor, but we were able to achieve a performance improvement of 2.7 through a few simple optimisations. The POLCOMS code has been substantially restructured for cache-based MPP systems and now does not vectorise at all well on the Cray X1, leading to poor performance.

The UK Meteorological Office, who operate an NEC SX-6 vector system, use a modified version of the POLCOMS code as a part of the UK's coastal flood warning system. Their modifications make a dramatic improvement to the performance on vector systems and the POLCOMS development team is working on ways to incorporate the vector modifications into the main version of the code in a portable way.

We conclude that both vector and MPP systems can deliver high performance levels but that, depending on the algorithm, careful software design may be necessary if the same code is to achieve high performance on different architectures.

References

[1] *The use of Vector Processors in Quantum Chemistry; Experience in the U.K.*, M.F. Guest and S. Wilson, Daresbury Laboratory Preprint, DL/SCI/P290T; in Supercomputers in Chemistry', ed. P.Lykos and I. Shavitt, A.C.S. Symposium series 173 (1981) 1.

[2] *Application of the CRAY-1 for Quantum Chemistry Calculations, V.R. Saunders* and M.F. Guest Computer Physics Commun. 26 (1982) 389.

[3] *The Study of Molecular Electronic Structure on Vector and Attached Processors: Correlation Effects in Transition Metal Complexes*, Supercomputer Simulations in Chemistry, ed. M. Dupuis, Lecture Notes in Chemistry, 44, (1986) 98 (Springer Verlag)

[4] TOP500 supercomputer sites http://www.top500.org/

[5] *Parallel Processing in Environmental Modelling*, M. Ashworth, in Parallel Supercomputing in Atmospheric Science: Proceedings of the Fifth Workshop on the Use of Parallel Processors in Meteorology, eds. G-R. Hoffmann and T. Kauranne, (1993), 1-25, (World Scientific).

[6] *Cray X1 Evaluation Status Report*, P.A. Agarwal et al (29 authors), Oak Ridge National Laboratory, Oak Ridge, TN, USA, Technical Report ORNL/TM-2004/13 http://www.ccs.ornl.gov/CRAYEvaluationTM2004-15.pdf

[7] *DL_POLY: A general purpose parallel molecular dynamics simulation package*, W. Smith and T.R. Forester, J. Molec. Graphics **14** (1996) 136.

[8] *DL_POLY: Applications to Molecular Simulation*, W. Smith, C. Yong and M. Rodger, Molecular Simulation **28** (2002) 385.

[9] *The DL-POLY Molecular Simulation Package*, W. Smith, http://www.cse.clrc.ac.uk/msi/software/DL_POLY/

[10] *Direct Numerical Simulation of Shock/Boundary Layer Interaction*, N.D. Sandham, M. Ashworth and D.R. Emerson, http://www.cse.clrc.ac.uk/ceg/sbli.shtml

[11] *Single Node Performance of Applications and Benchmarks on HPCx*, M. Bull, HPCx Technical Report HPCxTR0416, (2004) http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0416.pdf

[12] *Eddy Resolved Ecosystem Modelling in the Irish Sea*, J.T. Holt, R. Proctor, M. Ashworth, J.I. Allen, and J.C. Blackford, in Realizing Teracomputing: Proceedings of the Tenth ECMWF Workshop on the Use of High Performance Computing in Meteorology, eds. W. Zwieflhofer and N. Kreitz, (2004), 268-278, (World Scientific).

[13] *Optimization of the POLCOMS Hydrodynamic Code for Terascale High-Performance Computers*, M. Ashworth, J.T. Holt and R. Proctor, HPCx Technical Report HPCxTR0415, (2004) http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0415.pdf

Acknowledgments

The authors would like to thank Oak Ridge National Laboratory (ORNL) and the UK Engineering and Physical Sciences Research Council (EPSRC) for access to machines.

About the Authors

Mike Ashworth is Head of the Advanced Research Computing Group in the Computational Science & Engineering Department (CSED) at CCLRC Daresbury Laboratory and has special interests in the optimisation of coupled environmental models for high-performance systems. Ian Bush is a computational scientist at CSED with specialisation in high performance parallel algorithms for computational chemistry, molecular

dynamics and materials applications. Martyn Guest is Associate Director of CSED and leads the HPCx Terascaling Team. He is a leading researcher in the application of computational chemistry methods on high-performance computers. All three authors contribute to the HPCx Terascaling Team.

All authors can be reached at CCLRC Daresbury Laboratory, Warrington WA4 4AD, UK, E-Mails: m.ashworth@dl.ac.uk, i.j.bush@dl.ac.uk and m.f.guest@dl.ac.uk.