Performance Comparison of Various Capability Benchmarks on the IBM p690+ and the SGI Altix 3700

Mike Ashworth, Ian J. Bush, Martyn F. Guest, Martin Plummer and Andrew G. Sunderland
Computational Science and Engineering Department, CCLRC Daresbury Laboratory, Daresbury, Warrington, Cheshire, WA4 4AD, UK
Email: m.ashworth@dl.ac.uk, m.f.guest@dl.ac.uk

Joachim Hein
Edinburgh Parallel Computing Centre, University of Edinburgh, JCMB, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ, UK

Abstract

The UK's flagship scientific computing services are currently provided by two competing architectures: the IBM p690+ cluster, with 1.7 GHz POWER4+ processors (and previously p690 with 1.3 GHz POWER4 CPUs), operated by HPCx and the SGI Altix 3700, with 1.3 GHz Itanium 2 processors at CSAR. We present the results of a benchmarking programme to compare the performance of these systems across a range of capability applications of key importance to UK science. A total of ten codes are considered from a diverse section of application domains; materials science, molecular simulation and molecular electronic structure, computational engineering and environmental science. We have also included a number of kernel benchmarks in order better to understand the performance differences seen from the application codes.

In addition to the UK systems, our analysis includes results from "ram", an SGI Altix 3700 (with 1.5 GHz processors) at ORNL, plus those derived from the older SGI Origin 3800/R14k ("teras", at SARA) and the HP/Compaq AlphaServer SC system, "TCS1", at the Pittsburgh Supercomputing Centre.

We find a range of performance with some algorithms performing better on one system and some on the other. Four of the nine applications considered are seen to perform significantly better on the SGI Altix 3700 than on the IBM p690+, namely the engineering codes (PCHAN and THOR), the environmental code POLCOMS, and NAMD, the molecular simulation code. There can be little doubt that this is a consequence of the superior memory architecture of the Altix, at least in the case of the engineering and environmental codes, for these codes are known to be extremely demanding on memory bandwidth. In the case of NAMD, the communication demands of the simulation code are undoubtedly exposing the current limitations in Release 1 of the HPS microcode on the IBM system.

In contrast, the electronic structure codes, CASTEP, CPMD and CRYSTAL, and to a lesser extent DLPOLY, show superior performance on the IBM p690+, as indeed do the PDSYEVD-based matrix diagonalisation benchmarks. It is no coincidence that all four application codes that exhibit enhanced performance on the p690+ have been the subject of considerable optimisation efforts by the HPCx Terascaling Team and their collaborators. These efforts are discussed in the paper. AIMPRO is found to exhibit similar performance on both systems.

Contents

1. INTRODUCTION

For many years the provision of centralised High-Performance Computing (HPC) to academic research groups in the UK has been through the funding of a small number of Centres. The Centres collaborate to some extent in providing a service to the national user base, but they also inevitably find themselves competing for limited funding. The current national strategy has been developed by the HPC Strategy Committee (HSC) and is executed by the Engineering and Physical Sciences Research Council (EPSRC) on behalf of UK scientific community. This strategy comprises a series of procurements at approximately three-year intervals awarding a six-year contract to a Consortium that unites a leading HPC hardware vendor with one of the academic-based Centres. The two most recent procurements have led to the establishment of an SGI-based solution at CSAR at the University of Manchester and IBM systems operated by the HPCx Consortium.

CSAR is the National High Performance Computing (HPC) service run on behalf of the Research Councils by Computation for Science (CfS), a consortium comprising Computer Sciences Corporation (CSC), Silicon Graphics and the University of Manchester. The CSAR service started in November 1998 and is run as a 6-year PFI-contract to provide a world-class supercomputing facility to enable world-class science by UK researchers. This research is enabled by the CSAR supercomputers, now including a 256 Itanium2 processor SGI Altix 3700/1300, "Newton", and a 512 processor SGI Origin3800, "Green".

HPCx is the name of the UK's new National High Performance Computing Service. It is a large IBM p690 cluster whose configuration is specifically designed for high-availability capability computing. We define *capability computing* as running jobs which use a significant fraction of the total resource in contrast with *capacity computing*, which is the running of a large number of smaller jobs. HPCx is a joint venture between the Daresbury Laboratory of the Central Laboratory for the Research Councils (CLRC) and Edinburgh Parallel Computing Centre (EPCC) at the University of Edinburgh. IBM (UK) Ltd has been chosen as the hardware supplier for the six-year duration of the project.

Perhaps the key challenge for the UK's flagship services service is to deliver on the capability aspirations of the community across a broad spectrum of scientific and engineering disciplines. In this article we describe the initial progress towards this goal. Following a summary of the systems under discussion in section 2, we provide an overview of the challenges involved in capability-based computing (section 3) and specifically the challenges involved in communications and in managing the memory hierarchy (section 4). Sections 5 to 9 describe the current levels of delivered performance from ten well-known codes from five different applications areas on both the IBM and SGI systems. These codes are representative of the current workload on the HPCx system and illustrate the challenges of achieving Terascale performance from a range of algorithms.

2. THE SYSTEMS

2.1. The HPCx Phase 1 system

Delivery of the Phase 1 system commenced on 4th October 2002. The full user service opened officially on 9th December 2002, although many users had benefited from up to a month's early access prior to this date. The system comprises 40 "Regatta" pSeries 690 (p690) SMP compute nodes connected by the "Colony" SP Switch2.

Each shared memory node has 32 1.3GHz POWER4 processors and 32 GBytes memory giving the system an aggregate memory of 1.28 TBytes. The POWER4 processor has dual floating point units each of which, through the fused multiply-add instruction, is capable of delivering two results per clock cycle. This gives each processor a peak performance of 5.2 Gflop/s and the whole system a peak of some 6.6 Tflop/s.

In order to increase connectivity to the switch and improve communications throughput, each compute node is configured as four 8-way logical partitions (LPARs). The "Colony" SP Switch2 allows each LPAR to have two connections (PCI adapters) into the switch fabric (dual plane), providing approximately 20 μsec latency and 350 MBytes/sec bandwidth. Two additional 16 processor nodes are provided as I/O systems. The system runs AIX version 5, with GPFS for file system support (18TBytes EXP500) and HSM for backup and archive to tape storage (35TBytes LTO tape library).

2.2. The HPCx Phase 2 system

The HPCx Phase 2 system was built up over a period of time and the user service switched over from Phase1 to Phase2 on 29th April 2004. As with Phase 1 there was a period of about one month's free early user access prior to the commencement of the full user service. The system now has 50 p690+ SMP compute nodes. The nodes are similar to the Phase1 nodes but with the faster 1.7 GHz POWER4+ processors. Each 32-way SMP node has 32GBytes of memory (as in Phase 1) giving a system aggregate memory of 1.6 TBytes. Some improvements to the Level 3 cache firmware mean that in some cases serial performance may scale somewhat better than the raw clock speed ratio of 1.31.

The most significant difference with the Phase2 system is the introduction of IBM's High Performance Switch (HPS), formerly known as "Federation". Each Regatta frame has two network adapters and there are two links per adapter, making a total of four links between each of the frames and the switch network. In contrast to the SP Switch, the HPS connects directly to the GX inter-processor communications bus of the p690, obviating the need for dividing the frame into LPARs, so each p690 is now run as a single 32-way system.

All results presented here from the p690+ system were generated using Release 1 of the HPS switch microcode. This initial release is known to have significant performance limitations, arising from issues surrounding the receive-side microcode and associated locks. Early results from the restructured microcode are encouraging, with the associated Q3 release of the software promising to address a number of these communication limitations. We plan to revise the results presented here in due course.

2.3. The CSAR Newton system

The CSAR 'Newton' system entered user service on 1st October 2003. It is an SGI Altix 3700 system with 256 1.3 GHz Itanium2 processors and 384 GBytes total memory connected by SGI's NUMAflex interconnect. The 1.3 GHz Itanium2 processors have a peak performance of 5.2 Gflop/s - the same theoretical peak as the processors in the Phase 1 HPCx service. A number of processors are reserved for interactive, compilation use limiting the maximum job size on this system to 228 processors. The operating system on Newton is a 64-bit version of Linux, which runs with a Single System Image (SSI) of 64 processors although MPI jobs can run across multiple SSIs.

2.4. The ORNL Ram system

"Ram" is a 256-processor SGI Altix 3700 system under evaluation by the Center for Computational Sciences (CCS) at Oak Ridge National Laboratory (ORNL). "Ram" is unique in the CCS in that it has a very large, shared memory. The CCS is evaluating this architecture for use in scientific processing for the DOE Office of Science. Specific areas of interest are data analysis, shared memory versus distributed memory programming paradigms, and the scalability of the Linux operating system.

"Ram" has 256 Intel Itanium2 processors running at 1.5 GHz, each with 6 MBytes of L3 cache, 256K of L2 cache, and 32K of L1 cache. "Ram" has 8 GBytes of memory per processor for a total of 2 TBytes of total system memory. This system has a theoretical total peak performance of 1.5 Tflop/s. SGI normally supports up to 128 processors in a single system, but through collaboration with SGI, the CCS is testing an SSI of 256 processors.

2.5. Other systems

Although the primary focus of this paper is on the head-to-head comparison of the IBM p690 and p690+ clusters and the SGI Altix 3700 systems, we have included benchmark performance figures from other systems where appropriate (see [1]). These include the Cray T3E/1200E ("Turing"), now retired, and the SGI Origin 3800/R12k-400 system ("Green") operated by CSAR. Also included is the Compaq Alphaserver

ES45/1000, the TCS-1 system at Pittsburgh Supercomputing Centre (PSC)[1]. Readers should be aware that these systems cover a range of technologies, some of which are obviously more recent than others.

Table 1. Comparative specifications for the main systems used in this report.

	HPCx Phase1	**HPCx Phase2**	**Newton**	**Ram**
Model	IBM p690 cluster	IBM p690+ cluster	SGI Altix 3700/1300	SGI Altix 3700/1500
Interconnect	SP Switch2	HPS	NUMAflex	NUMAflex
Operator	HPCx[2]		CSAR[3]	ORNL[4]
Processor architecture	POWER4	POWER4	Itanium 2	Itanium 2
Number of processors	1280	1600	256	256
Single System Image (SSI) size	8	32	64	256
Processor clock (GHz)	1.3	1.7	1.3	1.5
Processor peak performance (Gflop/s)	5.2	6.8	5.2	6.0
Level 3 cache size (MB) per processor	16	16	3	6
Memory per processor (GB)	1	1	1.5	8
Total system memory (TB)	1.28	1.6	0.38	2.0
Total system peak performance (Tflop/s)	6.6	10.9	1.3	1.5

[1] Pittsburgh Supercomputing Center, Pittsburgh, PA, USA, http://www.psc.edu/

[2] HPCx, UK, http://www.hpcx.ac.uk/

[3] Computer Services for Academic Research (CSAR), University of Manchester, UK, http://www.csar.cfs.ac.uk/

[4] Center for Computational Sciences (CCS) at Oak Ridge National Laboratory (ORNL), TN, USA, http://www.ccs.ornl.gov/

3. TOWARDS CAPABILITY COMPUTING

During the next few decades, advances in computing technologies will increase the speed and capacity of computers, storage, and networks by several orders of magnitude. At the same time, advances in theoretical, mathematical, and computational science will result in computational models of ever increasing predictive capability and utility. The key goal for computational scientists and engineers is then to harness the power offered by present and future high-performance computers to solve the most critical problems in science and engineering. Such a goal demands a capability-driven approach, an approach in which the full power of a Terascale computer is bought to bear on a given scientific problem through effective utilisation of all available resources - CPUs, memory, interconnect, and in many cases high levels of I/O performance. Capability Computing contrasts strongly with the alternative capacity-based approach where many more modest problems are tackled, often simultaneously, on a machine, each with less demanding requirements. The primary mission of HPCx is that of Capability Computing, an approach reflected by our drive to ensure that the majority of jobs on the IBM p690 cluster are capable of utilising at least a significant fraction of the available resource.

Before considering our progress to date, we provide a brief overview of the challenges that must be addressed if we are fully to realise the benefits offered by high-performance computing [2]. The current generation of parallel supercomputers are, of course, built from thousands of processors using commodity memories (DRAM, the same memories used in personal computers) interconnected by a communications fabric that, although fast by networking standards, is still far slower than access to local memory. To make full use of such machines, computational scientists and engineers must address the three major challenges outlined below:

3.1. Management of the memory hierarchy

A key characteristic of today’s parallel machines is the presence of a deep memory hierarchy. Each processor in a parallel supercomputer has a memory hierarchy - registers, on- and/or off-chip caches, main memory, and virtual memory (disk) - with each level requiring successively more time to access (latency) and having slower transfer rates (bandwidth). A parallel computer adds an additional level to this hierarchy—remote memory. In exploiting the capabilities of today's parallel machines, it is critical carefully to manage this memory hierarchy. A programming model based on the concept of non-uniform memory access (NUMA) explicitly recognises this hierarchy, providing numerous advantages to the scientific programmer. We note in passing that the Global Arrays (GA) library [3] implements a very efficient NUMA programming model.

3.2. Expression and management of concurrency

Most scientific algorithms are rich in parallelism, although understanding the performance of these algorithms on parallel supercomputers requires more information into just how much parallelism is needed and at what level of granularity. The analysis of ref. 1 suggests the following:-

- Fine grain parallelism is essential to obtain efficient sequential execution given the mismatch in speed between that of the processor and that of the memory subsystem. Further, modern super-scalar processors achieve peak speed by executing multiple instructions per cycle, increasing the demand for fine grain parallelism.
- The granularity of coarse grain parallelism is determined by the ratio of the single processor speed to the average latency of remote memory references. This latency is controlled by the characteristics of both the communications hardware and the algorithm.

Current portable parallel environments require the user manually to express coarse grain parallelism (*e.g.*, with MPI or Global Arrays), while relying upon compilers or libraries e.g. the basic linear algebra subroutines (BLAS) to describe the fine grain parallelism. Significant progress has been made in addressing this area of concurrency, a key requirement in any capability-driven scenario e.g. the applications work at Pittsburgh Supercomputing Centre and the CRYSTAL and POLCOMS applications on HPCx (see below).

3.3. Efficient sequential execution

While efficiently using a large number of processors is recognised as a key requirement in any capability-driven approach, the more fundamental issue of ensuring efficient use of each of these processors is often overlooked. This is a non-trivial problem; to illustrate the magnitude of this problem, it is useful to compare the B/F ratio (the bandwidth to memory in Bytes/sec divided by the speed of the processor in operations/sec) for vector and parallel supercomputers. With a large B/F ratio, a processor can keep its functional units busy because the required data can be rapidly transferred from memory to the units. A small B/F ratio means, however, that the processor may often sit idle, as it will be waiting for data to be moved from memory to the functional units. The mathematical approaches and algorithms developed in the Cray era often made explicit use of the high memory bandwidths of Cray supercomputers - the last supercomputer built by Cray Research, Inc. (Cray T90) had a B/F ratio of 15. This was sufficient to run many vector operations at full speed out of main memory. The B/F ratio for the processor/memory subsystem in ASCI White, on the other hand, is just 0.67-1.33. This means that many of the traditional algorithms used in computational science and engineering will not perform well on parallel supercomputers without substantial tuning, revision, or even replacement.

