

Application Performance on the High Performance Switch

Mike Ashworth, Ian J. Bush, Martyn F. Guest, Martin Plummer and Andrew G. Sunderland

CCLRC Daresbury Laboratory, Warrington, WA4 4AD, UK

Email: m.ashworth@dl.ac.uk

Joachim Hein

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

Abstract

In 2004 the HPCx system underwent a major upgrade, including the replacement of the 1.3 GHz POWER4 processors by 1.7 GHz POWER4+ and the “Colony” SP Switch2 being superseded by IBM’s High Performance Switch (HPS). The aim of the upgrade was to effect a factor of two performance increase in the overall capability of the system.

We present the results of a benchmarking programme to compare the performance of the Phase1 and Phase2 systems across a range of capability applications of key importance to UK science. Codes are considered from a diverse section of application domains; materials science, molecular simulation and molecular electronic structure, computational engineering and environmental science.

Whereas all codes performed better on the Phase2 system, the extent of the improvement is highly application dependent. Those which already showed excellent scaling on the Phase1 system, CPMD, CRYSTAL, NAMD, PCHAN and POLCOMS, in some cases following significant optimisation effort by the HPCx Terascaling Team, showed a speed-up which was close to the increase in processor clock speed of 1.31. Other applications, AIMPRO, CASTEP, DL_POLY, GAMESS, and THOR, for which the performance of the interconnect was a critical issue, showed much greater performance increases ranging from 1.65 to 2.88. The PDSYEVD-based matrix diagonalization benchmarks showed even greater improvement, up to a factor of 3.88 for the larger matrix size.

This is a Technical Report from the HPCx Consortium.

Report available from <http://www.hpcx.ac.uk/research/publications/HPCxTR0417.pdf>

© UoE HPCx Ltd 2004

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

1	<i>Introduction</i>	1
2	<i>The HPCx Phase1 and Phase2 Systems</i>	1
2.1	Introduction	1
2.2	The new HPS switch	2
2.3	Change in LPAR size	3
2.4	Memory pages	3
2.5	Effect of the Change in LPAR size	3
2.6	New software environment	3
3	<i>Materials Science</i>	4
3.1	AIMPRO	4
3.2	CASTEP	5
3.3	CPMD	7
3.4	CRYSTAL	10
4	<i>Molecular Simulation</i>	13
4.1	DL_POLY	13
4.2	NAMD	16
5	<i>Molecular Electronic Structure</i>	17
5.1	GAMESS-UK	17
5.2	Parallel Eigensolver Performance	20
6	<i>Computational Engineering</i>	23
6.1	PCHAN	23
6.2	THOR	24
7	<i>Environmental Science: POLCOMS</i>	25
8	<i>Summary</i>	26
	<i>References</i>	28

1 Introduction

In 2004 the HPCx Capability Computing Service underwent a major upgrade from the Phase1 IBM p690 POWER4 cluster with “Colony” SP2 Switch to the Phase2 p690+ POWER4+ system with IBM’s new High Performance Switch (HPS), also known as “Federation”. This was a major upgrade involving all aspects of the system, with new processors, new interconnect, a new release of the AIX operating system and new software for the parallel operating environment.

The aim of the upgrade was to effect a factor of two performance increase in the overall capability of the system from 3 Tflop/s to 6 Tflop/s. This factor of two came from three sources: the increase in clock speed, the improved performance of the interconnect and the increase in number of processors in the application pool from 1024 to 1600. The result was an increase in the headline Linpack figure for the system from 3.241 Tflop/s to 6.188 Tflop/s.

In this report we present performance results for a range of application codes representing a diverse section of research areas: materials science, molecular simulation and molecular electronic structure, computational engineering and environmental science. 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.

Following a summary of the Phase2 upgrade in section 2, we look briefly at the performance of some collective communication operations in section 3. Sections 4 to 8 describe the delivered performance from a selection of well-known codes from five different applications areas before and after the upgrade.

2 The HPCx Phase1 and Phase2 Systems

2.1 Introduction

The HPCx service was launched on 6th December 2002. The six-year contract stipulates three hardware drops in years 0, 2 and 4, and the systems are known as Phase1, Phase2 and Phase3. The Phase1 system consisted of 1280 1.3 GHz p690 processors with 1.28 TBytes of memory and the “Colony” SP Switch2. The Phase2 system came into full user service on 29th April 2004 and was an entirely new delivery of IBM hardware based around an application region of 1600 1.7 GHz p690+ processors with 1.60 TBytes of memory and the High performance Switch (HPS) (formerly known as “Federation”).

The p690+ frames have the same architecture as the p690 frames of the Phase1 system but are equipped with 1.7 GHz POWER4+ (versus 1.3 GHz POWER4) processors. The memory system of the p690 and p690+ is clocked at a third of the processor clock rate so its performance increases pro rata. On the POWER4+ processors, the size of the Level2 caches has increased marginally from 1.44 MB to 1.5 MB. There have also been some modifications and improvements to the way the Level3 cache operates, which may improve cache performance for some codes. For

these reasons the serial performance of codes may be slightly in excess of the pure clock rate increase of $1.7/1.3 = 1.31$. Table 1 summarizes the main features of the HPCx Phase1 and Phase2 systems.

Table 1. Comparative specifications for the Phase1 and Phase2 systems.

	HPCx Phase1	HPCx Phase2
Model	IBM p690 cluster	IBM p690+ cluster
Interconnect	Colony	HPS
Processor architecture	POWER4	POWER4+
Number of processors	1024	1600
Single System Image (SSI) size	8	32
Processor clock (GHz)	1.3	1.7
Processor peak performance (Gflop/s)	5.2	6.8
Level 3 cache size (MB) per processor	16	16
Memory per processor (GB)	1	1
Total system memory (TB)	1.0	1.6
Total system peak performance (Tflop/s)	5.3	10.9

2.2 The new HPS switch

The other major difference with the Phase2 system is that the SP Switch2 "Colony" switch is replaced by IBM's new High Performance Switch (HPS), formerly known as "Federation" [1]. Whereas Colony was designed originally for POWER3 clusters, HPS has been designed specifically with POWER4 (and POWER5) in mind. HPS has greatly increased bandwidth and reduced message latency compared to Colony. Most users should see a reduction in communication time, which may result in some considerable reduction in elapsed time for applications which were constrained by the poorer Colony performance.

Following the introduction of the HPS switch into user service on 29th April 2004, there were two important upgrades to the HPS software (including the switch microcode). The first took place on 28th July 2004 as part of an upgrade to AIX known as Service Pack 7 (SP7) and delivered significant improvements to the point-

to-point message latency and the message transfer times, especially in the mid-range of message lengths (around 64kB). The second upgrade with Service Pack 9 (SP9) which was installed on 27th October 2004 had less effect on MPI performance but contained a significant performance improvement to some LAPI functions, affecting those codes which call LAPI directly, for example via the Global Array tools (see section 5.1).

2.3 Change in LPAR size

The introduction of the HPS switch has a number of knock-on effects. In order to increase the throughput over the Colony switch, the HPCx Phase1 system was configured with each 32-way frame logically divided into four 8-way LPARs each running its own copy of the AIX operating system. This allowed four pairs of adaptors per frame to attach to the Colony network. Due to the better specification of the HPS, dividing the frames into LPARs is no longer necessary. HPCx Phase2 operates as a cluster of 32-way SMP nodes, with just one copy of AIX per frame. For production sized jobs, each 32-way SMP node is dedicated and charged to a single compute job (exclusive access).

2.4 Memory pages

The IBM p690 and p690+ systems can run with two page sizes: small pages are 4 kB in size, large pages are 16 MB. HPCx Phase2 is configured to run with small pages for user jobs, which is the same as on HPCx Phase1. The HPS switch software requires large pages for its buffer space. Some large page space, which is not accessible to the users, is set aside for this purpose.

2.5 Effect of the Change in LPAR size

On the HPCx Phase1 system, with its 8-way LPARs, each LPAR was automatically mapped onto one of the MCMs (Multi-Chip Module) of the p690 frame. On Phase2, with its larger LPARs, user applications can now span across MCMs. It is possible for a process (MPI task) to find itself running on one MCM and its memory to be located in another. We therefore recommend setting memory affinity (set the MEMORY_AFFINITY environment variable to MCM) which requests AIX to allocate memory from the MCM where the process is running. This improves locality of reference. There are also issues around placement of processes with respect to MCMs (process affinity), which are currently under investigation. See also [2].

2.6 New software environment

In order to run the HPS switch, IBM has introduced an entirely new software stack. The Parallel Environment (PE) has been replaced by Cluster Systems Management (CSM) [3]. MPI, which is a part of both products, now sits on top of, and is fully integrated with, LAPI (Low-level Application Programming Interface) - IBM's proprietary one-sided message-passing API. Users should not see any difference from the change of software, but should be aware that any usage of PE specific features may have changed under CSM.

Phase2 runs version 5.2 of IBM's AIX operating system. Phase1 used AIX 5.1.

3 Materials Science

3.1 AIMPRO

AIMPRO (Ab Initio Modelling PROgram) is written by Patrick Briddon et al at Newcastle University [37]. 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.

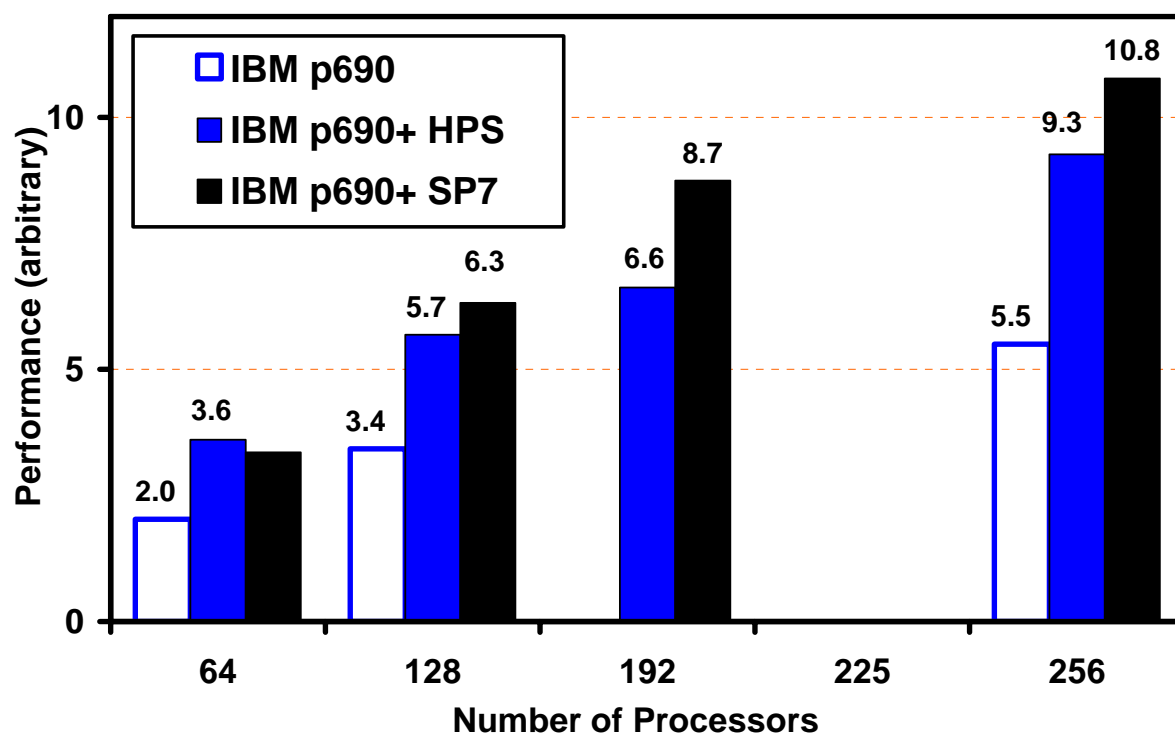


Figure 1 Performance of AIMPRO for the Dat3 dataset (see text) on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

Figure 1 shows execution times on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade. Corresponding times for Dat4 are shown in Figure 2. Performance of the p690+ is generally around twice that of the p690, with further improvements in performance under SP7. This improvement is more marked for Dat3, as the 4 k-point parallelism of Dat4 results in the switch performance having less impact. Consequently we see slightly superior scaling from 128 to 192 processors for Dat4 over Dat3 due to the additional k-point parallelism. The time in this benchmark is roughly split between ScaLAPACK eigensolves (PDSYEVX and PDSYEVJ) and matrix-matrix multiplications in the DIIS code.

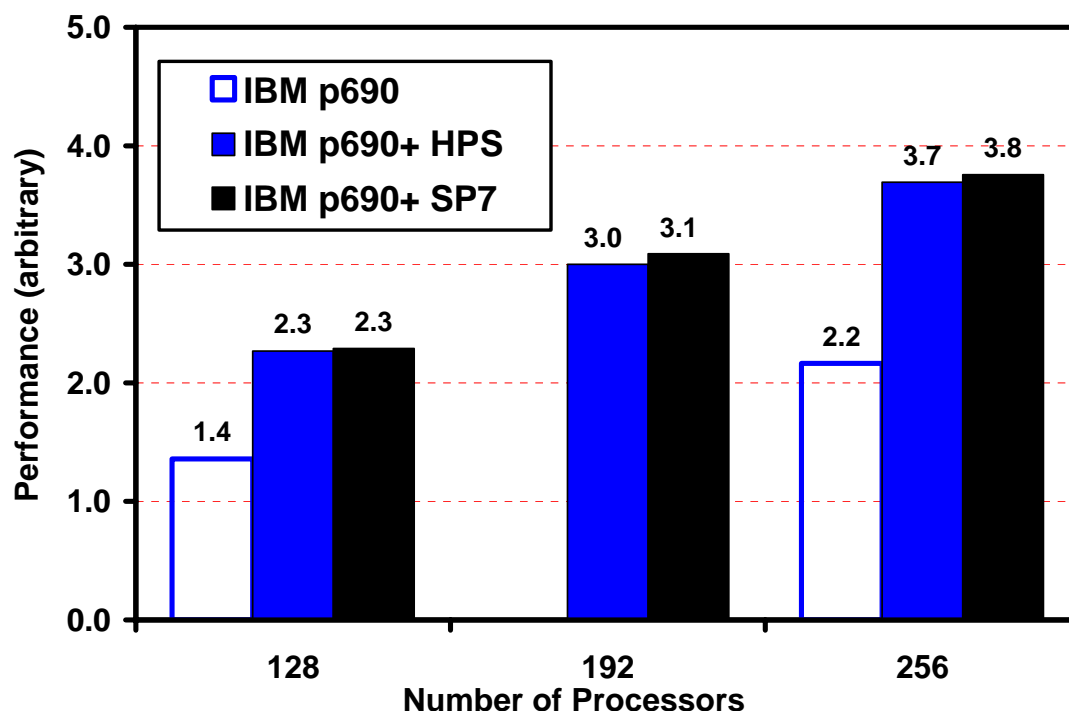


Figure 2 Performance of AIMPRO for the Dat4 dataset (see text) on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

3.2 CASTEP

The Materials Science code CASTEP [8] performs quantum mechanical (density functional) calculations of bulk materials, surfaces and interacting molecules and is distributed commercially by Accelrys Plc. The code expands electronic wavefunctions 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.

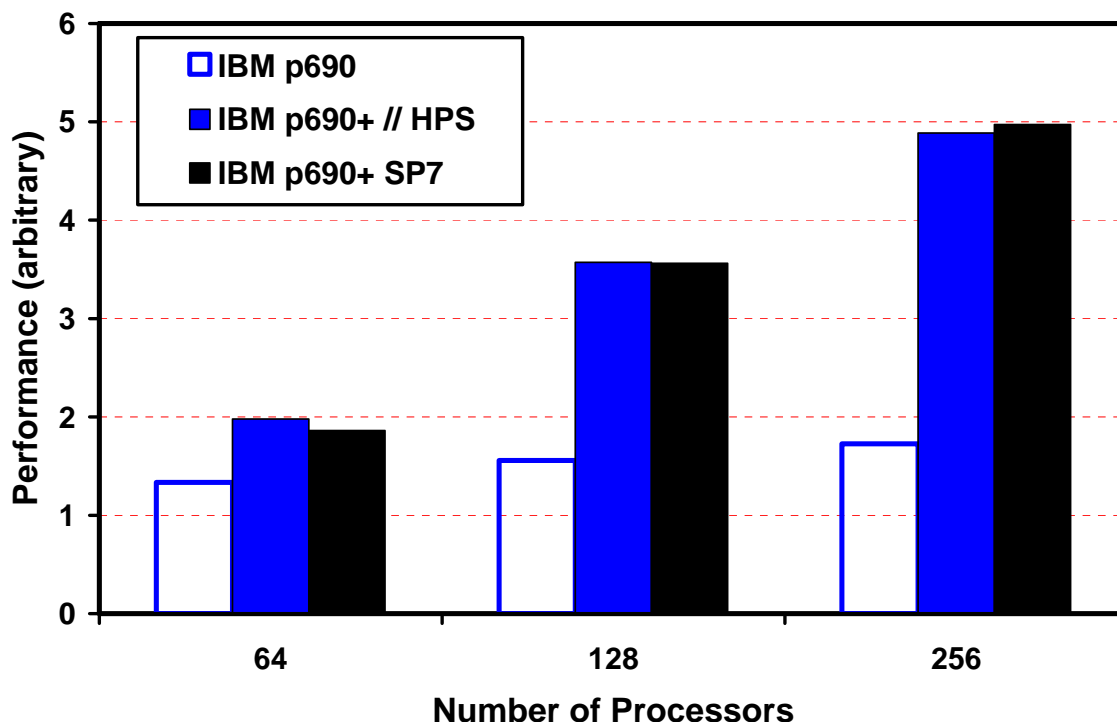


Figure 3 Performance of the CASTEP TiN-8k benchmark on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