There are numerous examples of this effect, where commendably high levels of concurrency are nevertheless accompanied by low overall delivery of peak performance. On the 3,000 processor TCS-1 system at PSC, the atmospheric general circulation model code, CCM/MP-2D, achieved 83% scaling efficiency on 2048 processors, but only 23% of peak. On the same machine, Klaus Schulten's NAMD code sustained a speedup factor of 1599 using 2250 processors on a molecular dynamics simulation of the 327,000 atom F_1-ATPase system - exceptional scaling for a bio-molecular MD code - but only 18% of peak.

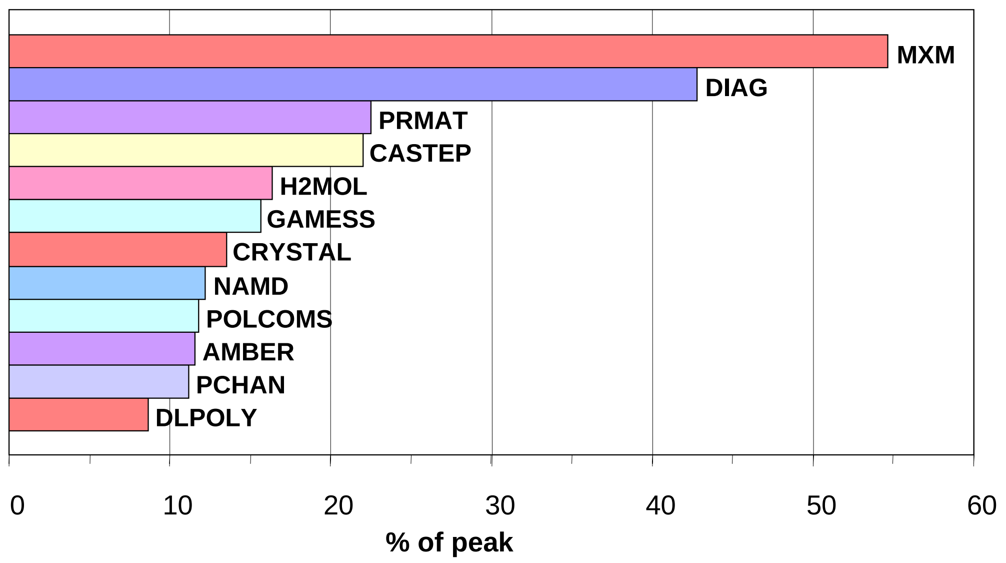

Figure 1. Best (usually single processor) performance results for ten applications codes and two application kernels on the IBM p690 using IBM's hpmlib to measure Mflop/s [4].

Designers of microprocessors and parallel supercomputers have attempted to mitigate the memory performance problem by a number of mechanisms e.g., by interposing a high-speed cache between the processor and main memory (with an 8 MByte cache, the B/F increases on the processors of ASCI White to 5.3). Clearly, if an algorithm can be structured so that the data are in cache when needed, the performance of the microprocessor will improve substantially. For many applications the key to successful utilization of the cache is data reuse, but this often requires a significant change in the algorithm. Other features designed to enhance memory performance include instruction and data pre-fetch, and the ability to schedule multiple outstanding memory requests. Although we expect to see substantial improvements in the B/F ratio of microprocessor-based parallel supercomputers in the next few years, there remains much to be gained by research on algorithms that promise to improve the sequential performance of parallel scientific and engineering codes.

In order to quantify the extent of this problem in the application space of interest, we have previously studied the percentage of sustained vs. peak performance delivered by both application kernels and full applications on the IBM p690 [4]. Using IBM's hpmcount, we found percentage delivery figures of between 9-23% of peak across ten of the key applications codes that feature in the HPCx service (see Figure 1). This study provided quantitative data on the impact of the marked decline in B/F ratios (the bandwidth to memory in bytes/sec divided by the speed of the processor in operations/sec) that characterise the present generation of MPP microprocessors compared to their vector counterparts. Attempts to mitigate these memory performance problems by interposing a high-speed cache, or hierarchy of caches, between the processor and main memory have, as this study revealed, met with only limited success.

While much of our work to date has focused on performance issues associated with increasing processor counts, it is clear from the above that realizing the full power of parallel supercomputers will demand an increased effort in developing new mathematical approaches and algorithms that make full use of caches. These require increasing the reuse of data already stored in the caches and using the ability of modern microprocessors to keep several memory references in flight at the same time to (effectively) reduce memory latency and increase memory bandwidth.

4. CHALLENGES OF CAPABILITY COMPUTING

4.1. Management of Memory Hierarchy

The previous section discussed the challenges associated with fully exploiting the capabilities of today's parallel machines. In this section we examine these challenges in more detail, through a range of simple benchmark codes.

Where appropriate, these benchmark codes have been studied on a range of platforms. Not all of these benchmarks are relevant on multiple platforms however, as they study performance issues specific to the HPCx platform. In these situations results are only presented on HPCx.

In the following section we discuss the performance and scaling of a range of applications across a variety of platforms. Many of the issues highlighted in this section contribute to the observed performance of these applications. Hence this section also provides insight into the performance and scaling of these applications on modern HPC systems.

Before considering the benchmarks we provide details of the memory hierarchy of HPCx. Following this we look at communication issues such as collective communications and overlapping communication with computation. We then consider sequential performance and the effective management of the memory hierarchy.

4.2. HPCx architecture

The HPCx system uses IBM p690+ Regatta nodes for the compute and IBM p690 Regatta nodes for login and disk I/O. Each Regatta node contains 32 POWER4+ processors. At present there are two p690 service nodes. As of June 2004, fifty p690+ nodes are used for compute jobs, offering a total of 1600 processors. The p690+ compute nodes utilise IBM POWER4+ processors. The POWER4+ is a 64-bit RISC processor implementing the PowerPC instruction set architecture. It has a 1.7 GHz clock rate, and has a 5-way super-scalar architecture with a 20 cycle pipeline. There are two floating point multiply-add units each of which can deliver one result per clock cycle, giving a theoretical peak performance of 6.8 Gflop/s. There is one divide and one square root unit, which are not pipelined.

Within a node there are 16 chips: each chip contains two processors with their own level 1 caches and a shared Level2 cache. The Level1 cache is split into a 32-kB data cache and a 64-kB instruction cache, has 128-byte lines, and is 2-way set associative and write-through. The Level2 cache is a 1.5-MB combined data and instruction cache, with 128 byte lines and is 8-way set associative and write-back.

The chips are packaged into a Multi-Chip Module (MCM) containing 4 chips (8 processors) and a 128 MB Level3 cache. The Level3 cache has 512 byte lines, and is 8-way set associative and write-back. Each p690+ node contains 4 MCMs and 32 GB of main memory. The MCMs are connected to each other and to main memory by a 4-way bus interconnect to form a 32-way symmetric multi-processor (SMP). Note that the Level3 caches lie on the memory side of the interconnect. Cache coherency is maintained on the Level2 caches via a snoopy bus protocol. This means that every processor in effect has access to all 512 MB of Level3 cache in the system.

4.3. Altix architecture

SGI's high-performance Itanium offering is known as the Altix 3000 Family of Servers and Superclusters. The high-end of this range is the SGI Altix 3700 Supercluster. The 3000 Series is built from a number of standard modules or *Bricks*: Compute Modules, Memory Modules, Interconnect Modules, etc. The Model 3700 offers Compute Modules (C-Bricks) with four processors, either 1.3 GHz Itanium2 with 3MB of Level3 cache, or 1.5 GHz Itanium2 with 6MB of Level3 cache. The Memory Modules (M-Bricks) allow up to 64 GBytes of memory with a 5-port crossbar per node board allowing a maximum aggregate bandwidth per brick of 20.4GBytes/sec.

The Itanium processor family is based on the Explicitly Parallel Instruction Computing (EPIC) architecture. The architecture is designed to execute up to six instructions per cycle, with a maximum of four memory, two integer, two floating point and three branch instructions. As in the POWER4, there is a fused multiply-add (FMA) instruction giving a peak floating point performance of four operations per cycle. The Itanium architecture uses software pipelining in order to hide latencies associated with loop execution and predication and control speculation to reduce the performance degradation associated with branches. Predication allows both sides of a branch to be executed and the discards the results from the invalid one. Control speculation allows loads to be hoisted above branches, thereby allowing execution to occur on both sides of the branch.

The Itanium has three levels of cache. Unlike the POWER4, where the Level3 cache is off-chip in the MCM, all three levels of the Itanium cache are on-chip. The Level1 cache is 32kBytes and split equally between instructions and data. Both are four-way associative with 64Byte cache lines and a one-cycle latency. The Level2 cache is a 256kBytes unified instruction and data cache. It is 8-way associative has a 128Byte line size and is organised into five banks. The latency varies by instruction type between five and eleven cycles. As stated above, the size of the Level3 cache depends on the system. It is also a unified cache. The latency varies from 12 cycles for an integer miss to 18 cycles for an instruction miss.

4.4. Parallel Execution - Communications

b. Collective communications

Many scientific applications use collective communications and to achieve good scaling on clusters of SMP systems these communications need to be implemented efficiently. Collective communications were included in the MPI standard to allow developers to implement optimised versions of essential communication patterns. A number of collective operations can be efficiently implemented using tree algorithms - including Broadcast, Gather, Scatter and Reduce. On systems where communication is faster within a node than between nodes, the tree algorithm can be constructed such that communications corresponding to branches of the tree at the same level should run at the same speed, otherwise the speed of each stage of the algorithm will be limited by the performance of the slowest communication.

Although this issue may apply generally to many clustered-SMP platforms, it is particularly important on HPCx, where communications within an LPAR are significantly faster than between LPARs. In considering communications performance, we continue to use the MPI Parallel Communications benchmarks from Pallas (PMB, Pallas MPI Benchmarks [5]). PMB considers a number of point-to-point communications (e.g. PingPong, Sendrecv and Exchange) plus a selection of MPI Collective Operations (Allreduce, Reduce, Reduce_scatter, Allgather, Allgatherv, Alltoall). Results for the former are reported in MBytes/sec, results for the latter as time (μsec) to complete. Each operation is run for a variety of message lengths (0 to 4194304 Bytes), with the collective operations performed for various combinations of the number of CPUs available. Focusing here on the latter, we have measured 64 CPU performance on the older SGI Origin 3800/R14k and HP/Compaq Alphaserver SC ES45/1000, and on the IBM p690+ and the SGI Altix 3700 systems. Normalising the performance with respect to that measured on the IBM p690+ (with release 1 of the HPS microcode) we find, perhaps somewhat surprisingly, two distinct categories of performance. While the p690+ performance is found to be superior in four of the collective benchmarks, notably Allreduce, Reduce,

Reduce_scatter and Allgatherv, this is not the case for Alltoall and Allgather when the SGI Altix 3700 systems clearly outperform the p690+. To illustrate this effect, we show the relative performance with respect to the IBM p690+ for both Allreduce (Figure 2) and Allgather (0).

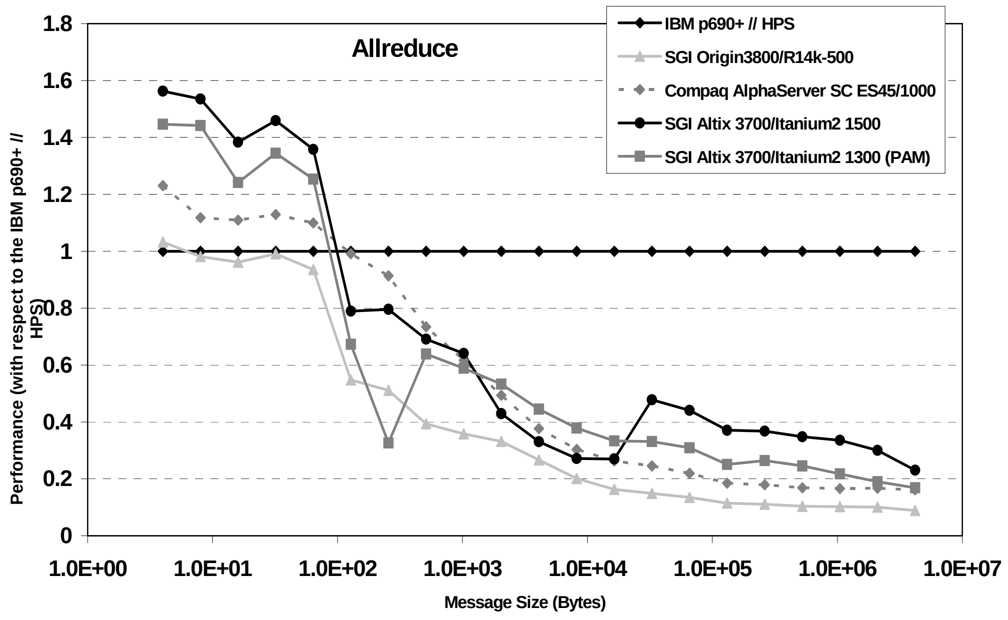

Figure 2. 64-CPU Performance for Allreduce (see text) as a function of message size (Bytes) relative to that measured on the IBM p690+.

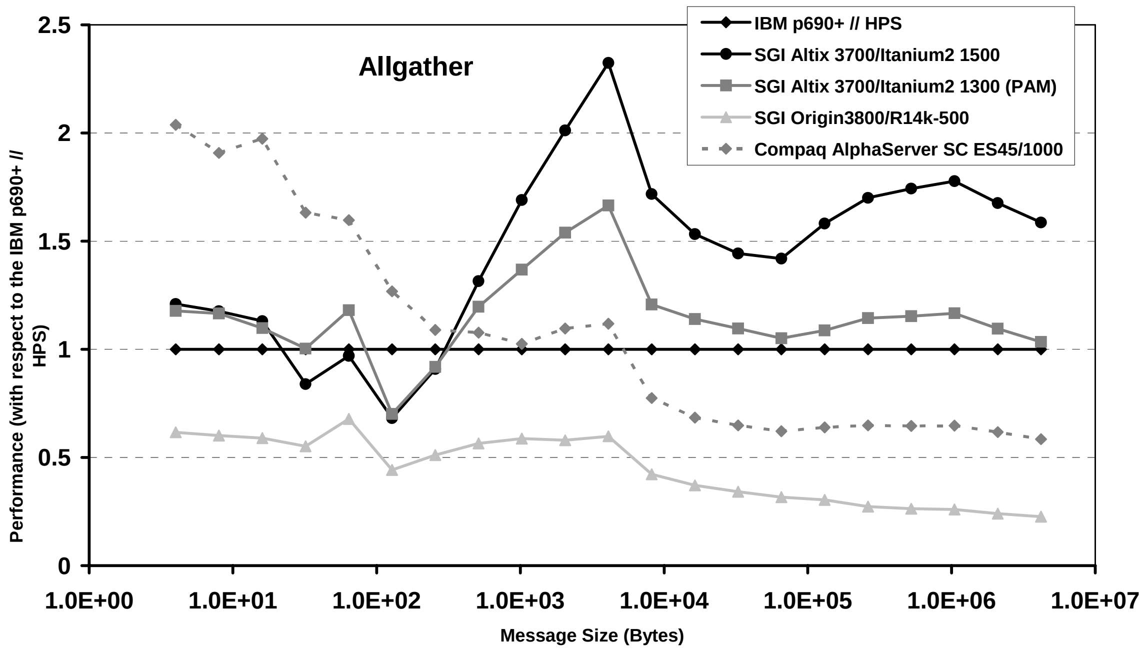

Figure 3. 64-CPU Performance for Allgather (see text) as a function of message size (Bytes) relative to that measured on the IBM p690+.