Since January 2003 the official version of CASTEP has been a completely rewritten new modular code now officially known as Castep 3.0. Castep 3.1 will be available early in 2005. Substantial optimizations have been made that have improved performance of this code on the Phase1 p690 and Phase2 p690+ systems by ensuring that the communications involved in the data redistribution (the MPI_AllToAllV calls surrounding the FFTs) are “SMP-aware” [9]. This means that, where possible and advantageous, the communications within the SMP node is separated from the inter-node communications. This algorithmic optimization is more complicated for the 4-MCM 32-way LPARs of HPCx Phase 2 than for the single MCM 8-way LPARs of Phase 1, but in combination with IBM memory and process binding (MEMORY_AFFINITY=MCM, MP_TASK_AFFINITY=MCM) the performance gain over Phase 1 (which had already seen a major performance gain over the original unoptimized code [9]) is impressive [10]. Most of the performance gain that was expected only to be possible with SP7 was achieved for the pre-SP7 Phase 2 system.

The 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. CASTEP benchmark Al2O3-G is a total energy calculation of Al₂O₃. There are 5 k-points, the cell contains a slab of 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 benchmark is indicative of other systems which only have a single k-point.

Performance data for the TiN-8k and Al2O3 benchmarks are shown in Figures 6 and 7 respectively. We find that scalability is generally better on the Phase2 p690+ than

the Phase1 p690. This is due to both the use of 32-way LPARs and the superior performance (especially latency) of the HPS interconnect. The 8-way k-point parallelism of the TiN-8k benchmark also essentially allows the collective communications to be limited to within the shared-memory LPARs of the Phase2 p690+ system. The effect of the SMP-aware breakdown of MPI_AllToAllV is seen in the Al₂O₃-G case, where the performance of the code is critically dependent on its performance across all the processors. We are currently concentrating on reducing the serial fraction of CASTEP and improving other aspects of the parallel fraction [10].

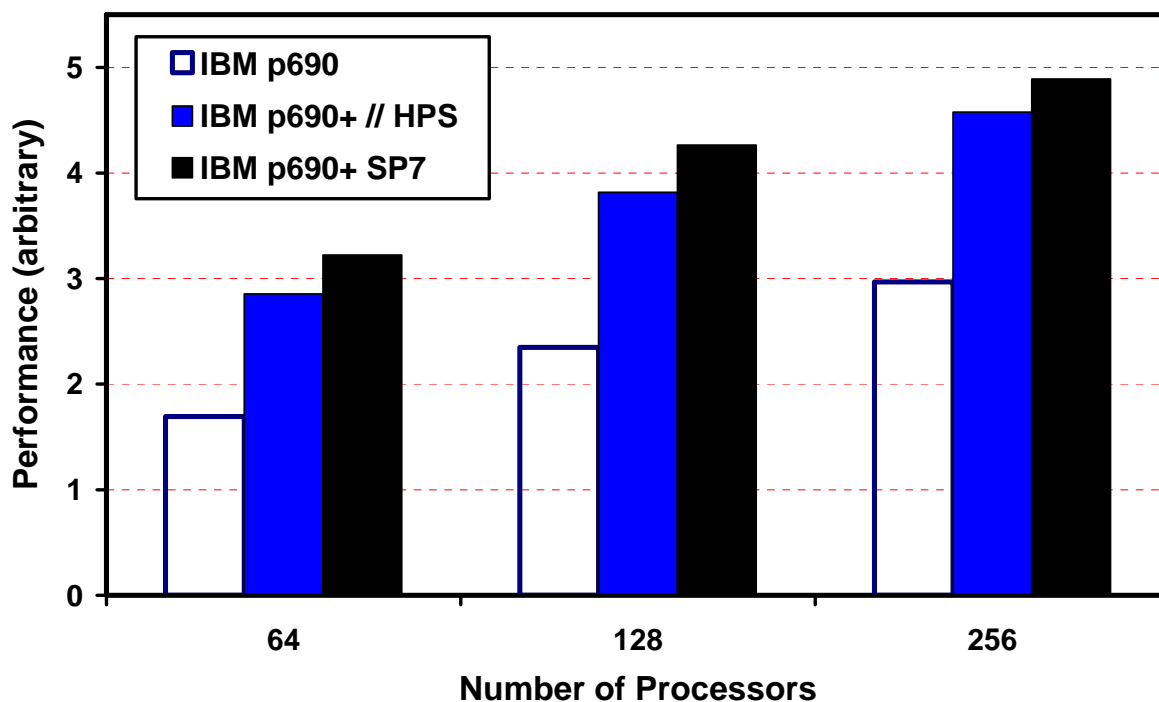


Figure 4 Performance of the CASTEP Al₂O₃-G benchmark on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

3.3 CPMD

The CPMD code is based on the original code by Car and Parrinello [38]. 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 [39][40].

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.

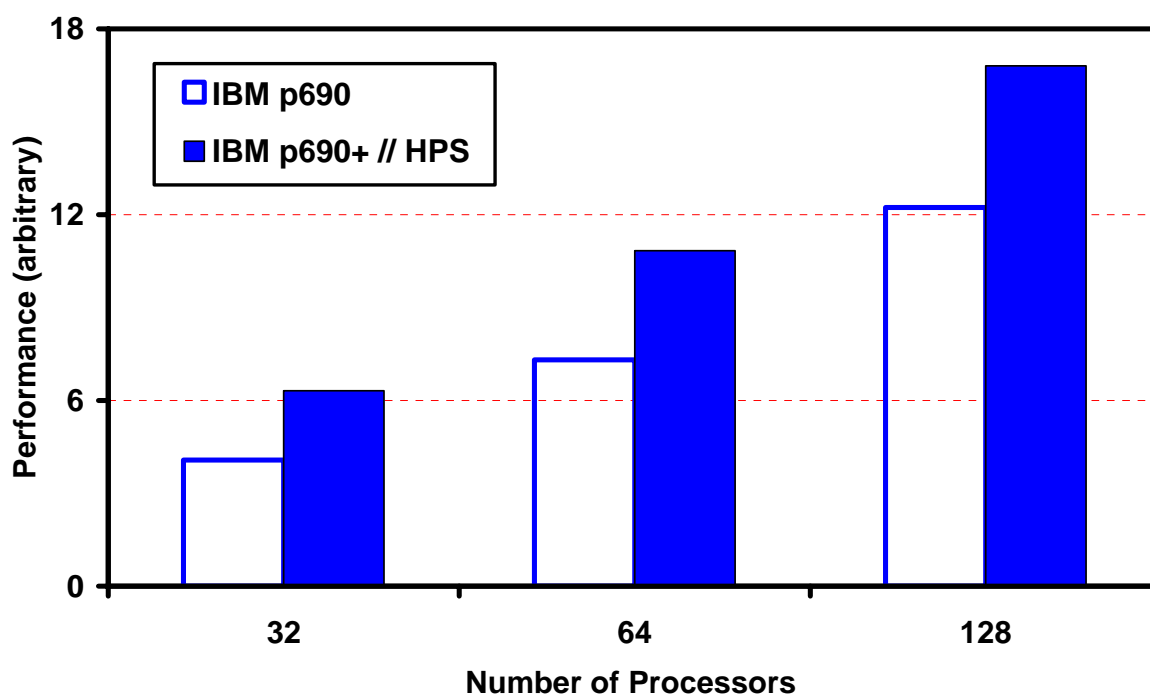


Figure 5 Performance of the CPMD code on the Phase1 p690 and initial Phase2 p690+ systems for the 512-atom Silicon Cluster benchmark case.

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.

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 [41].

Performance of a relatively small 512-atom Silicon Cluster benchmark on the Phase1 p690 and the initial Phase2 p690+ is shown in Figure 5. 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) where 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 and the IBM systems scale well up to 128 processors.