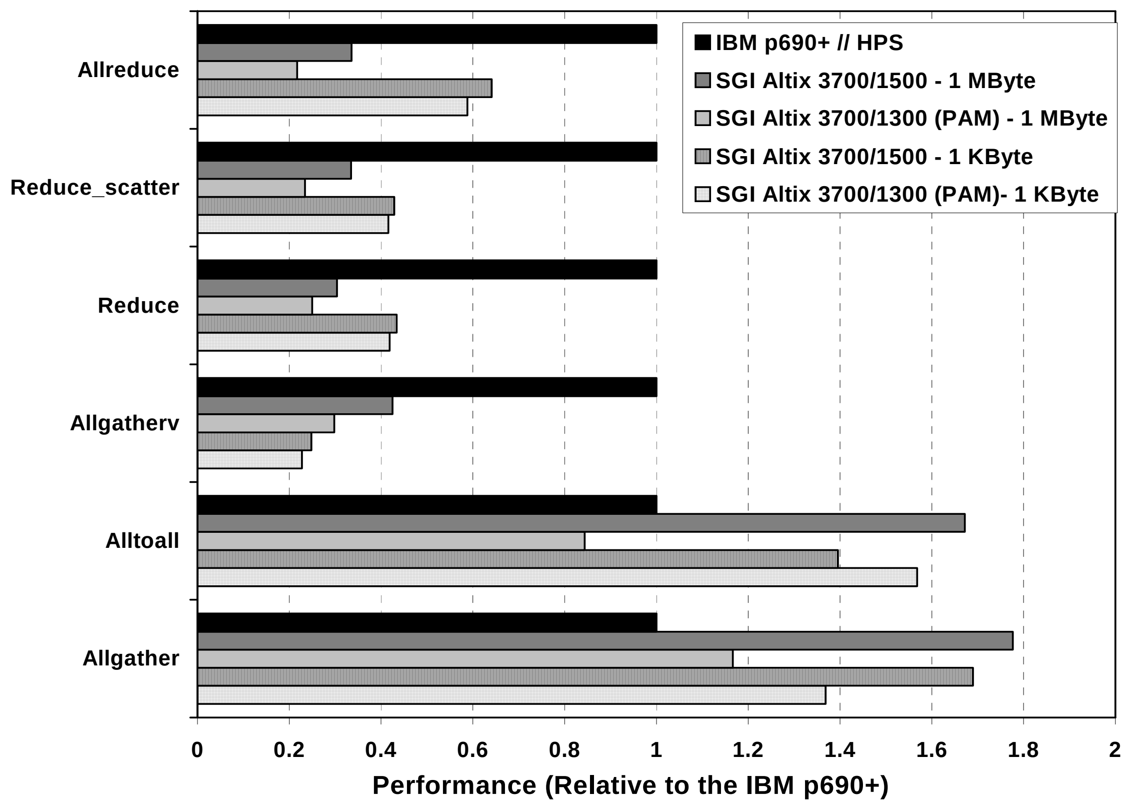

Figure 4. 64-CPU Performance for all collective operations on the SGI Altix 3700 relative to that measured on the IBM p690+ for messages of size 1KByte and 1MByte.

We note that for three of the four collectives for which the p690+ is faster, IBM have provided tuned, LPAR-sensitive versions. They are enabled by default for 64-bit codes and their use may also be controlled by the MP_SHM_CC environment variable, the documentation for which contains additional information.

When considering short messages (<128 Bytes), the ongoing latency issues around HPS are clearly apparent, with both Alphaserver and Altix systems outperforming the p690+. In the case of Allreduce, increasing message length leads rapidly to superior HPS performance, with factors against "ram", the Altix 3700/1500, of 1.56, 3.03 and 2.98 for message sizes of 1K, 4K and 1MByte respectively. "ram" in turn outperforms "newton" by a factor of 1.4 for long messages. The p690+ demonstrates far better performance compared to the HP/Compaq AlphaServer SC and SGI Origin 3800 for long messages - factors of 6.2 and 11.3 respectively for 4MByte messages. The Alphaserver demonstrates almost identical performance to the Altix 3700/1300 at this asymptote.

A quite different picture emerges from a consideration of Allgather. The Altix systems now outperform the p690+ with increasing message length - "ram" shows factors of 1.69, 2.32 and 1.78 for message lengths of 1K, 4K and 1MByte respectively. Interestingly, "ram" consistently outperforms "newton", with an asymptotic factor of 1.6; this superiority coincides with an almost identical asymptotic performance of HPCx and "newton". Even in this case, however, the p690+ demonstrates a significant advantage over the AlphaServer SC and Origin 3800 for long messages - factors of 1.7 and 4.4 respectively for 4MByte messages.

Focusing on the IBM p690+ and SGI Altix systems, we show in Figure 4 the relative performance for each of the six collective operations, again on 64 CPUs for two distinct message sizes, 1 KByte and 1 MByte. These again show the two classes of performance, with the p690+ exhibiting superior performance in Allreduce, Reduce, Reduce_scatter and Allgatherv, while the Altix systems are dominant in both Alltoall and Allgather.

In the former four cases, the performance lead of the p690 increases significantly on increasing the message length from 1K to 1MByte. In every case the SGI Altix 3700/1500 outperforms the 3700/1300, this superiority becoming more marked with increasing message length. The same effect is noted for the two collectives where the Altix outperforms the p690+ - in only one of the 12 possible cases does "newton" outperform "ram", in Alltoall with 1Kbyte messages.

In this section we have attempted to highlight the importance of making efficient use of the memory hierarchy and of understanding and optimising the communication patterns of our applications. In the following sections we examine the performance and scaling of a range of applications across a variety of disciplines, including materials science, molecular simulation, atomic and molecular physics, molecular electronic structure, computational engineering and environmental science.

5. MATERIALS SCIENCE

5.1. AIMPRO, CASTEP, CPMD and CRYSTAL

AIMPRO: AIMPRO (Ab Initio Modelling PROgram) is written by Patrick Briddon et al at Newcastle University [6]. It can be used to study both molecules and 3D periodic systems using a Gaussian basis set to solve the Kohn-Sham equations for the electronic structure. Recent developments of the code have reduced the dominance of parallel eigensolves of the Hamiltonian matrices by incorporating the Direct Inversion in the Iterative Subspace (DIIS) algorithm. After several parallel eigensolves, the calculation will typically have converged sufficiently to switch to DIIS. iterations, which are dominated by parallel matrix multiplications (PBLAS).

Two datasets were employed: Dat3 models an interstitial defect in bulk silicon with 433 atoms, 12124 basis functions and one k-point and Dat4 is the same system but simulated with 4 k-points. The increased number of k points for Dat4 makes for a more demanding calculation, but also allows for another level of parallelism to be employed: across the k-points. So for a 128 processor run Dat3 must perform all the parallel linear algebra, such as diagonalisations and matrix multiplies, over all 128 processors, but for Dat4 each k-point can be handled simultaneously by teams of 32 processors. This reduces the communications load (e.g. between 32-way frames on the p690+) and allows scaling to a larger number of processors. The computation involves three diagonalisations followed by five DIIS steps. Each stage takes roughly half the run time.

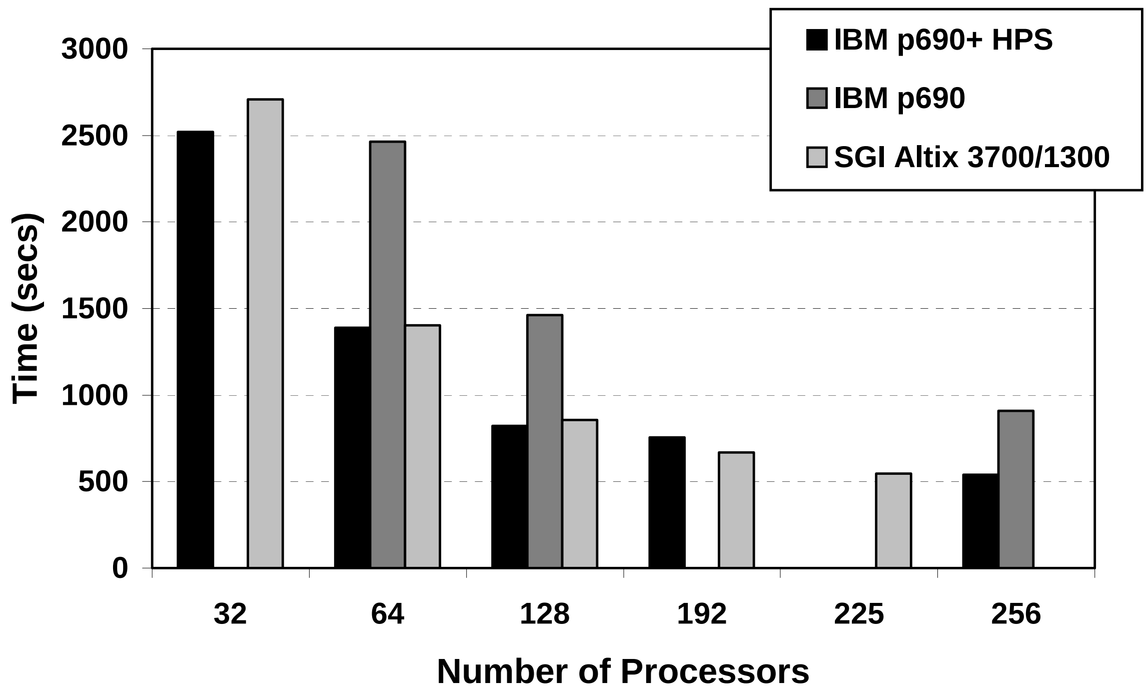

Figure 5. Execution time in seconds for AIMPRO for the Dat3 dataset (see text) on the IBM p690 and p690+ and SGI Altix 3700/1300.

Figure 5 shows execution times on the IBM p690 and p690+ and SGI Altix 3700/1300 for Dat3. Corresponding times for Dat4 are shown in Figure 6. Performance of the p690+ and the Altix is very similar and around twice that of the p690. For Dat3 you can see the p690+ starting somewhat faster on 32 processors, but not scaling quite so well and ending up a little slower on 192 processors. The same is true for Dat4 and the superior scaling from 128 to 192 processors over Dat3 due to the additional k-point parallelism can also be seen. The time in this benchmark is roughly split between ScaLAPACK eigensolves (PDSYEVX and PDSYEVD) and matrix-matrix multiplications in the DIIS code. Internal timing breakdowns show that the Altix is faster at matrix-matrix multiplies whereas the p690+ wins on the eigensolves (see section 7.2), so that the two effects roughly cancel.

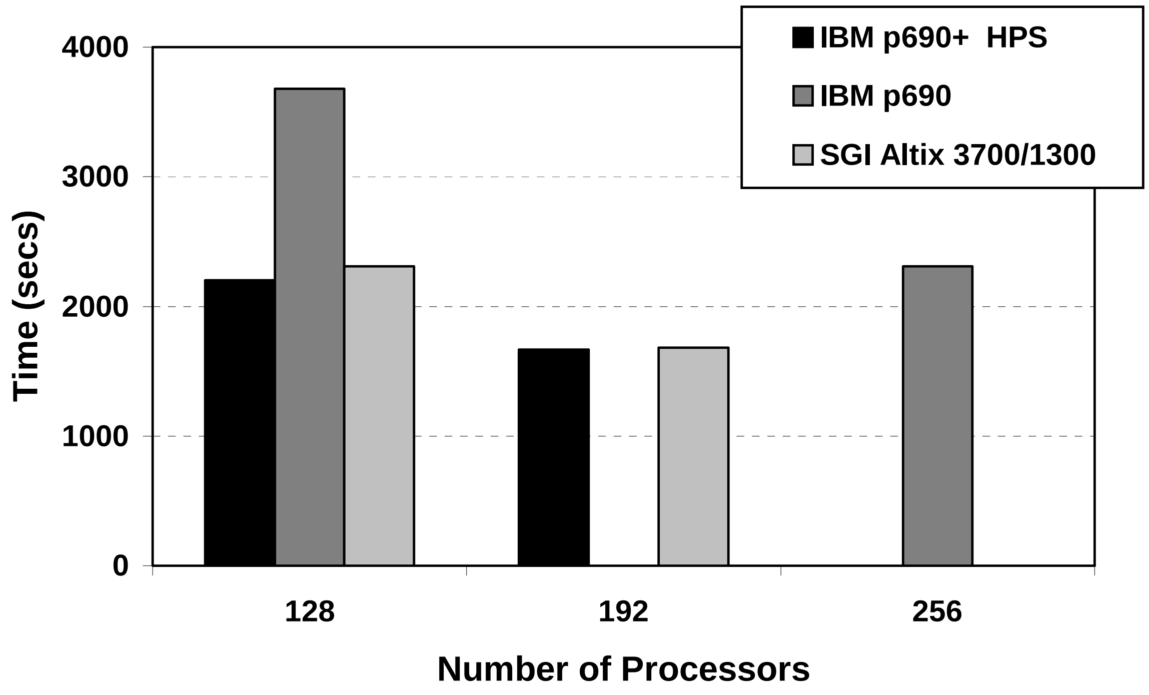

Figure 6. Execution time in seconds for AIMPRO for the Dat4 dataset (see text) on the IBM p690 and p690+ and SGI Altix 3700/1300.

CASTEP: The Materials Science code CASTEP [7] performs quantum mechanical (density functional) calculations of bulk materials, surfaces and interacting molecules. It has been developed within the UKCP Consortium and is distributed commercially by Accelrys Plc. The code expands electronic wave-functions in plane waves with many associated transformations between real and reciprocal space during a direct minimization of the electronic energy. The code uses three-dimensional Fast Fourier Transforms (3d-FFTs) on a distributed grid, performed using MPI_AllToAllV combined with serial FFTs. These electronic calculations occur in all the various scientific applications of the code. CASTEP uses a column distribution of the grid to maximize load-balancing throughout the program, resulting in two MPI_AllToAllV calls per 3d-FFT. CASTEP also has a higher level parallelism in which processors are first distributed among Brillouin zone sampling k-points and the 3d-FFT is then distributed among processors associated with a particular k-point.

Since January 2003 the official version of CASTEP has been a completely rewritten new modular code now officially known as Castep 3.0. Substantial optimizations have been made that have improved performance of this code on the IBM p690 systems by ensuring that the communications involved in the data redistribution (the MPI_AllToAllV calls surrounding the FFTs) is “SMP-aware” [8]. This means that, where possible and advantageous, the communications within the SMP node is separated from the inter-node communications. This optimisation is not appropriate on the SGI Altix.

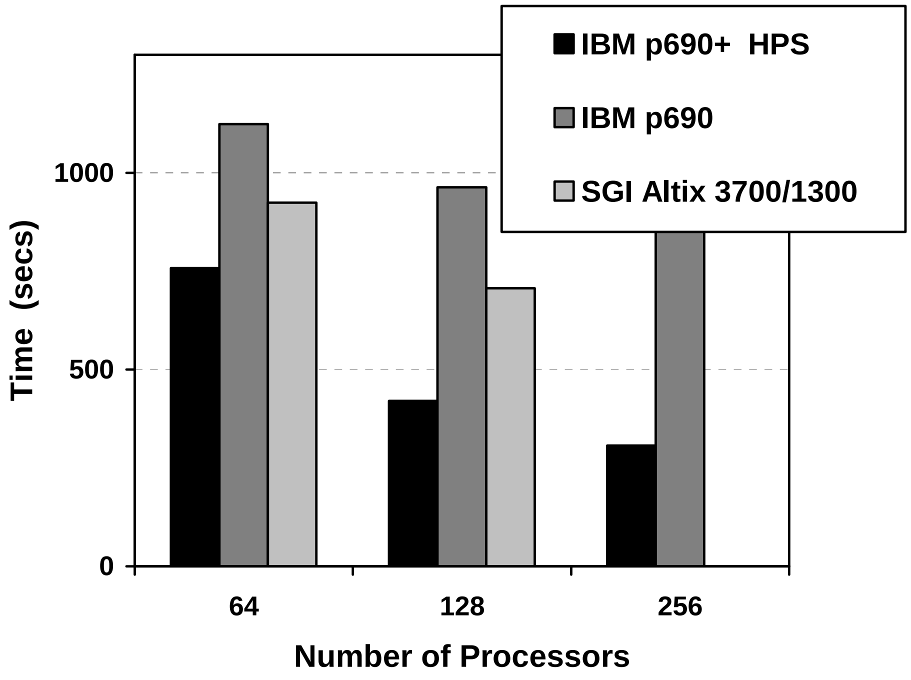

Figure 7. Elapsed time for the CASTEP TiN-8k benchmark on the IBM p690 and p690+ and the SGI Altix 3700/1300 systems.