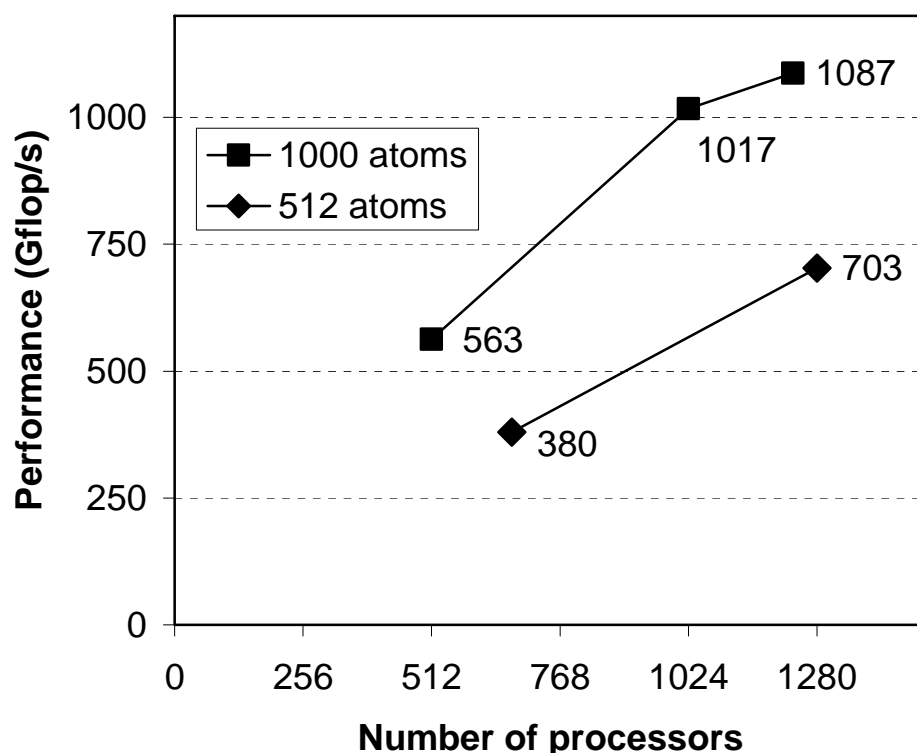


Figure 6 Performance of the CPMD code on the Phase1 p690 for a 1000-atom system

The performance of CPMD has been further tested on the Phase1 p690 using up to 1280 processors (Figure 6) ([41] 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.

3.4 CRYSTAL

The CRYSTAL [11] 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 wave-functions 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 [12].

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 [13]. 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.

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 [14] with 1284 atoms per cell is reported in Figure 7 for the Phase1 p690 system and initial Phase2 p690+ system. 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. Performance on the Phase2 p690+ exceeds the Phase1 system by a ratio that starts at around 1.20 and rises to 1.32 at 1024 processors. The clock speed ratio is 1.31. This benchmark achieves a speed-up of 635 on 1024 processors.

For this run the time is dominated by the evaluation of the four centre integrals and the diagonalization of the Fock matrix, and it is the performance of the latter that explains the improved scaling on the p690+. The evaluation of the integrals scales almost perfectly on both systems, as little message passing is required in this section of the code. The diagonalizer, however, requires extensive message passing, and the improvements due to the HPS are the main reason behind the improved scaling in this section of the code.

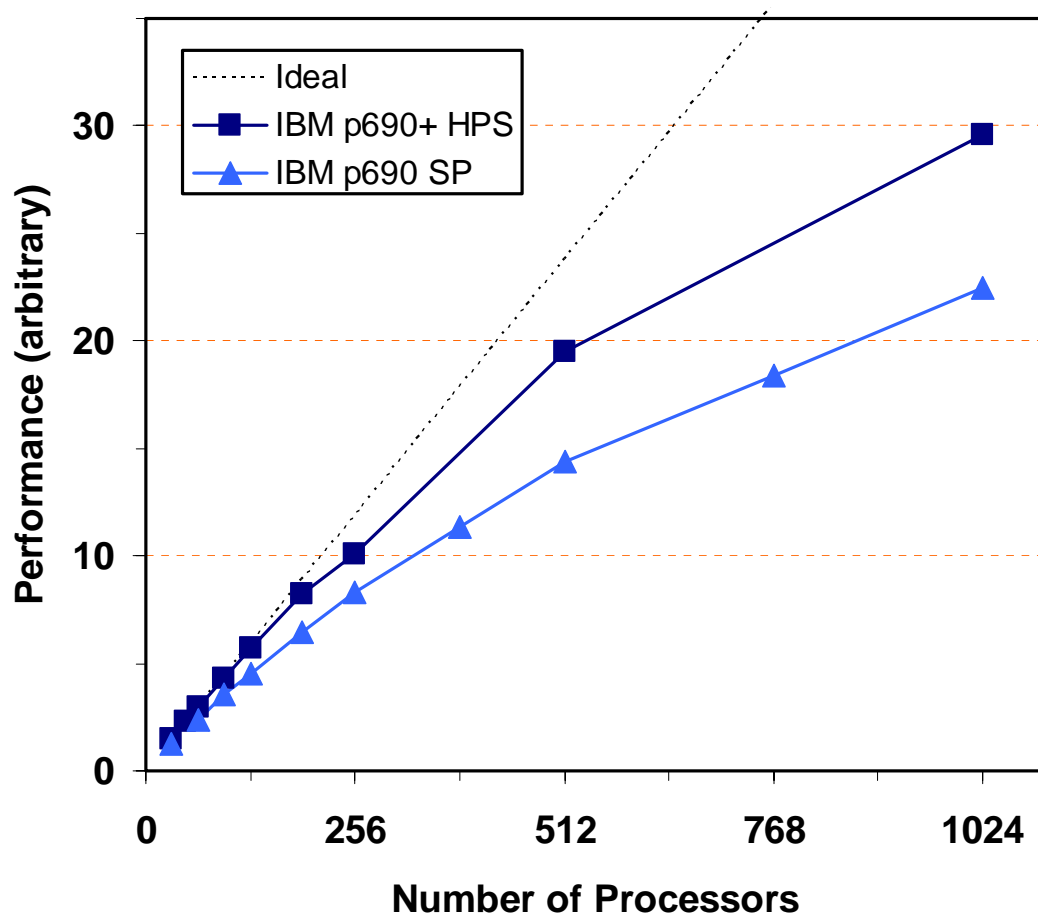


Figure 7 Performance of CRYSTAL-2003 in calculations on crystalline Crambin on the Phase1 p690 system and the initial Phase2 p690+ system.

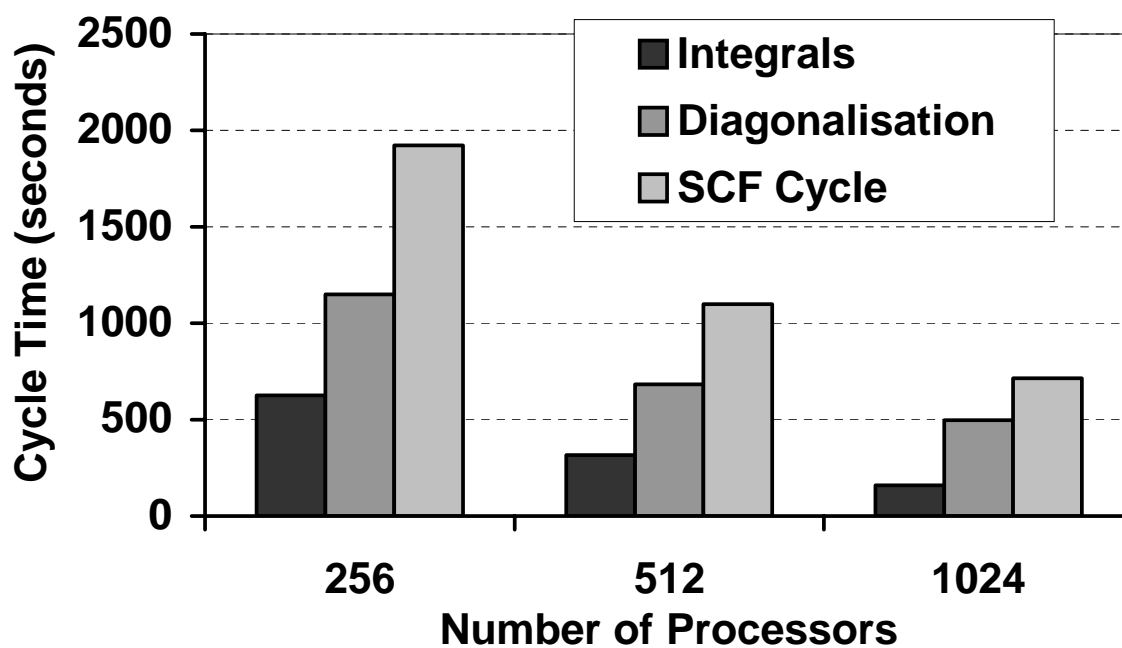


Figure 8 Breakdown of times for the first SCF cycle of CRYSTAL on the Phase1 p690 for 6-31G** calculations on Crystalline Crambin.