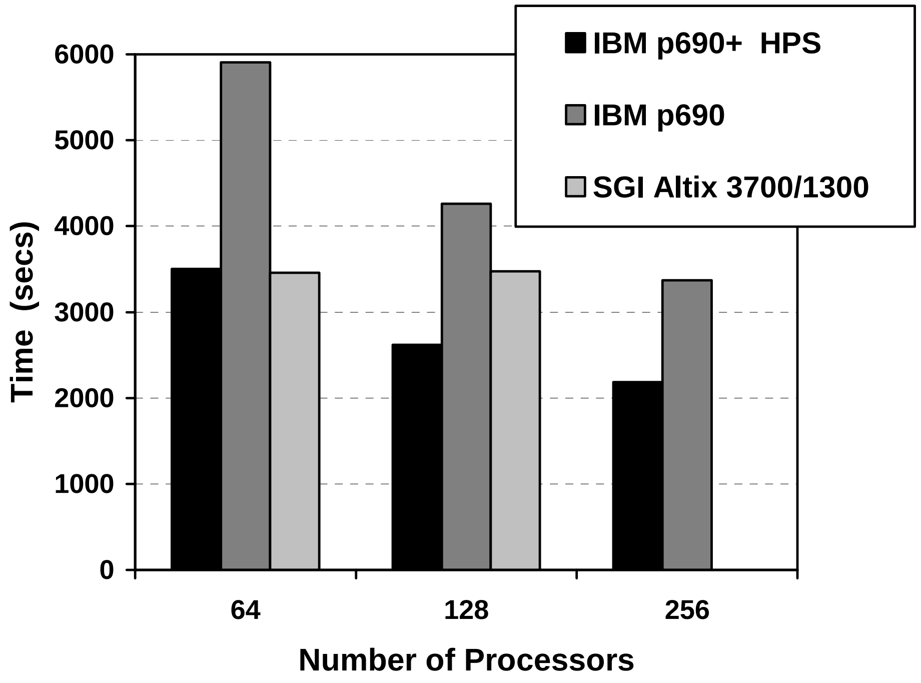

Figure 8. Elapsed time for the CASTEP Al2O3-G benchmark on the IBM p690 and p690+ and the SGI Altix 3700/1300 systems.

CASTEP benchmark TiN-8k is an 8 k-point spin-polarized total energy (SCF) calculation of TiN with a hydrogen defect. The cell contains 33 atoms and the 3d-FFT grid size is 108x36x36. Timings are presented for 34 SCF cycles. CASTEP benchmark Al2O3-G is a total energy calculation of Al_2O_3. There are 5 k-points, the cell contains 120 atoms and the 3d-FFT grid size is 60x60x160. This benchmark is designed specifically to test communications and the parallelism is set-up so that the MPI_AllToAllV is distributed across all the processors. This test is indicative of other systems which only have a single k-point. Timings are given for up to 12 iterations.

Times for the TiN-8k and Al2O3 benchmarks are shown in Figure 7 and Figure 8 respectively. We find that scalability is generally better on the p690+ than the p690 owing to the superior performance (especially latency) of the HPS interconnect. The 8-way k-point parallelism of the TiN-8k benchmark allows the collective communications to be limited to within the shared-memory LPARs of the p690+ system. The Altix performs less well than the p690+. This is particularly evident in the Al2O3-G case, where the performance of the code is critically dependent on the performance of MPI_AllToAllV across all the processors, and the Altix sees no performance gain in moving from 64 to 128 processors. There is still scope for optimisation of this code for the Altix.

CPMD: The CPMD code is based on the original code by Car and Parrinello [9]. It is a production code with many unique features and currently has about 150,000 lines of code, written in Fortran 77 with the MPI communications library. Besides the standard Car-Parrinello method, the code is also capable of computing many different types of properties, including the inclusion of quantum effects on nuclei with the path integral method and interfaces for QM/MM calculations. Since January 2002 the source code has been freely available for non-commercial use. Several thousand users from more than 50 countries have compiled and run the code on platforms ranging from notebooks to some of the largest high performance parallel computers [10].

Fast Fourier Transforms (FFT) are an essential part of all plane-wave calculations. In fact, it is the near linear scaling of FFTs that make large scale DFT calculations with plane wave basis sets possible. FFTs are used to transform the charge density and local potential between real space and reciprocal space. The number of these transforms is fixed and does not depend on the system size. On the other hand the transforms of wave functions from reciprocal space to real space and back (needed in the force calculation) has to be done for each state and dominates execution time for small and medium sized systems. Only for large systems (number of atoms larger than 1000) do the cubic scaling inner products and orbital rotations become dominant.

Different strategies are followed in parallel implementations of plane-wave / pseudo-potential codes. Parallelization of the CPMD code was done on different levels. The central parallelization is based on a distributed-memory coarse-grain algorithm that is a compromise between load balancing, memory distribution and parallel efficiency. This scheme achieves good performance on computers with up to about 200 CPUs, depending on system size and communication speed. In addition to the basic scheme, a fine-grain shared-memory parallelization was implemented. The two parallelization methods are independent and can be mixed. This allows us to achieve good performance on distributed computers with shared memory nodes and several thousands of CPUs, and also to extend the size of the systems that can be studied completely *ab initio*, to several thousand atoms.

Some methods implemented in CPMD allow a further level of parallelization. These methods, such as path-integral molecular dynamics or linear response theory, are embarrassingly parallel on the level of the energy calculation. Typically 2 to 16 copies of the energy and force calculation can be run in parallel. For these methods, an efficient use of computers with tens of thousands of CPUs can be envisaged. However, in this work only the main Car-Parrinello molecular dynamics part of the code has been used.

Our coarse-grain distributed-memory parallelization is driven by the distribution of wave-function coefficients for all states to all CPUs. Real-space grids are also distributed, whereas all matrices that do not include a plane-wave index are replicated (especially overlap matrices). All other arrays are only distributed if this does not cause additional communications. For a general data distribution in both spaces, each transform making up the 3D FFT would include communication between all processors. The data distribution in CPMD tries to minimize the number of communication steps while still having optimum load balancing in both spaces. This scheme requires only a single data communication step after the first transform. In addition, we can make use of the sparsity of the wave-function representation still present after the first transform and only communicate non-zero elements.

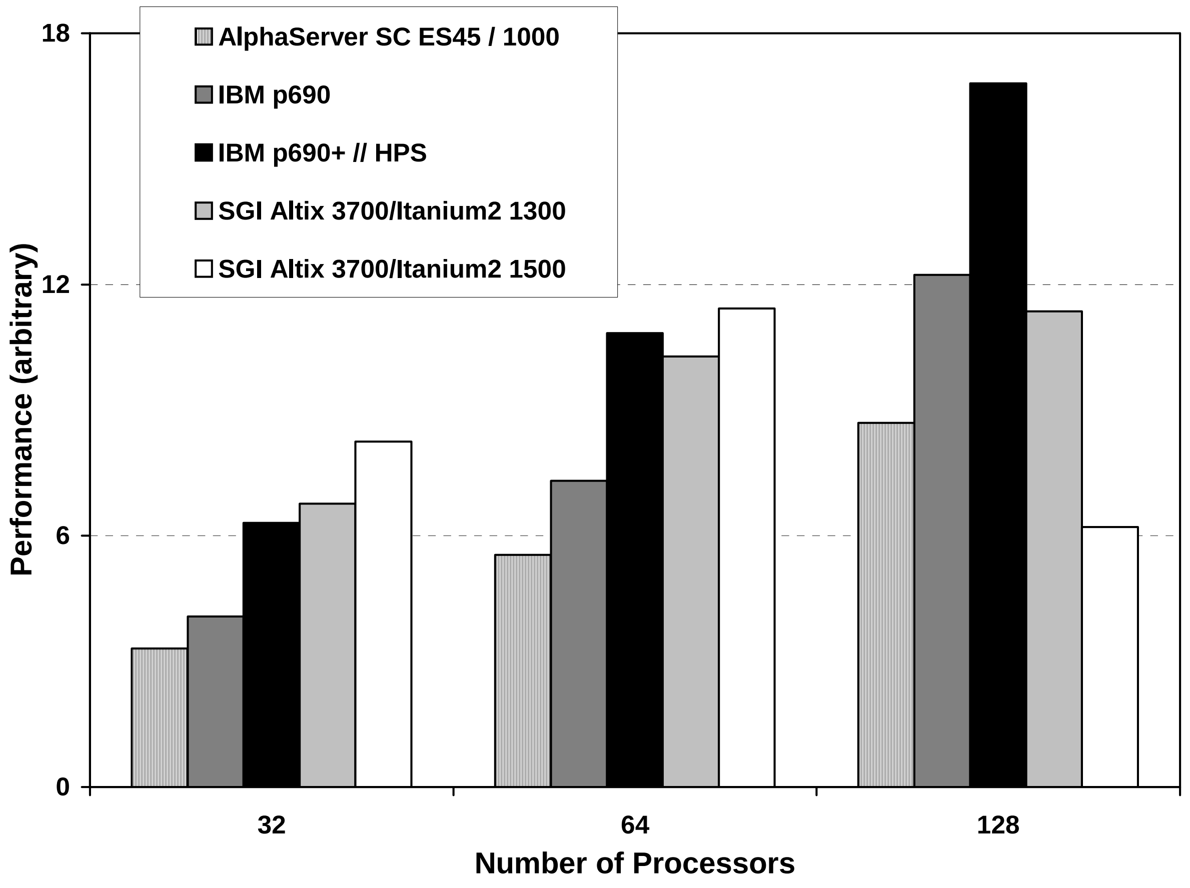

Figure 9. Performance of the CPMD code on the IBM p690 and p690+, the SGI Altix 3700/1300 and 1500, and the Alphaserver SC ES45/1000 for the 512-atom Silicon Cluster benchmark case.

The various load-balancing requirements are interrelated, and we use a heuristic algorithm to achieve near-optimum results. Our experience is that for all cases good load balancing is achieved for the reciprocal space. The restriction to full-plane distributions in real space, however, introduces severe problems in the case of a large number of processors. The number of planes available is typically about 50 for small systems and 200 to 300 for large systems. This restricts the maximum number of processors that can be used efficiently. The coarse granularity of this approach is also responsible for the appearance of magic numbers of processors where especially good performance can be achieved. This is no major problem because the appearance of these numbers is fully transparent. The efficiency of the scheme described above has a number of limitations which are discussed elsewhere [11].

Performance of a relatively small 512-atom Silicon Cluster benchmark on the Alphaserver SC ES45/1000, IBM p690 and p690+ and the SGI Altix 3700 /1300 is shown in Figure 9. While the SGI and Alphaserver systems rely on a standard MPI-based implementation, the IBM p690 and p690+ implementation features a hybrid parallelization mode for which, in addition to the coarse-grain MPI programming between LPARs, shared-memory loop-level parallelization within LPARs is achieved by using OpenMP compiler directives and multi-threaded libraries (BLAS and FFT) if available. In our tests, IBM’s multi-threaded ESSL library has been used. Compiler directives have been used to ensure parallelization of all longer loops (those that depend on the number of plane waves or the number of grid points in real space), and to avoid parallelization of the shorter ones. This type of parallelization is independent of the MPI parallelization and can be used alone or in combination with the distributed-memory approach. Tests on various shared-memory computers have shown that an efficient parallelization up to 16 processors can be achieved. The benefits of this hybrid scheme are clear. While the IBM systems scale well up to 128 processors, there is clear indication that the MPI-based implementation on the Altix systems shows inferior scaling. We are unable to explain the collapse in performance noted at 128 processors of the SGI Altix 3700/1500.

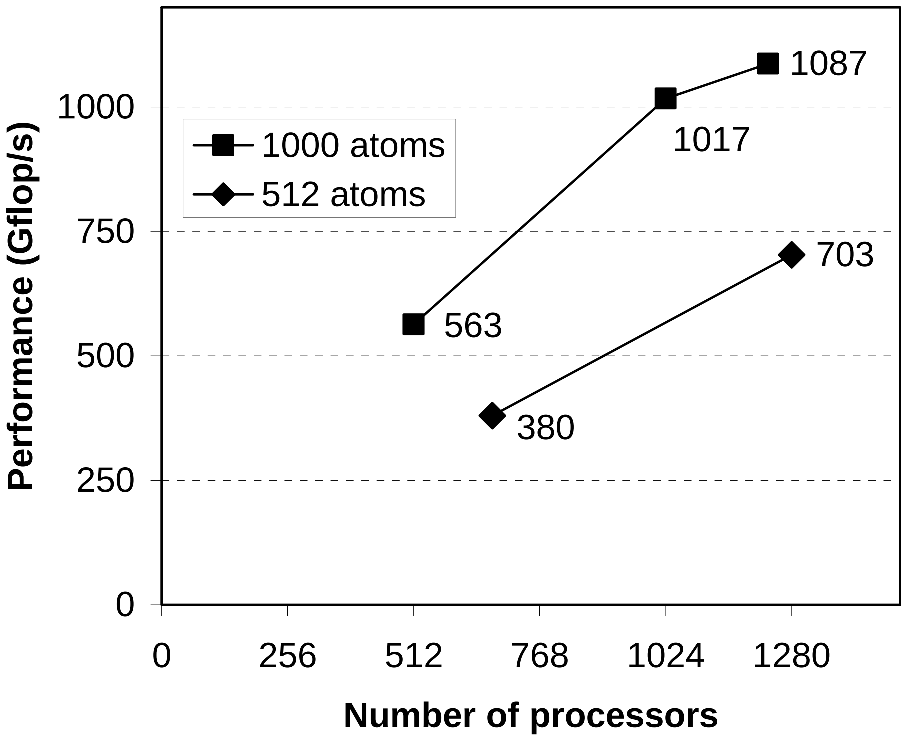

Figure 10. Performance of the CPMD code on the IBM p690 for a 1000-atom system

The performance of CPMD has been further tested on an IBM p690 using up to 1280 processors (Figure 10) ([38] and Curioni, private communication). The maximum performance achieved was above 1 Teraflop/s for a system consisting of 1000 atoms. This represents ~20% of the peak performance (33% of the Linpack Benchmark performance) and an overall parallel efficiency (calculated with an assumed single processor performance estimated from smaller systems) of 45%. This has to be compared with the maximum performance that can be achieved using BLAS3 library calls of about 66% of peak performance on a POWER4 processor. Tests with smaller systems reveal that the main performance bottlenecks are the memory bandwidth on the shared-memory nodes and the latency of the node interconnect. The novel mixed parallelization scheme presented here is opening new frontiers for the *ab initio* study of molecular systems with several thousand of atoms on modern supercomputers.

CRYSTAL: The CRYSTAL [12] code uses a periodic Hartree-Fock or density functional Kohn-Sham Hamiltonian and various hybrid approximations in the calculation of wave-functions and properties of crystalline systems. The wavefunctions are expanded in atom centred Gaussian type orbitals (GTOs) providing a highly efficient and numerically precise solution with no shape approximation to the density or potential. The code is developed within a long-standing collaboration between the Theoretical Chemistry Group at the University of Torino, Italy, and the Computational Materials Science Group at Daresbury Laboratory, and is widely distributed internationally. Details of the code and its applications can be found on the CRYSTAL web pages [13].

Recent enhancements to the parallel distributed data version of the code, MPP CRYSTAL 2003, include the incorporation of a somewhat faster, and more numerically stable version of the parallel Jacobi diagonalizer

[14]. Further, since it can diagonalize not only real symmetric but also Hermitian matrices, the ScaLAPACK library routines are no longer required. This is advantageous because the latter routines both scale poorly and also, dependent upon the eigenvalue spectrum, may require unacceptably large amounts of replicated memory.

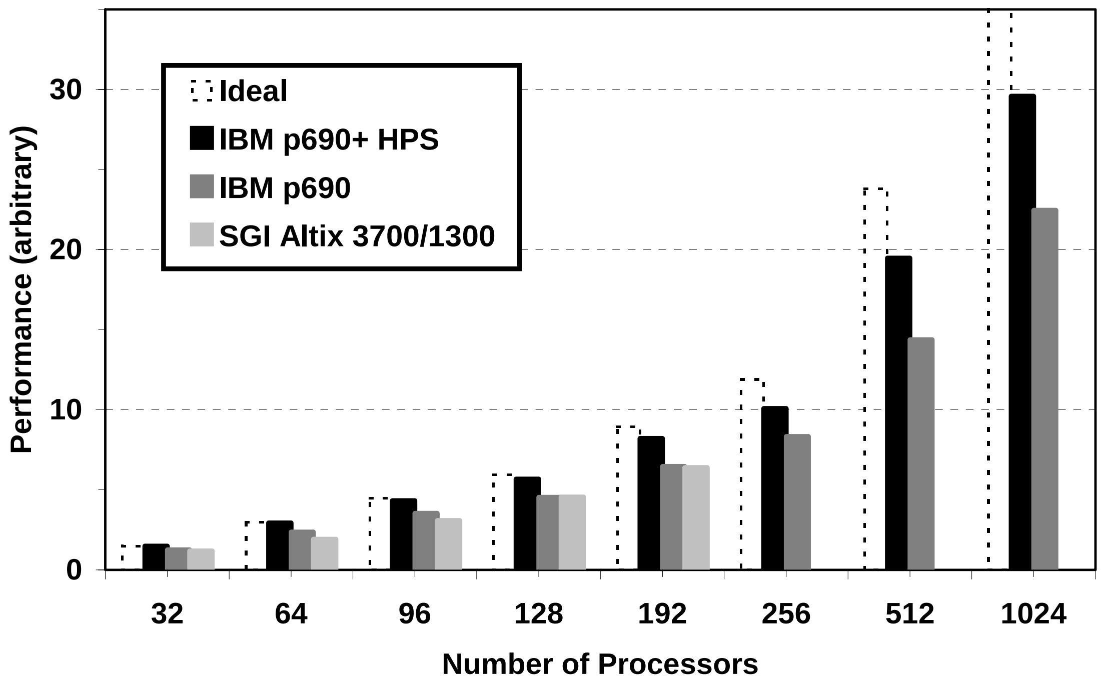

Figure 11. Scalability of CRYSTAL-2003 in calculations on crystalline Crambin on the IBM p690 and p690+ and the SGI Altix 3700/1300 systems. The dashed bars labelled 'Ideal' assume perfect scaling from the IBM p690+ 32-processor timing.

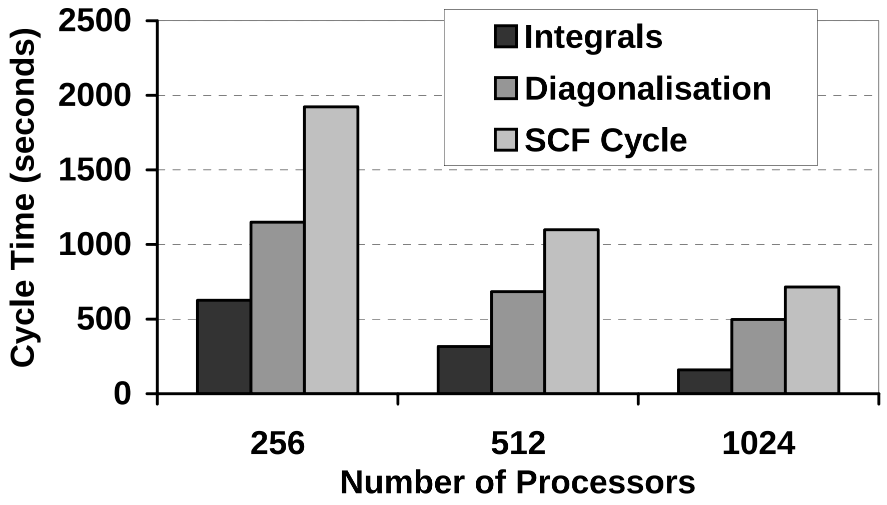

Figure 12. Breakdown of times for the first SCF cycle of CRYSTAL on the IBM p690 for 6-31G** calculations on Crystalline Crambin.

The rationalization of the memory management within the code has continued. All large arrays are now dynamically allocated, and further general purpose Fortran 90 modules are available for more complex data structures, such as linked lists. On MPP systems disk access is often an expensive process, and so as far as is possible this is now avoided. Data is either recalculated or, for distributed objects, stored in memory, the latter being possible because of the memory capacity typically found on these machines.

Performance (proportional to inverse time) of CRYSTAL 2003 for a benchmark calculation on crystalline Crambin [15] with 1284 atoms per cell is reported in Figure 11 for the IBM p690, p690+ and SGI Altix 3700/1300. The structure of Crambin is derived from XRD data at 0.52 Å. These timings were performed with the 6-31G basis set (7,194 GTOs) and refer to three cycles of the iterative SCF process. The Altix runs almost neck-and-neck out to 192 processors with the IBM p690 (which has an identical clock speed). Performance on the IBM p690+ exceeds both systems by a ratio that starts at around 1.20 and rises to 1.32 at 1024 processors. The clock speed ratio is 1.31.

Additional timings (not shown) for the larger, more accurate, 6-31G** basis set with 12,354 GTOs, have shown that the integrals scale almost perfectly, with the diagonalisation around 3.1 times quicker on 1024 compared to 256 CPUs. Figure 12 provides a breakdown of the cycle time as a function of processor count.

6. MOLECULAR SIMULATION

6.1. DL_POLY and NAMD

DL-POLY: DL_POLY [16] 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. Two graphical user interfaces are available, one based on the CERIUS^2 Visualiser from Accelrys and the other on the Java programming language.

DL_POLY supports a wide range of application areas, including [17] ionic solids, solutions, metals, zeolites, surfaces and interfaces, complex systems (e.g. liquid crystals), minerals, bio-systems, and those in spectroscopy. Comprehensive benchmarking of the replicated data (RD) version (Version 2.11) of DL_POLY [18,19] clearly reveals the limitations inherent in the RD strategy, with restrictions in the size of system amenable to study, and limited scalability on current high-end platforms. These limitations apply not only to systems possessing complex molecular topologies and constraint bonds, but also to systems requiring simple atomic descriptions, systems that historically exhibited excellent scaling on the Cray T3E/1200E. Significant enhancements to the code’s capabilities have arisen from the recent release of the distributed data (or domain decomposition) version (DL_POLY 3) [19], developments that have been accelerated in light of the arrival of the HPCx system.

Evaluation of the Coulomb potential and forces in DL_POLY is performed using the smooth particle mesh Ewald (SPME) algorithm [20]. As in all Ewald [21] methods, this splits the calculation into two parts, one performed in real space and one in Fourier space. The former only requires evaluation of short ranged functions, which fits in well with the domain decomposition used by DL_POLY 3, and so scales well with increasing processor count. However the Fourier component requires 3 dimensional FFTs to be performed. These are global operations and so a different strategy is required if good scaling is to be achieved.

The original implementation involved replicating the whole FFT grid on all processors and performing the FFTs in serial after which each processor could evaluate the appropriate terms for the atoms that it held. This method clearly has a number of well-known drawbacks.

While both open source 3D parallel FFTs (such as FFTW [22]) and proprietary routines (such as Cray's PCCFFT) are available, neither adequately address all the issues. The problem is that they impose a data distribution, typically planes of points, that is incompatible with DL_POLY's spatial domain decomposition, so while a complete replication of the data is not required, it is still necessary to perform extensive data redistribution which will limit the scaling of the method.

To address these limitations, a parallel 3D FFT has been written [23] which maps directly onto DL_POLY's data distribution; this involved parallelizing the individual 1D FFTs in an efficient manner. While the method will be slower than the proprietary routines for small processor counts, at large numbers it is attractive, since

(a) while moving more data in total, the method requires much fewer messages, so that in the latency dominated regime it should perform better, and (b) global operations, such as the *all to all* operations used in both FFTW and PCCFFT, are totally avoided. More generally the method is extremely flexible, allowing a much more general data distribution than those of other FFTs, and as such should be useful in other codes which do not map directly onto a "by planes" distribution.

In the present section we present recent results obtained on the IBM and SGI Altix systems, the Compaq AlphaServer ES45/1000 and the SGI Origin 3800/R14k-500, results which highlight the drastic improvements in both system size and performance made possible through these developments. The four benchmarks reported in Table 2 include two Coulombic-based simulations of NaCl, one with 27,000 ions, the second with 216,000 ions. Both simulations involve use of the Particle Mesh Ewald Scheme, with the associated FFT treated by the algorithm outlined above [23] in which the traditional all-to-all communications are replaced by the scheme that relies on column-wise communications only. The reported timings are for 500 time steps in the smaller calculation, and 200 time steps in the larger simulation. The other two benchmarks are macromolecular simulations based on Gramicidin-A; the first includes a total of 99,120 atoms and 100 time steps. The second, much larger simulation, is for a system of eight Gramicidin-A species (792,960 atoms), with the timings reported for just 50 time steps. In terms of time to solution, we see that the AlphaServer SC outperforms the Origin 3800 at all processor counts in all four benchmarks; both 256 CPU runs for the larger NaCl and Gramicidin-A simulations suggest factors of 1.3-1.4.

Table 2. Time in Wall Clock Seconds for four DL_POLY 3 benchmark Calculations on the SGI Origin 3800, Compaq AlphaServer SC ES45/1000, IBM p690 and p690+, SGI Altix 3700/1300 and SGI Altix 3700/1500.

CPUs	SGI Origin 3800 / R14k-500	Compaq Alpha ES45 / 1000	IBM p690	IBM p690+	SGI Altix 3700 / 1300	SGI Altix 3700 / 1500
NaCl ; 27,000 ions, 500 time steps						
16	313	183	160		95	82
32	168	103	97		54	44
64	92	57	61		32	29
128	53	37	42			
256	36	24				
NaCl; 216,000 ions, 200 time steps						
16	764	546	455		282	241
32	387	273	234	146	156	128
64	201	143	128	78	80	69
128	116	85	78	47	56	43
256	71	60	48	31		
512			41			
Gramicidin A; 99,120 atoms, 100 time steps						
16	282	173	166		100	84
32	167	109	107		62	53
64	100	75	74		46	36
Gramicidin A; 792,960 atoms, 50 time steps						
32	654	370	311	203	208	169
64	273	186	196	117	122	91
128	140	109	139	77	89	55
256	97	68	103	56		
512						

These results show a marked improvement in performance compared to the replicated data version of the code [18], with the gratifying characteristic of enhanced scalability with increasing size of simulation, both in the ionic and macromolecular simulations. Considering the NaCl simulations, we find speedups of 139 and 122 respectively on 256 processors of the Origin 3800 and AlphaServer SC in the 27,000-ion simulation. These figures increase to 172 and 171 respectively in the larger simulation featuring 216,000 ions. Considering performance of the latter on the more recent IBM and SGI systems, we find 128 processor speedups of 81 and 90 on the SGI Altix 3700/1300 and /1500 respectively, and 93 and 99 on the IBM p690 and p690+. The p690+ performance is seen to be intermediate between that measured on the two SGI Altix systems. A speedup of 152 is found on 256 processors of both the p690 and p690+.

For these systems the FFT routine remains the poorest scaling; the time required to exchange scales poorly on the IBM p690 with increased processor count. Doubling the number of processors roughly halves the amount of data a processor has to send, so one would expect the time to roughly halve. In practice this is not the case, and at certain processor counts the time dramatically increases. This can be traced to the message passing occurring via the switch rather than through shared memory. A more detailed timing breakdown than presented here reveals that for the x direction, which never has to use the switch in the present case, the scaling is reasonable, while for the z direction, which has to use the switch for the first time at 16 processors, there is a major spike in the timings.

A more compelling improvement with system size is found in the macromolecular Gramicidin-A simulations. Again the SPME algorithm is used for evaluation of the Coulomb field, but in these simulations there is the extra complication of constraints on the atoms' motions, which reflects chemical bonds in the system. The shake algorithm is used to evaluate the constraints, and this is again potentially a global operation and so, as for the FFT, good scaling is difficult to achieve. In the distributed data implementation, both SHAKE and short-range forces require only nearest neighbour communications, suggesting that communications should scale linearly with the number of nodes, in marked contrast to the replicated data implementation. This is borne out in practice. In the larger simulation (with 792,960 atoms) we find speedups of 218 and 175 on 256 processors of the Origin 3800 and AlphaServer SC respectively. These speedups are significantly reduced on the more recent hardware, with the corresponding figures of 97 and 116 on the IBM p690 and p690+ respectively. This level of scalability still represents a significant advance over that exhibited by both DL_POLY 2 and CHARMM [24]. The p690+ performance is again seen to be intermediate between that measured on the two SGI Altix systems, with the SGI Altix 3700/1500 exhibiting far better speedup than the 3700/1300.

NAMD is the parallel, object-oriented molecular dynamics program designed for high performance simulations of large biomolecular systems [25]. NAMD employs the prioritized message-driven execution capabilities of the Charm++/Converse parallel runtime system, allowing excellent parallel scaling on both massively parallel supercomputers and commodity-based workstation clusters. Rick Kufrin (NCSA) carried out an initial implementation of the code on HPCx. Figure 13 shows performance based on the elapsed time per time step for the standard NAMD F_1-ATPase benchmark [26], a system comprising 327,000 atoms. This shows that up to 224 processors the Altix outperforms the IBM systems, by a factor of 1.3-1.5 in the case of the faster IBM p690+. However, whereas the scaling on the IBM systems is excellent right out to 1024 processors, there are signs that the scalability on the Altix is declining. That the scaling performance is almost identical on the p690 SP system and the p690+ HPS system, indicates that communications overheads are not playing a large part and that some other factor is causing the performance on the Altix to turn over. The speed-up of 660 on 1024 processors achieved on the p690+ system compares favourably with that reported on the Compaq AlphaServer ES45/1000 TCS-1 system at Pittsburgh (e.g. a speedup of 778 on 1024 CPUs for the F_1-ATPase simulation).