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 8 provides a breakdown of the cycle time as a function of processor count.

4 Molecular Simulation

4.1 DL_POLY

DL_POLY [15] 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 CERIOUS² Visualiser from Accelrys and the other on the Java programming language.

DL_POLY supports a wide range of application areas, including [16] 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 [17][18] 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) [18], 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 [19]. As in all Ewald [20] 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 [21]) 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 [22] 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.

Two benchmarks are considered. One (Figure 9) is a Coulombic-based simulation of NaCl, with 216,000 ions that involves use of the Particle Mesh Ewald Scheme, with the associated FFT treated by the algorithm outlined above 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 200 time steps.

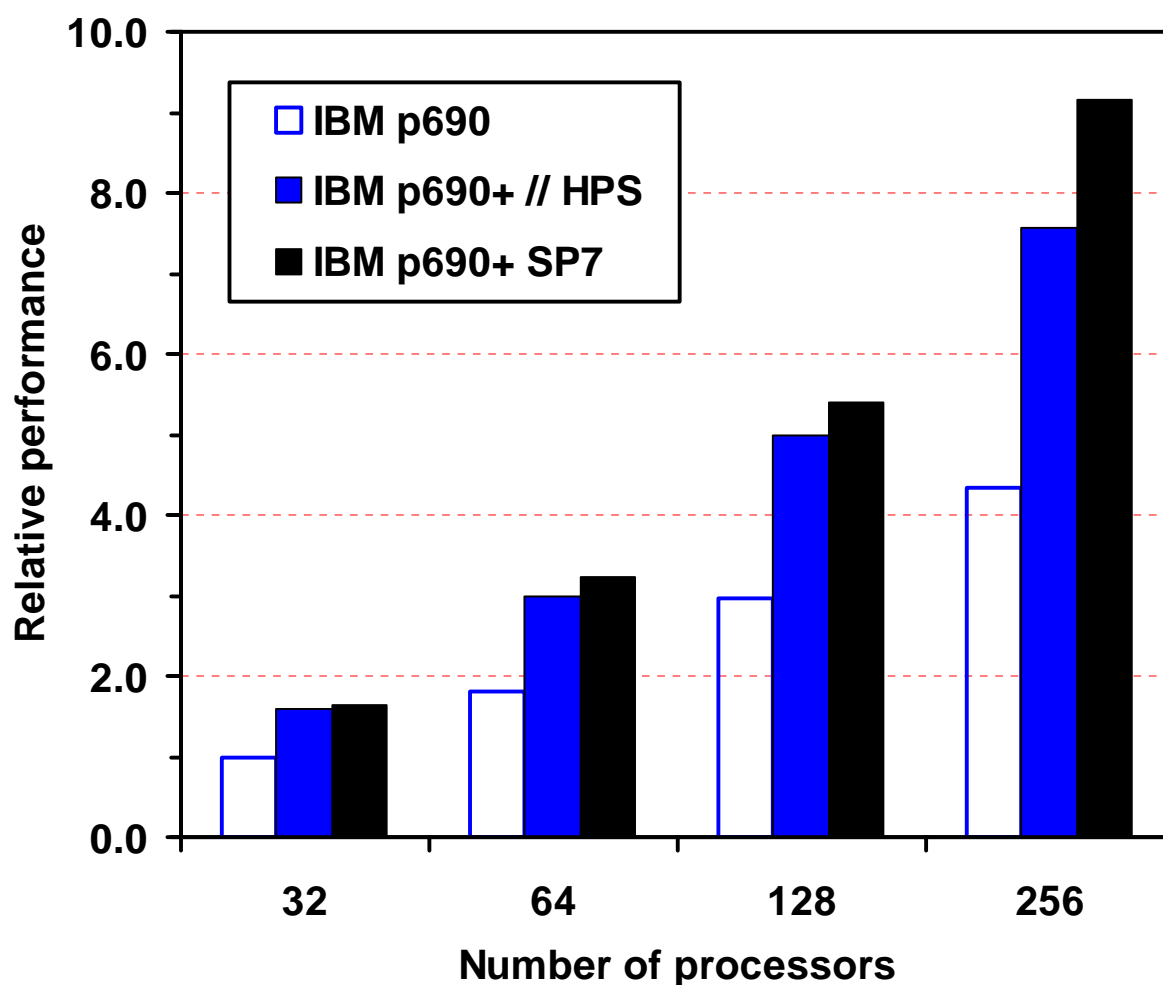


Figure 9 Performance of DL_POLY for NaCl (216,000 atoms, 200 timesteps) on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

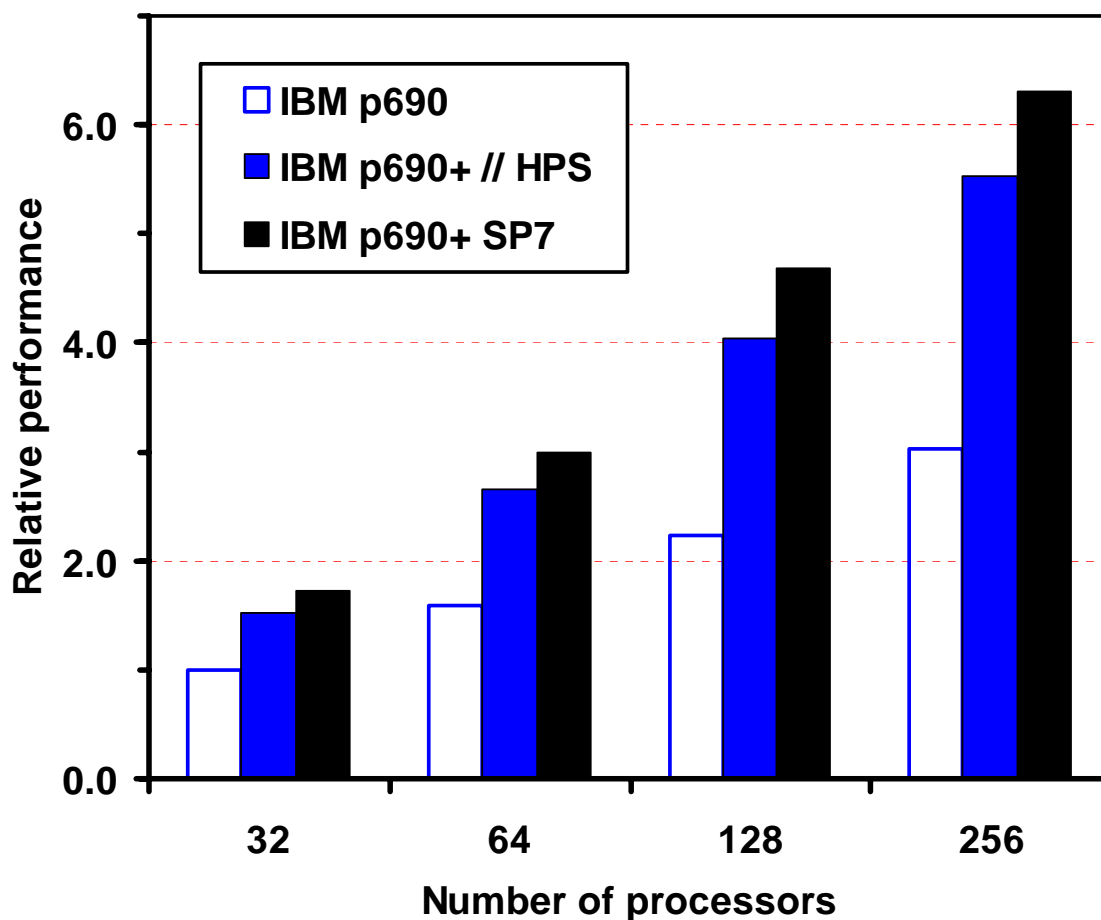


Figure 10 Performance of DL_POLY for Gramicidin A (792,960 atoms, 50 timesteps) on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

The second much larger macromolecular simulation (Figure 10) is for a system of eight Gramicidin-A species (792,960 atoms), with the timings reported for just 50 time steps.

Both benchmarks display marked improvements in scaling due to the HPS. For the simulation of NaCl, shown in Figure 9, we find speedups of 139 and 179 for 256 processors of the Phase1 p690 and Phase2 p690+ SP7 respectively. For this simulation the scaling is almost entirely determined by the scaling of the Ewald summation, which itself reflects the scaling of the FFT. The improved performance reflects both the HPS, and also the move from 8-way to 32-way LPARs. On the Phase1 p690 marked spikes in the timings for the FFT at certain processor counts could be traced back to the messages being passed via the switch rather than through shared memory. The larger size of the LPARs in the Phase2 p690+ means that this effect is much less pronounced, and the improved switch performance reduces the impact on scalability even further.

For the macromolecular Gramicidin-A simulation (Figure 10) the SPME algorithm is again 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 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. However depending on the atomic configuration many short messages may have to be sent, making SHAKE sensitive to the communications latency. It is clear from Figure 10 that the HPS has again improved the scalability of the code. Also shown on the graphs is the effect of Service Pack 7 on the timings. This has again improved the performance in both cases.

4.2 NAMD

NAMD is a parallel, object-oriented molecular dynamics program designed for high performance simulations of large bio-molecular systems [42]. 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 NAMD versions 2.4 and 2.5 β 2 on HPCx. We presently have NAMD version 2.5 installed on the service. On the HPCx Phase1 system we observed a substantial performance improvement between version 2.4 and 2.5 [43]. Figure 11 shows the performance of NAMD version 2.5 based on the NAMD benchmark time for the standard NAMD F₁-ATPase benchmark [44], a system comprising 327,506 atoms.

Comparing the Phase1 system to the Phase2 system prior to SP7, the performance improved by the clock ratio (1.31) for numbers of processor up to 256. However for 512 and 1024 processors, this improvement was not maintained and the performance improvement dropped to 18% and 13% respectively, substantially below the clock ratio. For smaller benchmarks, such as DHFR with 23,558 atoms [45], the situation was even worse. For DHFR the Phase2 performance was inferior to the Phase1 system for 256 and more processors. This highly unsatisfactory situation was resolved with the microcode updates in SP7. When using large number of processors, we now observe a performance improvement, which substantially exceeds the clock ratio. For F₁-ATPase on 1024 processors the performance improves by 47% when comparing to the Phase1 system and for the small DHFR benchmark on 256 processors we see an improvement of 70%. We traced the poor performance of the Phase2 system prior to service pack 7 to an inability of the system to handle a large number of small messages [46]. According to the NAMD website, the parallel scaling provided by the HPS for this code presently exceeds the abilities of Myrinet and SGI's Numalink technology [44].

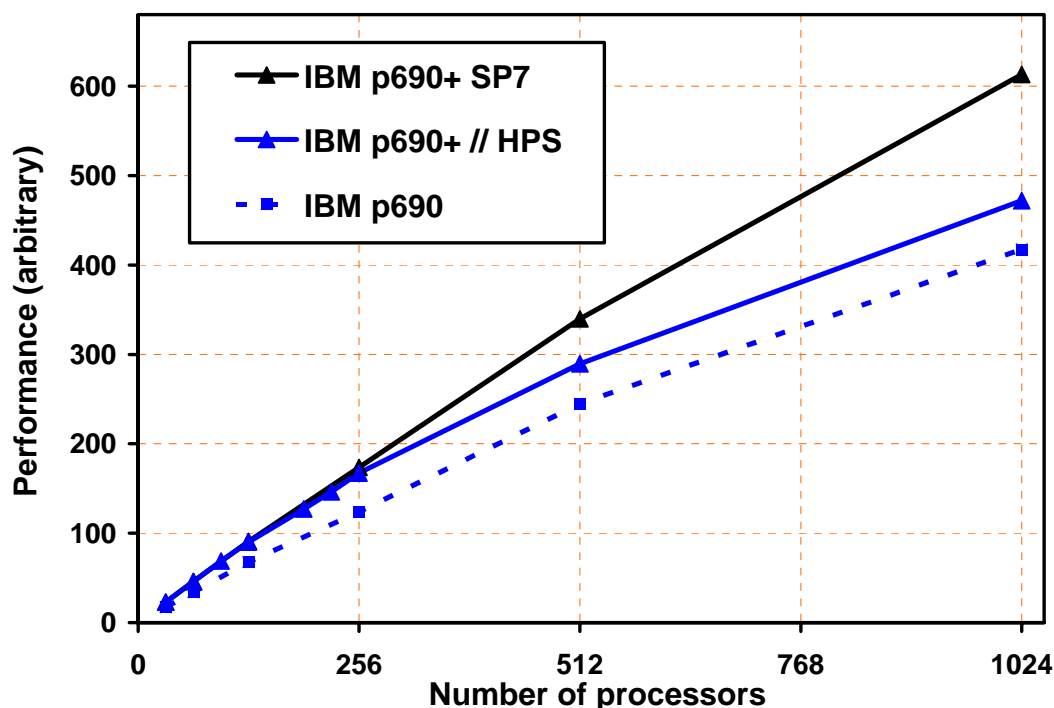


Figure 11 Performance of the NAMD F_1 -ATPase benchmark on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

5 Molecular Electronic Structure

5.1 GAMESS-UK

GAMESS-UK [24] 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 [4] 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 [25].

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, 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 [28].

Substantial modifications were required to enable the SCF and DFT 2nd derivatives [29] 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.

The performance of the DFT and SCF 2nd Derivative modules on the Phase1 p690 and Phase2 p690+ SP7 and p690+ SP9 systems is shown in Table 2. Unless stated otherwise, the DFT calculations did not exploit CD fitting, but evaluated the coulomb matrix explicitly. The less than impressive scalability of both GAMESS-UK (and also NWChem) on the Phase1 p690 and Phase2 p690+ systems (up to and including SP7) arises to some extent from the dependency of both codes on a Global Array implementation that is dependent on IBM's LAPI communication library [31]. The implementation of LAPI on POWER4-based architectures has proved far from optimal, with the measured latencies and bandwidths significantly inferior to those measured on corresponding POWER3-based systems. The recent release of SP9 on the Phase2 p690+ has done much to address these shortcomings, as is evident from the timings on higher processor counts in Table 2.

Table 2. Time in Wall Clock Seconds for Four GAMESS-UK Benchmark Calculations on the Phase1 p690 and Phase2 p690+ SP7 and p690+ SP9 systems.

CPU's	IBM p690	IBM p690+ (SP7)	IBM p690+ (SP9)
Cyclosporin (1000 GTOs) DFT/B3LYP 6-31G			
32	556	385	392
64	369	254	231
128			160
Cyclosporin (1855 GTOs) DFT/B3LYP 6-31G**			
32	2047	1366	1319
64	1429	876	804
128	1193	739	555
Valinomycin (882 GTOs) DFT/HCTH			
32	1099	802	816
64	635	463	441
128	489	309	260
Valinomycin (882 GTOs) DFT/HCTH (J-fit)			
32	473	269	279
64	370	243	193
128	355		
Valinomycin (1620 GTOs) DFT/HCTH			
32	4723	3513	3502
64	2690	1966	1886
128	1777	1259	1104
256	1343	985	731
(C ₆ H ₄ (CF ₃)) ₂ 2 nd Derivatives DFT/HCTH			
32	842	457	471
64	471	362	295
128	294	240	180

Considering the DFT results, modest speedups ranging from 55 (Phase1 p690) to 76 (Phase2 p690+ SP9) 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 [30]. Speedups of 85 and 89 are obtained on 128 processors of the Phase1 p690 and the Phase2 p690+ SP7, rising to 101 with the SP9 upgrade, the last figure being comparable to that found on other competing systems in the 1620 GTO calculation. The enhanced performance of the DFT 2nd Derivative module arises from the decreased dependency on latency exhibited by the current implementation compared to the DFT energy module.

5.2 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 [46] and Parallel Eigensolver System software (PeIGS) library routines (from PNNL, [25][26][27]).

The latest release of ScaLAPACK (v1.7) includes new parallel algorithms based on divide-and-conquer techniques [48]. Also, algorithms developed recently by Dhillon et al [49] are included in a new release of PeIGS (v3.0) [25]. In addition, Ian Bush at Daresbury Laboratory has developed BFG [10], a parallel one-sided Block Factored Jacobi implementation based upon the techniques described by Littlefield et al [25].

The fastest serial diagonalisation algorithm is that based upon Householder reduction to tridiagonal form followed by diagonalization [50]. 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. For the very large matrices that one wishes to diagonalize on high-end systems this is highly undesirable, and may even be impossible.