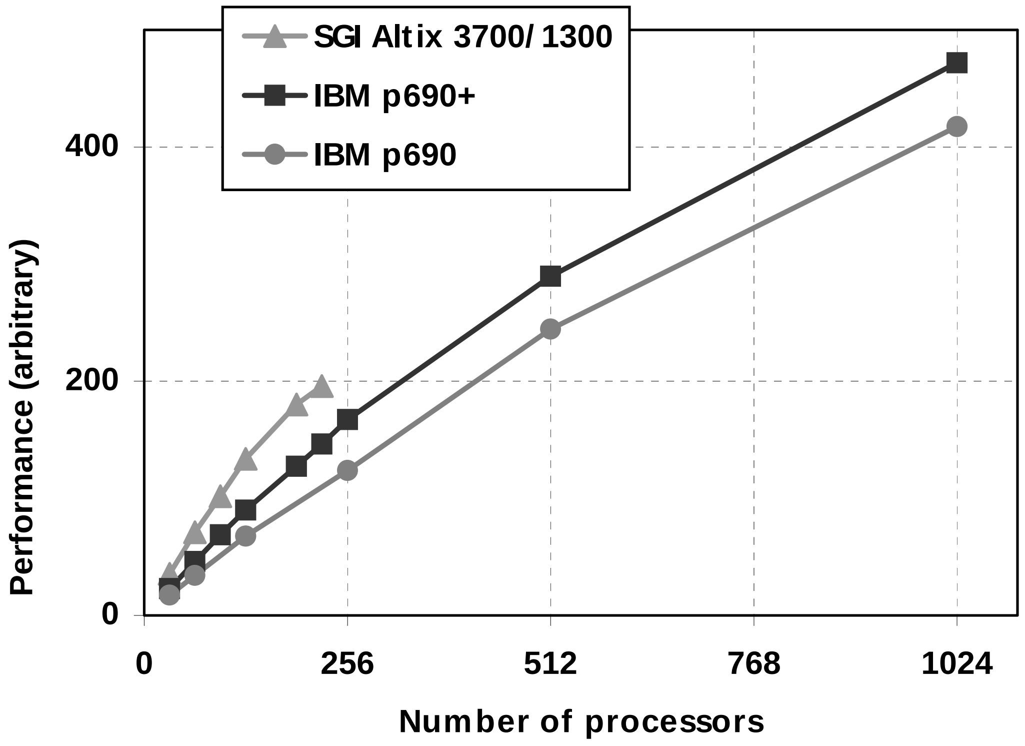

Figure 13. Performance of the NAMD F_1-ATPase benchmark on the IBM p690 and p690+ systems and the SGI Altix 3700/1300.

7. MOLECULAR ELECTRONIC STRUCTURE

7.1. GAMESS-UK

GAMESS-UK [27] represents a typical established electronic structure code, comprising some 800K lines of Fortran that permits a wide range of computational methodology to be applied to molecular systems. Improving the scalability of the parallel code has been addressed in part by adopting a number of the tools developed by the High Performance Computational Chemistry (HPCC) group from the Environmental Molecular Sciences Laboratory at PNNL, in Richland, Washington. These include the Global Array (GA) toolkit [3] that provides an efficient and portable "shared-memory" programming interface for distributed-memory computers, and the scalable, fully parallel eigensolver, PeIGS whose numerical properties satisfy the needs of the chemistry applications [28,29].

The main source of parallelism in the DFT module is the computation of the one- and two-electron integrals together with the exchange correlation contributions, and their summation into the Fock matrix. The computation of these quantities is allocated dynamically using a shared global counter. With the capabilities afforded by the GA tools [3], some distribution of the linear algebra becomes trivial. As an example, the SCF convergence acceleration algorithm (DIIS - direct inversion in the iterative subspace) is distributed using GA storage for all matrices, and parallel matrix multiply and dot-product functions. This not only reduces the time to perform the step, but the use of distributed memory storage (instead of disk) reduces the need for I/O during the SCF process. Diagonalisation of the resulting Fock matrix is now based on the PeIGS module from NWChem [30].

Substantial modifications were required to enable the SCF and DFT 2nd derivatives [31] to be computed in parallel. The conventional integral transformation step has been omitted, with the SCF step performed in direct fashion and the MO integrals, generated by re-computation of the AO integrals, and stored in the global memory of the parallel machine. The GA tools manage this storage and subsequent access. The basic principle by which the subsequent steps are parallelised involves each node computing a contribution to the current term from MO integrals resident on that node. For some steps, however, more substantial changes to the algorithms are required. The coupled Hartree-Fock (CPHF) step and construction of perturbed Fock matrices are again parallelised according to the distribution of the MO integrals. The most costly step in the serial 2nd derivative algorithm is the computation of the 2nd derivative two-electron integrals. This step is trivially parallelised through a similar approach to that adopted in the direct SCF scheme - using dynamic load balancing based on a shared global counter. In contrast to the serial code, the construction of the perturbed Fock matrices dominates the parallel computation. It seems almost certain that these matrices would be more efficiently computed in the AO basis, rather than from the MO integrals as in the current implementation, thus enabling more effective use of sparsity when dealing with systems comprising more than 25 atoms.

Table 3. Time in Wall Clock Seconds for Four GAMESS-UK Benchmark Calculations on the Compaq AlphaServer SC ES45/1000, SGI Origin 3800/R14k-500, IBM p690 and p690+, and SGI Altix 3700/1300 and 3700/1500.

CPUs	SGI Origin 3800 / R14k-500	Compaq Alpha ES45 / 1000	IBM p690	IBM p690+	SGI Altix 3700 / 1300	SGI Altix 3700 / 1500
Cyclosporin (1000 GTOs) DFT/B3LYP 6-31G						
32	1049	580	556	385	438	379
64	587	355	369	254		221
128		269				
Cyclosporin (1855 GTOs) DFT/B3LYP 6-31G**						
32	3846	2084	2047	1366		1383
64	2271	1276	1429	876		808
128	1585	896	1193	739		615
Valinomycin (882 GTOs) DFT/HCTH						
32	1947	1024	1099	802	907	780
64	1065	577	635	463		420
128	623	369	489	309		263
Valinomycin (882 GTOs) DFT/HCTH (J-fit)						
32	649	429	473	269	267	237
64	425	283	370	243		
128	352		355			
Valinomycin (1620 GTOs) DFT/HCTH						
32	9636	4738	4723	3513		3595
64	5110	2556	2690	1966		1901
128	3024	1557	1777	1259		1190
256	2026	1013	1343	985		
$(C_6H_4(CF_3))_2$ 2nd Derivatives DFT/HCTH						
16			1416			887
32	1772	922	842	457	565	497
64	1081	557	471	362		338
128	826	337	294	240		326

The performance of the DFT and SCF 2nd Derivative modules on the SGI O3800/R14k-500, Compaq AlphaServer SC ES45/1000 and IBM p690 and p690+ and SGI Altix systems are shown in Table 3. Note that the DFT calculations did not exploit CD fitting, but evaluated the coulomb matrix explicitly. Considering the DFT results, modest speedups ranging from 55 (IBM p690) to 78 (SGI Origin 3800) are obtained on 128 processors for the larger cyclosporin calculation. Somewhat better scalability is found in both Valinomycin

DFT calculations where a greater proportion of time is spent in integral evaluation arising from the more extended basis sets [32]. Speedups of 87 and 89 are obtained on 128 processors of the IBM p690 and p690+, figures that improve somewhat on the SGI Altix 3700/1500 and AlphaServer (97) and SGI Origin 3800 (102) in the 1620 GTO calculation. Note that the performance of both GAMESS-UK and NWChem using the current release of the Global Array Tools (V3.3) show a marked deterioration on SGI Altix systems that involve partitioning across two or more linux64 partitions. This is a common occurrence given the 64-way partitions characterising the CSAR system, so that discussion of the Altix times of Table 3 are limited to those on the 256-way single system image characterising the "ram" system at ORNL. The optimum 128 CPU performance in the DFT calculations is indeed delivered by the SGI Altix 3700/1500, which outperforms the IBM p690+ by ca. 1.15-1.20. The enhanced performance of the DFT 2nd Derivative module on the IBM p690 and p690+ arises from the decreased dependency on latency exhibited by the current implementation compared to the DFT module. Thus the timings of Table 3 suggest that, at least for the 2nd derivatives module, the IBM p690+ is outperforming the SGI Altix 3700/1500 on 128 processors.

The less than impressive scalability of both GAMESS-UK (and also NWChem) on the IBM p690 and p690+ systems arises to some extent from the dependency of both codes on a Global Array implementation that is based on IBM's LAPI communication library [33]. The current implementation of LAPI on POWER4-based architectures is far from optimal, with the measured latencies and bandwidths significantly inferior to those measured on corresponding POWER3-based systems. We are currently working with PNNL and IBM's LAPI team to further understand and address these shortcomings.

7.2. Parallel Eigensolver Performance

Many scientific MPP applications under development involve the computation of the standard real symmetric eigensystem problem. In particular, this diagonalization stage in parallel quantum chemistry codes often consumes a significant percentage of overall run time. Several public domain parallel eigensolvers are available for general use including ScaLAPACK library routines [34] and Parallel Eigensolver System software (PeIGS) library routines (from PNNL, [28,29]).

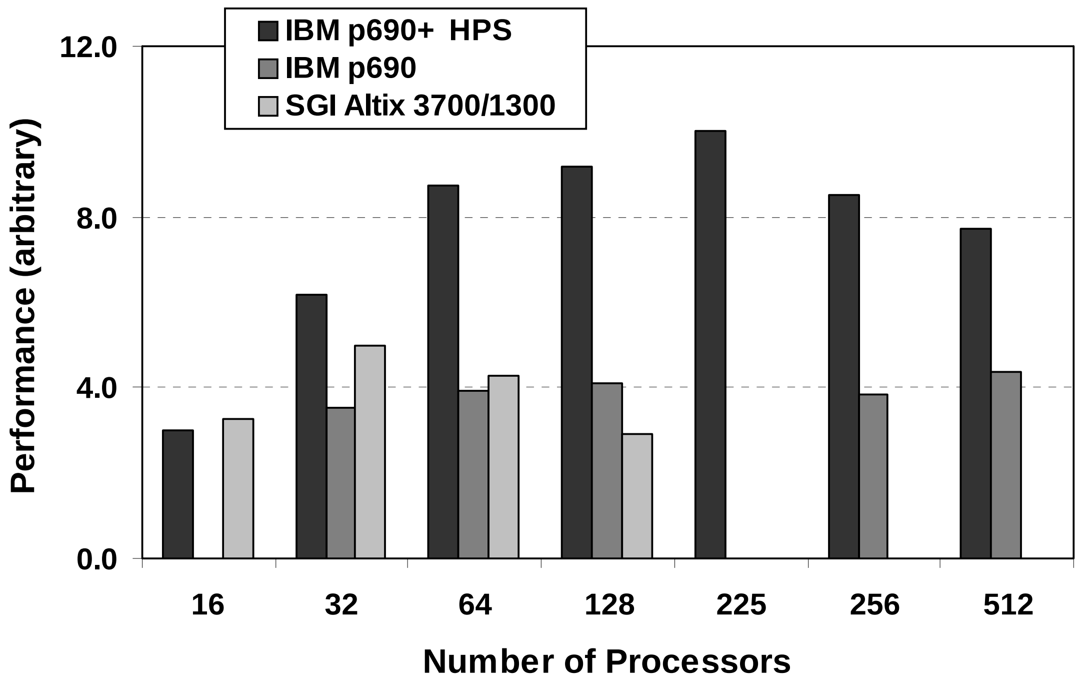

Figure 14. Performance (inverse time) for the ScaLAPACK divide-and-conquer routine (PDSYEVD) on the IBM p690 and p690+ and SGI Altix 3700/1300 for a matrix dimension of 3888.

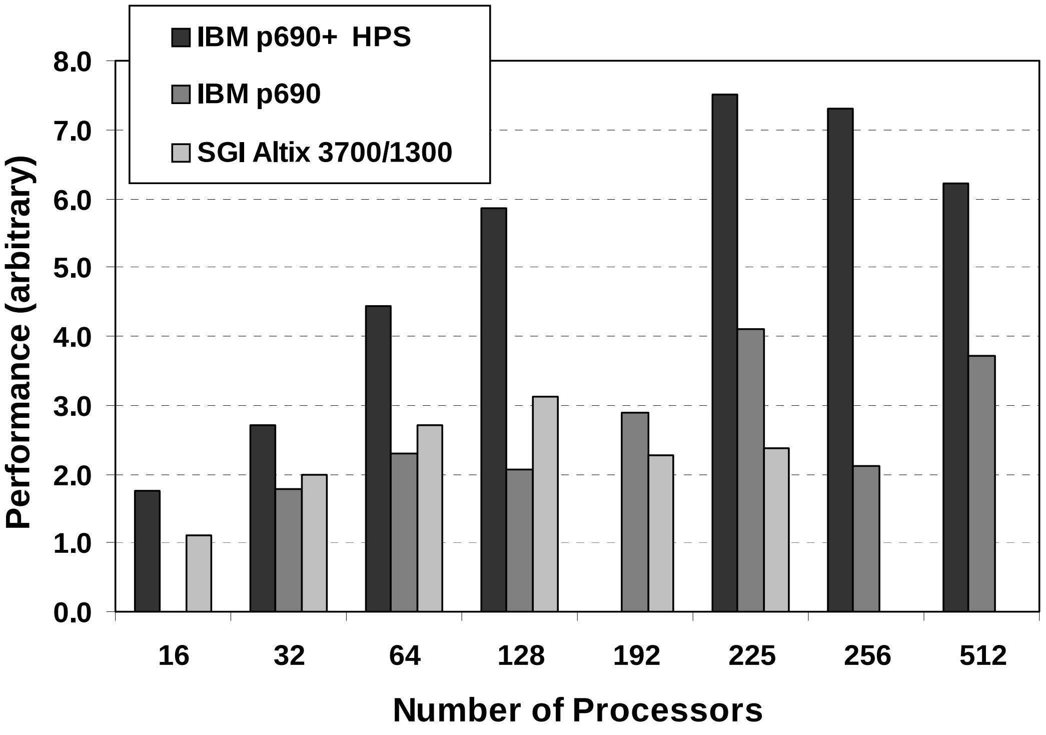

Figure 15. Performance (inverse time) for the ScaLAPACK divide-and-conquer routine (PDSYEVD) on the IBM p690 and p690+ and SGI Altix 3700/1300 for a matrix dimension of 7194.

The latest release of ScaLAPACK (v1.7) includes new parallel algorithms based on divide-and-conquer techniques [35]. Also, algorithms developed recently by Dhillon et al [36] are included in a new release of PeIGS (v3.0) [29]. In addition, Ian Bush at Daresbury Laboratory has developed BFG [14], a parallel one-sided Block Factored Jacobi implementation [28] based upon the techniques described by Littlefield et al [29].

The fastest serial diagonalisation algorithm is that based upon Householder reduction to tridiagonal form followed by diagonalization [37]. A number of parallel implementations of this algorithm exist, e.g. ScaLAPACK and the PeIGS package, but they all suffer from a number of drawbacks. The most obvious is that even for quite large matrices the scaling is less than impressive at large processor counts (see below).

However on closer investigation further problems can be seen. Both ScaLAPACK and PeIGS require a rather indeterminate amount of workspace to operate, and this workspace is replicated across all the processors; in ScaLAPACK in the worst case scenario the size of this workspace can be the size of the whole matrix. Obviously for the very large matrices that one wishes to diagonalize on high-end systems this is highly undesirable, and may even be impossible.