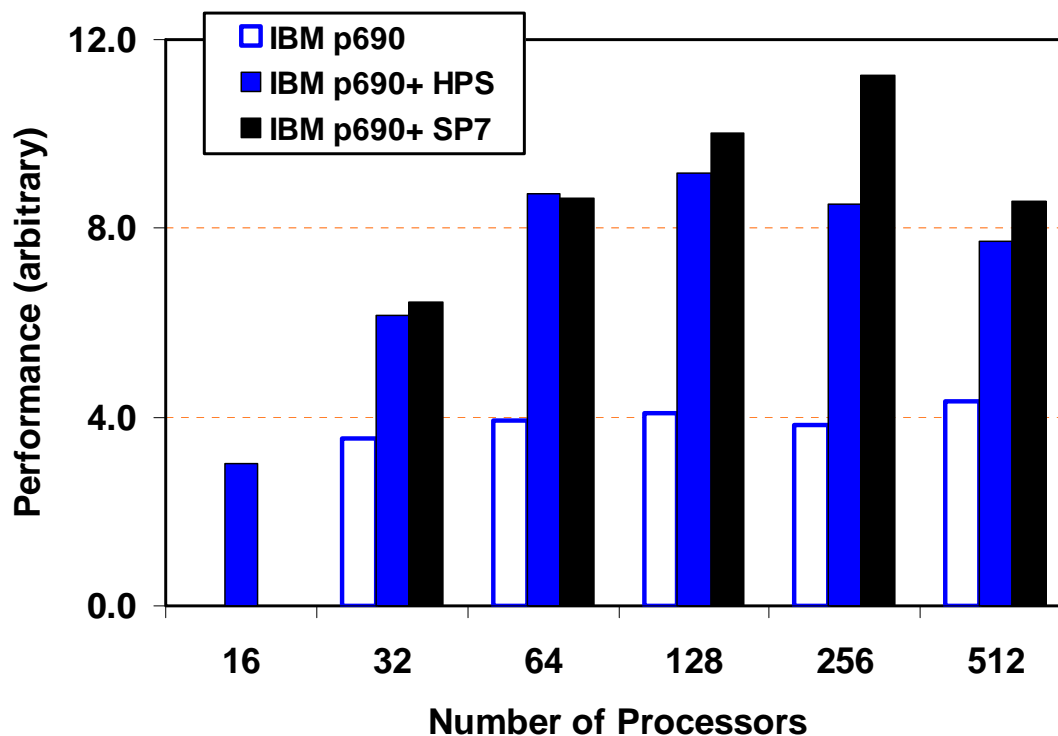


Figure 12 Performance (inverse time) of the ScaLAPACK divide-and-conquer routine (PDSYEVD) for a matrix dimension of 3888 on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

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 [25], 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.

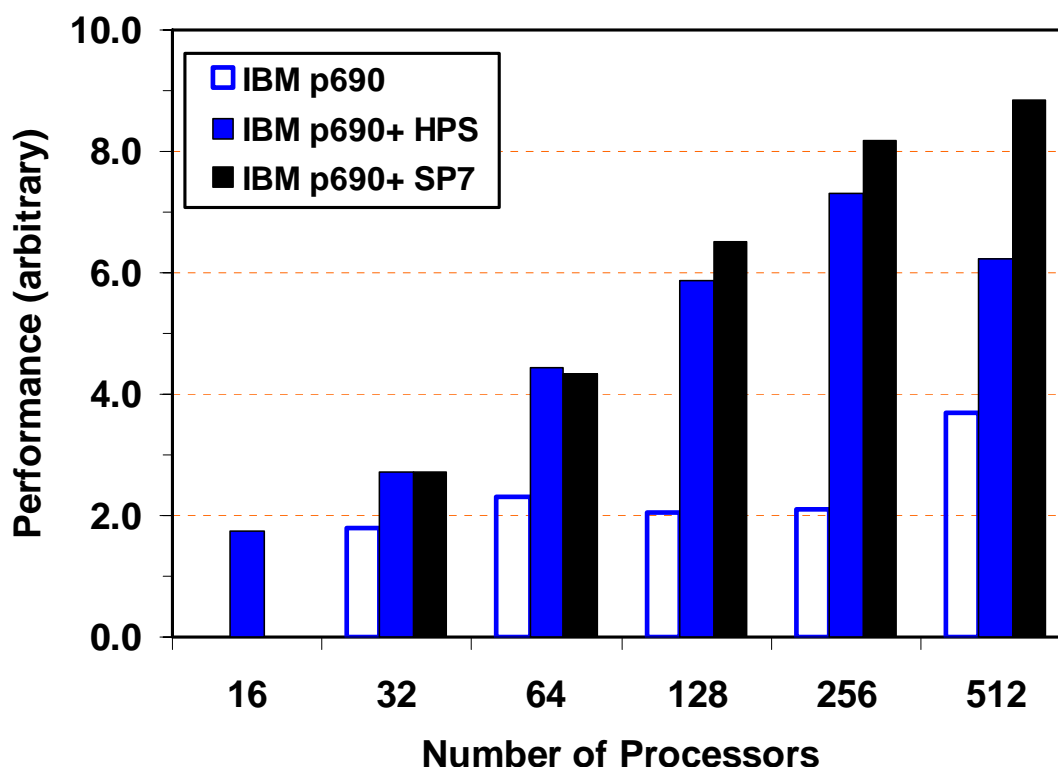


Figure 13 Performance (inverse time) of the ScaLAPACK divide-and-conquer routine (PDSYEVD) for a matrix dimension of 7194 on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

We have shown elsewhere [32] 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 [49]. Relative performance data are shown in Figure 12 for a matrix of dimension 3888 and in Figure 13 for a matrix of dimension 7194 on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade. For the smaller matrix size the Phase2 system is scaling much better than Phase1, with a Phase2:Phase1 ratio on 256 processors of 2.2 rising to 2.9 with the SP7 upgrade. As expected the larger problem shows scalability out to larger numbers of processors and the improvement over Phase1 is even greater with ratios at 256 processors of 3.5 (initially) and 3.9 (SP7). The PDSYEVD performance with SP7 now continues to rise up to 256 processors for the 3888 case, and up to 512 processors for the 7194 case, in contrast to the initial Phase2 performance which tailed off much earlier.

6 Computational Engineering

6.1 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.

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 [33]. The code has been written using standard Fortran 90 code together with MPI in order to be efficient, scalable and portable across a wide range of high-performance platforms. The PCHAN benchmark is a simple turbulent channel flow benchmark using the SBLI code. Figure 14 shows performance results for the T3 benchmark data case (360x360x360) from the Phase1 system, the initial Phase2 system and Phase2 following the SP7 upgrade. The performance shows scaling on all systems which is close to ideal. The SP7 microcode upgrade clearly improves the scaling at higher numbers of processors where the communications becomes significant, with a 10% improvement at 512 processors and 18% at 768. The Phase2 p690+ system is seen to outperform the Phase1 p690 by a factor of 1.41 at 512 processors, a factor which is somewhat greater than the clock speed ratio of 1.31.

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 Phase2 p690+ all the way out to 1280 processors. Hardware profiling studies of this code have shown that its absolute performance is highly dependent on the cache utilisation and bandwidth to main memory [7]. This remains the subject of further study.