Further many algorithms can easily generate a good guess at the eigenvectors, and it would be very useful if this guess could be used to speed up the diagonalization. Unfortunately this is difficult within the Householder methodology; an alternative here is Jacobi's algorithm. Although slower in serial, typically by a factor of three to five, it has a number of attractive properties for practical parallel codes; the scaling is good (see below), the memory requirement scales strictly with the inverse of the number of processors and is known a priori, and guesses at the eigenvectors, provided they are mutually orthogonal, can be used very effectively to speed up the diagonalization. The DL-based BFG code implements a version of this algorithm as described by Littlefield and Maschoff [28], with the current version offering extended functionality, better numerical stability, increased single processor performance and better scaling than its predecessors. It can diagonalize both real symmetric and Hermitian matrices, and has interfaces for matrices distributed either in a block-cyclic (like ScaLAPACK) fashion, or by columns (PeIGS). It is written in standard conforming Fortran and uses MPI for message passing, and so it is portable to a wide range of parallel machines. Further extensive use of

MPI communicators make it flexible enough that matrices may be diagonalized across just a subset of the processors, thus allowing a number of diagonalizations to be carried out at once.

We have shown elsewhere [38] the impressive times to solution that may be obtained from obtained from the divide-and-conquer routines. We have used example fully symmetric Fock matrices from a CRYSTAL simulation, representing lithium fluoride with an F centre, to measure the performance of PDSYEVD, the ScaLAPACK Parallel Divide and Conquer routine [36]. Execution times are shown in Figure 14 for a matrix of dimension 3888 and in Figure 15 for a matrix of dimension 7194 on the IBM p690+ and SGI Altix 3700/1300 systems. For the smaller matrix size the Altix starts out faster on 16 processors but the IBM system scales better. On both machines the algorithm is running out of scalability above 64 processors. As expected the larger system shows scalability out to larger numbers if processors. The Altix scales better to 128 processors but its performance falls off giving a relative advantage to the IBM at 192 and 225 processors. This may be due to the performance implications of running over multiple 64-way single system images on the CSAR system

8. COMPUTATIONAL ENGINEERING

8.1. PCHAN

Fluid flows encountered in real applications are invariably turbulent. There is, therefore, an ever-increasing need to understand turbulence and, more importantly, to be able to model turbulent flows with improved predictive capabilities. As computing technology continues to improve, it is becoming more feasible to solve the governing equations of motion – the Navier-Stokes equations – from first principles. The direct solution of the equations of motion for a fluid, however, 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.

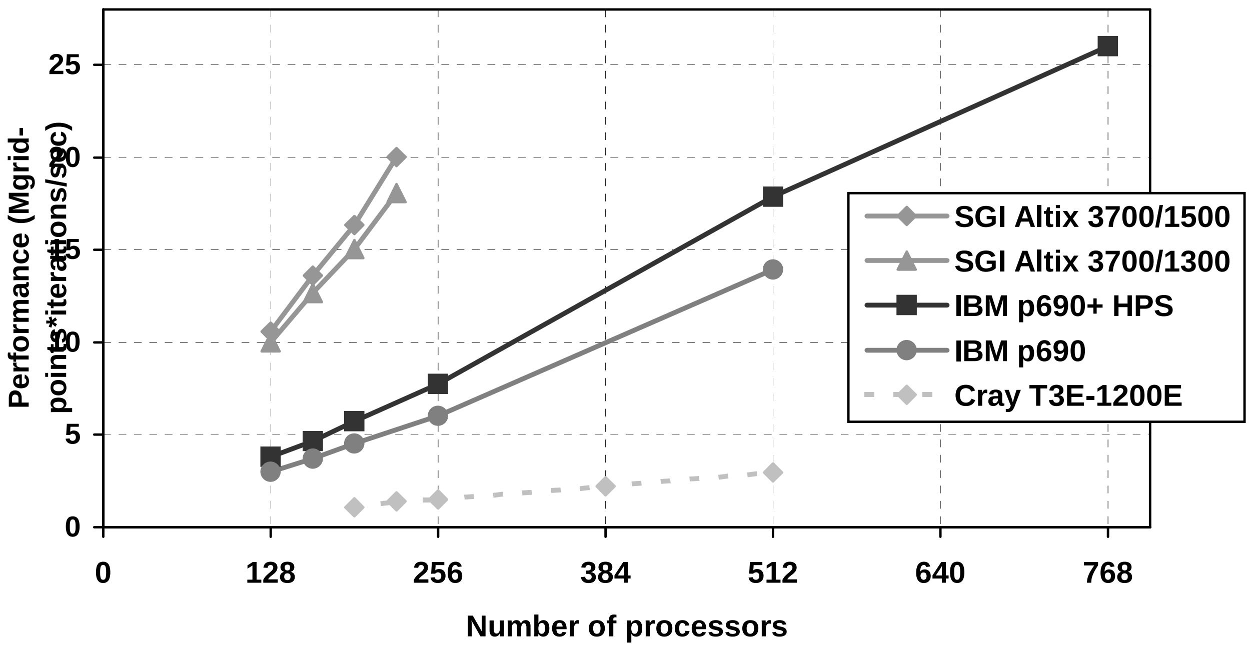

Figure 16. The PCHAN T3 (360x360x360) benchmark on the IBM p690 and p690+, the SGI Altix 3700/1300 and 3700/1500, and the Cray T3E/1200E systems.

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 [39]. 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. Performance with the T3 Grid Benchmark data case (360x360x360) shows close to ideal scaling on both the Cray T3E/1200E and on the IBM p690 systems, for which we have data from both the Cheetah system (at ORNL) and from the HPCx system. Where benchmark runs allow direct processor-for-processor comparison between the IBM and Cray systems (192-512 processors), the performance ratio is fairly constant at around 6.2 to 7.2. e.g., the 512 CPU T3 Benchmark required 227 elapsed seconds, with the HPCx system outperforming the Cray T3E/1200E (1,575 seconds) by a factor of 6.94. Figure 16 shows performance results from the two IBM systems, the two SGI Altix systems and the Cray T3E.

The most important communications structure within PCHAN is a halo-exchange between adjacent computational 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. We see almost linear scaling from all systems and in the case of the p690+ all the way out to 768 processors. Hardware profiling studies of this code have shown that its performance is highly dependent on the cache utilisation and bandwidth to main memory. With a performance ratio of around 2.6 between the Altix systems and the p690+ it is clear that the memory management for this code is taking place much more efficiently on the Altix. Whether this is in the form of better compiler optimisations or better utilisation of the cache at the hardware level remains unclear.

8.2. THOR

THOR [40] is a CFD package, which can be used to compute laminar and turbulent flows in complex geometries, with or without chemical reactions. The finite volume approach is used to discretise the governing equations expressed in body-fitted curvilinear coordinates. The multiblock strategy is adopted to deal with complex geometries and the code implements first and second order turbulence models, including k-epsilon, k-omega, LRR and SSG. Different discretisation schemes, including first order upwind and higher order QUICK, SMART and CUBISTA, are implemented. Furthermore implicit first- and second-order time-dependent schemes were added to THOR to model unsteady flows, such as oscillating turbulent flames, large scale explosions and blood flows in veins.

THOR is parallelised with MPI. Domain decomposition is employed and by using the multiblock structure in the code, the computational domain was decomposed to multiblocks according to boundary conditions and processor number. Different blocks can then be allocated to different processors. One block must be within one processor but to enhance the efficiency, one processor can have more than one block.

The benchmark run here has 2619156 nodes and calculates 50 iterations towards convergence of a turbulent flow using the k-epsilon turbulence model. Performance of THOR on 32 and 64 processors is shown in Figure 17. Like PCHAN, THOR contains nearest neighbour halo-type communications in order to update boundary data between the partitions of the computational domain. It also has high memory access requirements although whereas in PCHAN the access is sequential, the finite volume data structures involve a more randomised sequence of accesses. Scaling from 32 to 64 processors on the IBM systems is poor. Both Altix 3700 systems outperform the IBMs, e.g. by a ratio of 1.6 over the p690+ on 64 processors.

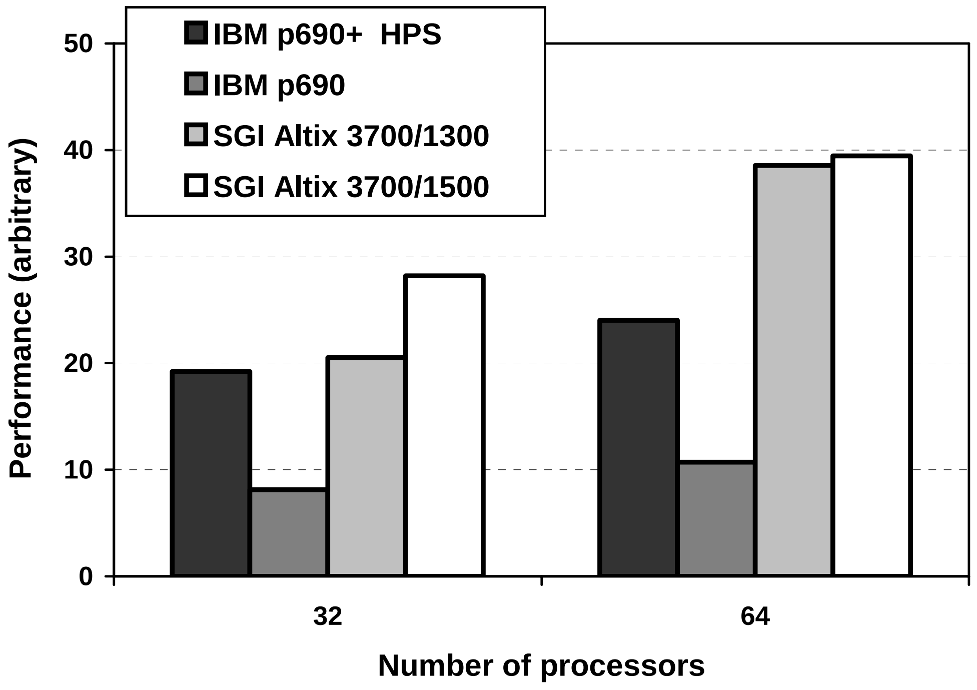

Figure 17. Performance of the THOR code on the IBM p690 and p690+ and the SGI Altix 3700/1300 and 3700/1500 systems.

9. ENVIRONMENTAL SCIENCE

9.1. POLCOMS

The Proudman Oceanographic Laboratory Coastal Ocean Modeling System (POLCOMS) has been developed to tackle multi-disciplinary studies in coastal/shelf environments [41]. 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) [42]. 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 18 as the amount of work (gridpoints × timesteps) divided by the time.

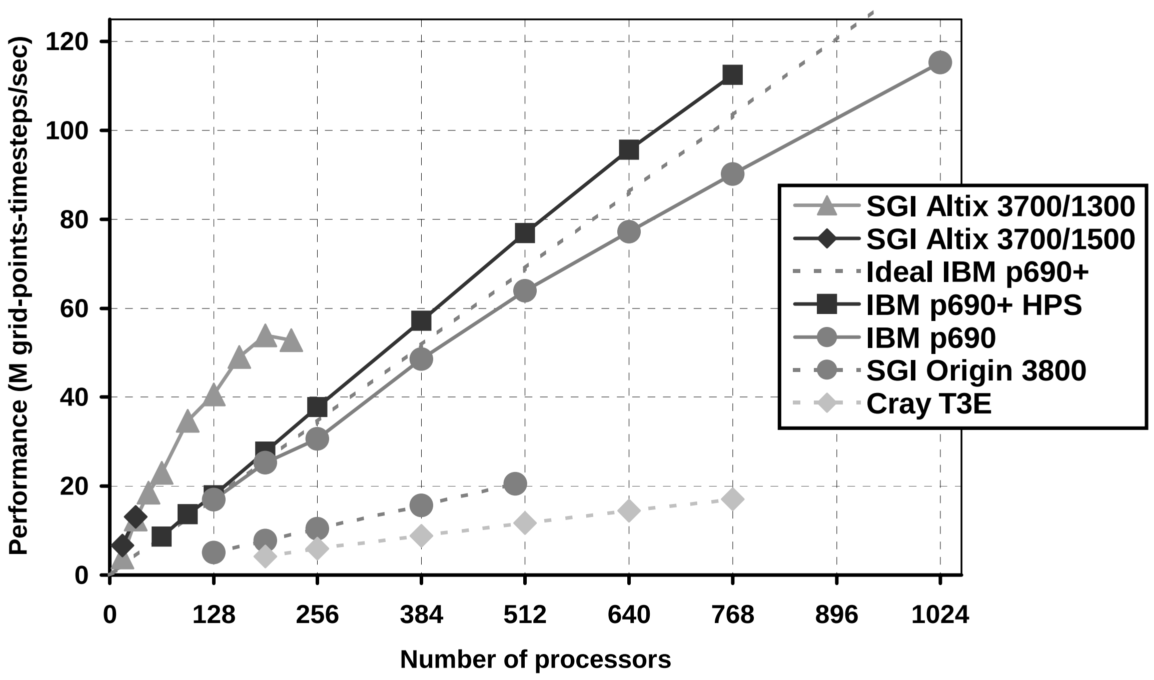

Figure 18. Performance of the POLCOMS code running a 2km simulation (1200x1200x34 gridpoints) on the IBM p690 and p690+, SGI Origin 3800, Altix 3700/1300 and Altix 3700/1500 and the Cray T3E/1200E systems. The dashed line labelled 'Ideal' indicates perfect scaling from the p690+ 64 processor data point.

The perfect linear scaling found on the p690+ HPS as compared to the p690 SP Switch indicates the extent to which communications limits the performance. As with PCHAN, the absolute performance is highly dependent on cache and memory issues and the SGI Altix systems significantly outperform the IBM systems. However, as seen for the NAMD code, the scaling on the Altix 3700/1300 is starting to run out above 128 processors.

10. SUMMARY

The key goal behind our work is to enable complex and increasingly coupled and multi-disciplinary scientific applications to harness the potential performance of current and future high-performance parallel systems in order to meet critical scientific objectives. We have identified from the outset three key issues: management of the memory hierarchy, expression and management of concurrency and efficient sequential execution. These are key issues on all current systems. We find that applications which scale best across all systems (CPMD, CRYSTAL, PCHAN and POLCOMS) are those where the developers have most successfully optimised the global data management for multi-level cache architectures and maintained a high compute to communications ratio. There is no substitute for careful design of data structures, data access (loop indexing) and data distribution. For legacy codes this may require a complete re-design.

We have presented the results from a benchmarking programme that compares the performance of those systems that characterise the UK's flagship scientific computing services, the IBM p690+ cluster, with 1.7 GHz POWER4+ processors, operated by HPCx and the SGI Altix 3700, with 1.3 GHz Itanium 2 processors at CSAR. A total of ten applications have been considered from a diverse section of disciplines; materials science, molecular simulation and molecular electronic structure, computational engineering and environmental science.

We find a wide range of performance with some applications performing better on one system and some on the other. By way of summary, we show in Figure 19 the relative performance of a 128-CPU partition (64-CPU in the case of THOR) of the IBM p690+ delivered by the SGI Altix 3700/1300 (i.e. $T_{128\text{-CPU}}$IBM p690+/ $T_{128\text{-CPU}}$SGI Altix3700/1300), for fourteen of the examples presented in the preceding sections involving nine of the ten applications considered.

Four of the nine applications considered are seen to perform significantly better on the SGI Altix 3700 than on the IBM p690+, namely the engineering codes (PCHAN and THOR), the environmental code POLCOMS, and NAMD, the molecular simulation code. There can be little doubt that this is a consequence of the superior memory architecture of the Altix, at least in the case of the engineering and environmental codes, for these codes are known to be extremely demanding on memory bandwidth. In the case of NAMD, the communication demands of the simulation code are undoubtedly exposing the current limitations in Release 1 of the HPS microcode.

In contrast, the electronic structure codes, CASTEP, CPMD and CRYSTAL, and to a lesser extent DLPOLY, show superior performance on the IBM p690+, as indeed do the PDSYEVD-based matrix diagonalisation benchmarks. It is no coincidence that all four application codes that exhibit enhanced performance on the p690+ have been the subject of considerable optimisation efforts by the HPCx Terascaling Team and their collaborators. This work has been outlined in the preceding sections, notably;

- Substantial optimization of CASTEP, by ensuring that the communications involved in the data redistribution (the MPI_AllToAllV calls surrounding the FFTs) are “SMP-aware” [8].
- The mixed-mode, hybrid-programming model used in the p690+ implementation of CPMD [10]
- The incorporation of a somewhat faster, and more numerically stable version of the parallel Jacobi diagonalizer [14] in CRYSTAL 2003, and the rationalization of the memory management within the code
- The introduction of a parallel 3D FFT [23] which maps directly onto DL_POLY's data distribution, requiring far fewer messages, so that in the latency dominated regime it should perform better, and all-to-all operations, which are used, for example, in FFTW and PCFFT, are totally avoided.

AIMPRO is found to exhibit similar performance on both systems. The benchmarks combine in about equal weight eigensolves, which perform better on the IBM p690+ with matrix-matrix multiplications, which perform better on the Altix.

It is the applications with unavoidable large-scale global communications (from this paper DL-POLY, GAMESS, and NAMD, but also H2MOL and PFARM [38]) which provide the most severe test for the communications sub-systems of these machines. With processor speeds continuing to increase faster than interconnect speeds, scalability is not likely to improve for algorithms which depend on collective, global operations.

The benchmark results obtained in this work and the performance levels achieved have highlighted a wide range of performance, with some algorithms scaling far better than others. Our focus on algorithm development and the drive to remove dependencies on collective, global operations have been successful in several cases in improving the scalability of these codes. Where all these issues have been addressed we find excellent levels of scalability and performance, with the platform-dependence far less critical.

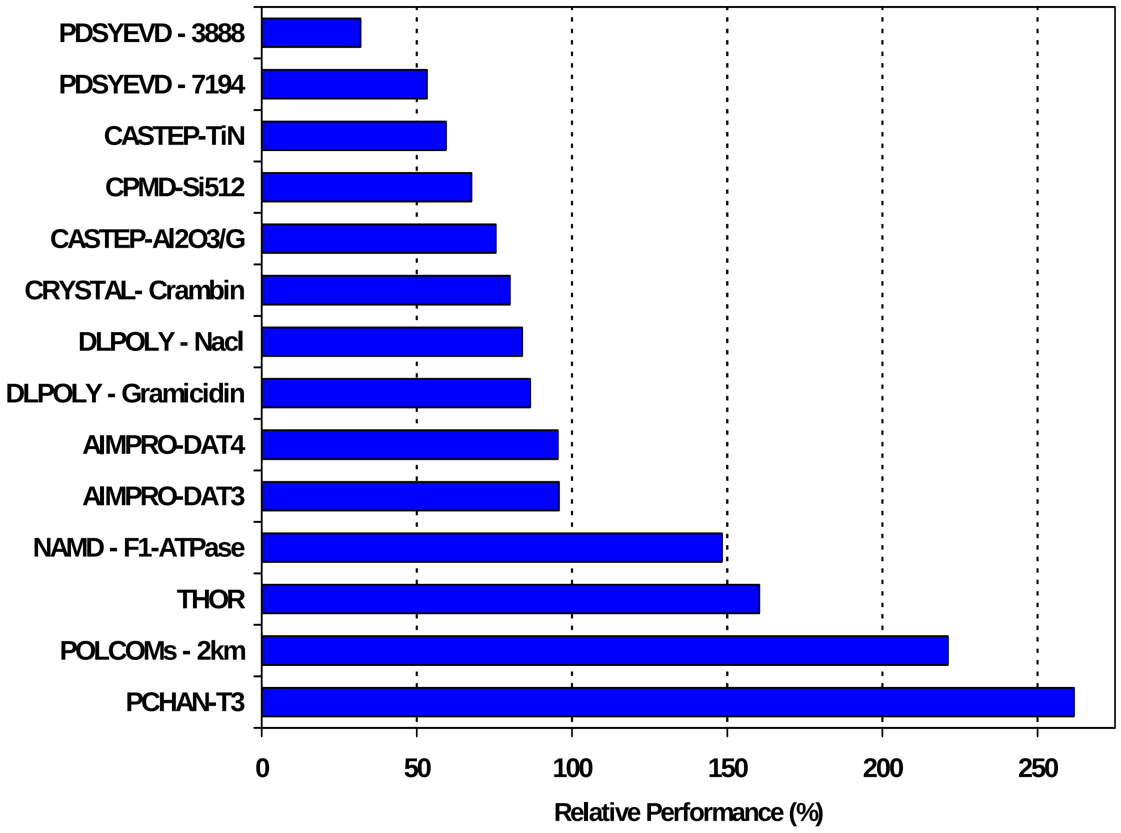

Figure 19. Application Performance: Percentage of a 128-processor partition of the IBM p690+ achieved by 128-processors of the SGI Altix 3700/1300 i.e. $T_{128\text{-CPU}}$IBM p690+/ $T_{128\text{-CPU}}$SGI Altix3700/1300 across a variety of applications and associated data sets. (For THOR the comparison is made at 64 processors)

11. REFERENCES

[1] *Computational Chemistry Applications: Performance on High-End and Commodity-class Computers* M. F. Guest and P. Sherwood, Proceedings of HPCS 2002, Moncton, Canada, 2002; *HPCx: a new resource for UK Computational Science*, M. Ashworth, J.J. Bush, M.F. Guest, M. Plummer, A.G. Sunderland, S.Booth, D.S. Henty, L. Smith and K. Stratford, in Proceedings of the 17th Annual International Symposium on High Performance Computing Systems and Applications, Ed. D. Senechal (NRC Research press, Canada, 2003)

[2] *Promise and Challenge of High-Performance Computing, with Examples from Molecular Modelling*, T.H. Dunning Jr., R.J. Harrison, D. Feller and S.S. Zantheas, Phil. Trans. Soc. A, **360-1795** (2002) 1079-1105.

[3] *Global Arrays: a nonuniform memory access programming model for high-performance computers*, J. Nieplocha, R.J. Harrison and R.J. Littlefield, J. Supercomput. **10** (1996) 197-220.

[4] *Application Performance on HPCx: The Impact of Serial Efficiency*, M. Ashworth, I.J. Bush, M.F. Guest, M. Plummer, A.G. Sunderland, H.J.J. Van Dam, J. Hein and L. Smith, HPCx Technical Report, 2003.

[5] PMB, a comprehensive set of MPI benchmarks written by Pallas, targeted at measuring important MPI functions: point-to-point message-passing, global data movement and computation routines, one-sided communications and file-I/O, Version 2.2.1, see: http://www.pallas.de/pages/pmb.htm

[6] *The structures and properties of tetrafluoromethane, hexafluoroethane, and octafluoropropane using the AIMPRO density functional program*, Zoellner RW, Latham CD, Goss JP, Golden WG, Jones R**,** Briddon PR**,** Journal of Fluorine Chemistry, **121** (2) (2003) 193-199.

[7] *First-principles simulation: ideas, illustrations and the CASTEP code*, M.D. Segall, P.J.D. Lindan, M.J. Probert, C.J. Pickard, P.J. Hasnip, S.J. Clark and M.C. Payne, J. Phys. Condensed Matter **14** (2002) 2117.

[8] *"An LPAR-customized MPI_AllToAllV for the Materials Science code CASTEP"*, Martin Plummer and Keith Refson, HPCx Technical Report, 2004, HPCxTR0401.

[9] R. Car and M. Parrinello, Phys. Rev. Lett. 55, 2471 (1985); D. Marx and J. Hutter, *ab-initio Molecular Dynamics: Theory and Implementation*, in "Modern Methods and Algorithms of Quantum Chemistry", J. Grotendorst (Ed.), NIC Series, Vol. 1, FZ Jülich, Germany, 2000; see also www.fz-juelich.de/nic-series/Volumel.

[10] CPMD V3.8, Copyright IBM Corp. 1990-2003, Copyright MPI für Festkorperforschung Stuttgart 1997-2001. See also http://www.cpmd.org.

[11] *Dual-level Parallelism for* ab initio *Molecular Dynamics: Reaching Teraflop Performance with the CPMD code*, J. Hutter and A. Curioni, IBM Research Report, RZ 3503 (2003).

[12] *CRYSTAL 98 User's Manual*, V.R. Saunders, R. Dovesi, C. Roetti, M. Causa, N.M. Harrison, C.M. Zicovich-Wilson, University of Torino, Torino, 1998. http://www.chimifm.unito.it/teorica/crystal

[13] Ab Initio *Modelling in Solid State Chemistry*, European Summer School, MSSC2002, 8-13 Sept. 2002, Torino, Italy, http://www.chimifm.unito.it/teorica/mssc2002.

[14] *A New Implementation of a Parallel Jacobi Diagonalizer*, I.J. Bush http://www.cse.clrc.ac.uk/arc/bfg.shtml

[15] *Accurate protein crystallography at ultra-high resolution: Valence-electron distribution in crambin* , C. Jelsch, M.M. Teeter, V. Lamzin, V. Pichon-Lesme, B. Blessing, C. Lecomte, PROC.NAT.ACAD.SCI.USA V. 97 3171 2000.

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

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

[18] *Application Performance on High-end and Commodity-class Computers*, M.F. Guest, http://www.ukhec.ac.uk/publications/reports/benchmarking.pdf

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

[20] *A smooth particle mesh Ewald method*, U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee and L.G. Pedersen. J. Chem. Phys. **103** (1995) 8577.

[21] *Computer Simulation of Liquids*, M.P. Allen and D.J. Tildesley, Clarenden Oxford 1987.

[22] *The Fastest Fourier Transform in the West*, M. Frigo, S.G. Johnson, MIT Technical Report, MIT-LCS-TR-728, Sept. 11, 1997. http://www.fftw.org/

[23] *A Parallel Implementation of SPME for DL_POLY 3*, I.J. Bush and W. Smith, http://www.cse.clrc.ac.uk/arc/fft.shtml

[24] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan and M. Karplus, J. Comp. Chem., 4(2), 1983, 187-217.

[25] *NAMD2: Greater scalability for parallel molecular dynamics*, L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. J. Comp. Phys., **151** (1999) 283-312.

[26] NAMD Performance, Theoretical and Computational Biophysics Group, NIH Resource for Macromolecular Modeling and Bioinformatics, http://www.ks.uiuc.edu/Research/namd/performance.html

[27] GAMESS-UK is a package of ab initio programs written by M.F. Guest, J.H. van Lenthe, J. Kendrick, K. Schoeffel and P. Sherwood, with contributions from R.D. Amos, R.J. Buenker, M. Dupuis, N.C. Handy, I.H. Hillier, P.J. Knowles, V. Bonacic-Koutecky, W. von Niessen, R.J. Harrison, A.P. Rendell, V.R. Saunders, and A.J. Stone. The package is derived from the original GAMESS code due to M. Dupuis, D. Spangler and J. Wendoloski, NRCC Software Catalog, Vol. 1, Program No. QG01 (GAMESS), 1980.

[28] *Parallel inverse iteration with reorthogonalization*, G. Fann and R.J. Littlefield in: *Sixth SIAM Conference on Parallel Processing for Scientific Computing (SIAM)*, (1993) pp.409-413; R.J.Littlefield, K.J. Maschhoff, Investigating the Performance of Parallel Eigensolvers for Large Processor Counts, *Theoretica Chimica Acta* **84** (1993) 457-473.

[29] *PeIGS - Parallel Eigensystem Solver*, G. Fann, http://www.emsl.pnl.gov/docs/nwchem/doc/peigs/docs/peigs.html

[30] M.F. Guest, E. Apra, D.E. Bernholdt, H.A. Fruechtl, R.J. Harrison, R.A. Kendall, R.A. Kutteh, X. Long, J.B. Nicholas, J.A. Nichols, H.L. Taylor, A.T. Wong, G.I. Fann, R.J. Littlefield and J. Nieplocha, *Future Generation Computer Systems* 12 (1996) pp. 273-289.

[31] *A Parallel Second-Order Møller Plesset Gradient*, G. D. Fletcher, A. P. Rendell and P. Sherwood. *Molecular Physics.* **91** (1997) 431.

[32] N. Godbout, D. R. Salahub, J. Andzelm and E. Wimmer, Can. J. Chem. **70**, (1992) 560.

[33] *Performance and Experience with LAPI – a New High-Performance Communication Library for the IBM RS/6000 SP*, G. Shah , J. Nieplocha , J. Mirza, C. Kim, R.J. Harrison, R.K. Govindaraju, K. Gildea, P. DiNicola, C. Bender, Supercomputing 2002.

[34] *ScaLAPACK Home Page*, http://www.netlib.org/scalapack/scalapack_home.html

[35] *A Parallel Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem on Distributed Memory Architectures*, F. Tissuer and J. Dongarra, , *SIAM J. Sci. Comput.* Vol. 20, No. 6, pp. 2223-2236.

[36] *Application of a New Algorithm for the Symmetric Eigenproblem to Computational Quantum Chemistry*, I. Dillon, G. Fann, B. Parlett, Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, March 1997. http://www.cs.utexas.edu/users/inderjit/public_papers/fannchem1.ps.gz

[37] *Linear Algebra*, Wilkinson and Reinsch, Vol. 2 of Handbook for Automatic Computation (1971)

[38] *HPCx: Towards Capability Computing*, M. Ashworth, I.J. Bush, M.F. Guest, A. G. Sunderland, S. Booth, J. Hein, L. Smith, K. Stratford and A. Curioni, , Concurrency and Computation: *Practice and Experience*, (2004) in press.

[39] *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

[40] *THOR-2D: A two-dimensional computational fluid dynamics code*, X. J. Gu & D. R. Emerson, Technical Report, Computational Science and Engineering Department, CLRC Daresbury Laboratory, June 2000.

[41] *Coupled Marine Ecosystem Modelling on High-Performance Computers*, M. Ashworth, R. Proctor, J.T. Holt, J.I. Allen, and J.C. Blackford in Developments in Teracomputing, eds. W. Zwieflhofer and N. Kreitz, 2001, 150-163, (World Scientific).

[42] *A highly spatially resolved ecosystem model for the North West European Continental Shelf*, J.I. Allen, J.C. Blackford, J.T. Holt, R. Proctor, M. Ashworth and J. Siddorn, SARSIA **86** (2001) 423-440.

Acknowledgements

The success of the early stages of the HPCx project and of the implementation of the application benchmarks owes considerably to the efforts of many people. Among those without whom this work would not have been possible are Richard Blake, Paul Durham, Paul Sherwood, Bill Smith and Nic Harrison from CLRC, Mike Brown, Alan Simpson and Arthur Trew from EPCC, and Luigi Brochard and colleagues from IBM.

The HPCx service is funded by the Engineering and Physical Sciences Research Council on behalf of the UK Government Office of Science and Technology.

We are grateful to CSAR, ORNL, PSC and SARA for access to machines.