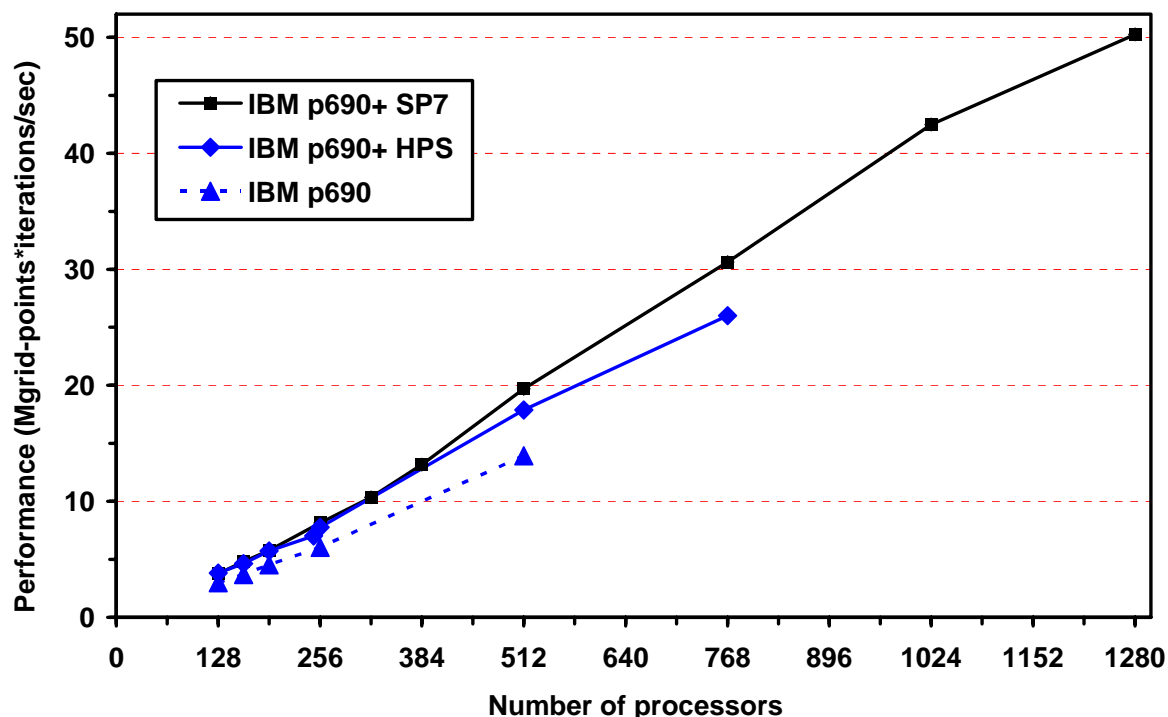


Figure 14 The PCHAN T3 (360x360x360) benchmark on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

6.2 THOR

THOR [51] 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 multi-block 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-order 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 multi-block structure in the code, the computational domain was decomposed to multi-blocks 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 2,619,156 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 15. 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 sequences of indirect memory accesses. Scaling from 32 to 64 processors on the Phase2 system was poor at 1.32. For the Phase2 p690+ SP7 system this improves to 1.55. That the Phase2:Phase1 ratio on 64 processors of 2.24 (initially) and 2.52 (SP7) is much greater than the clock speed ratio of 1.31, shows the importance of the increased communications performance of the HPS for this code. Initially we could not run the benchmark on 32 processors on Phase2 due to memory constraints, but this has subsequently been resolved.

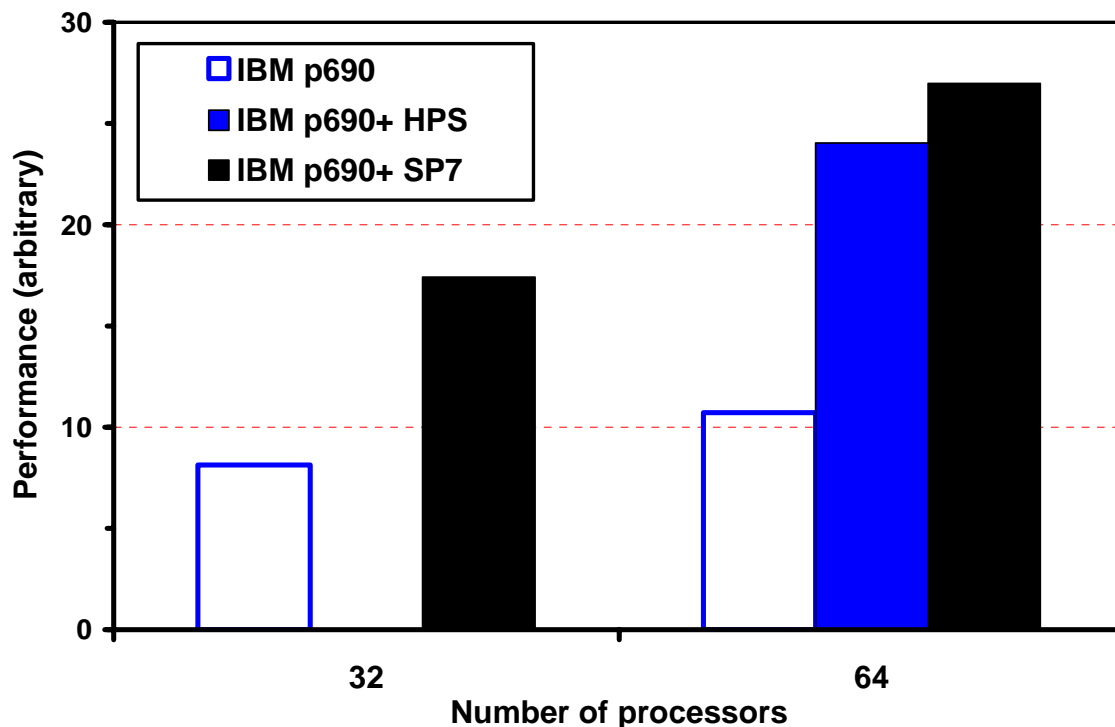


Figure 15 Performance of the THOR code on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

7 Environmental Science: POLCOMS

The Proudman Oceanographic Laboratory Coastal Ocean Modeling System (POLCOMS) has been developed to tackle multi-disciplinary studies in coastal/shelf environments [34]. 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) [35]. 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 16 as the amount of work (gridpoints \times timesteps) divided by the time.

The excellent scaling found on the Phase1 system has been further improved, both with the initial Phase2 system, which gave a speed-up of 1.25 on 768 processors, and with the SP7 upgrade, which improved this ratio to 1.37. As with PCHAN, the absolute performance is highly dependent on cache and memory issues [7].

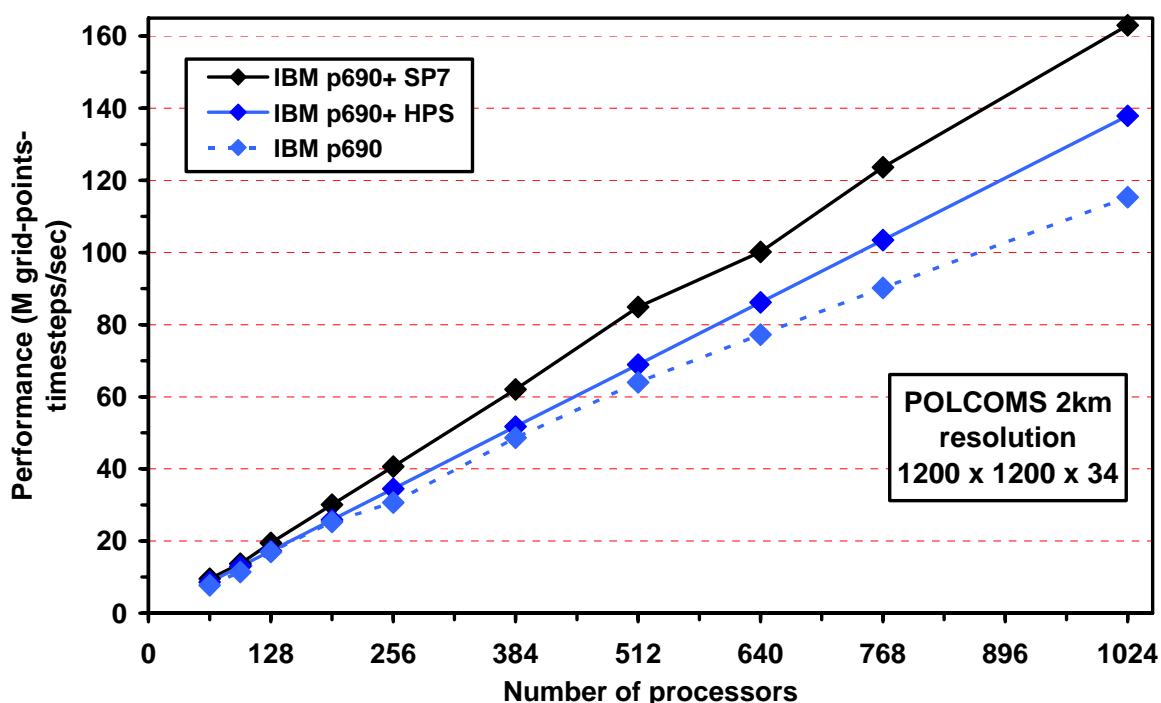


Figure 16 Performance of the POLCOMS code running a 2km simulation (1200x1200x34 grid points) on the Phase1 p690 system, the initial Phase2 p690+ and following the SP7 upgrade.

8 Summary

We have presented performance results from ten full application codes and from one of the ScaLapack eigensolvers that characterise the evolution of the HPCx service from the initial Phase1 p690 1.3 GHz SP Switch2 offering through the Phase2 upgrade to p690+ 1.7 GHz processors with the new IBM HPS switch. Applications have been considered from a diverse section of disciplines: materials science, molecular simulation and molecular electronic structure, computational engineering

and environmental science. Figure 17 summarizes the relative performance of these applications at 128 processors between Phase1 and the Phase2 SP7 configuration.

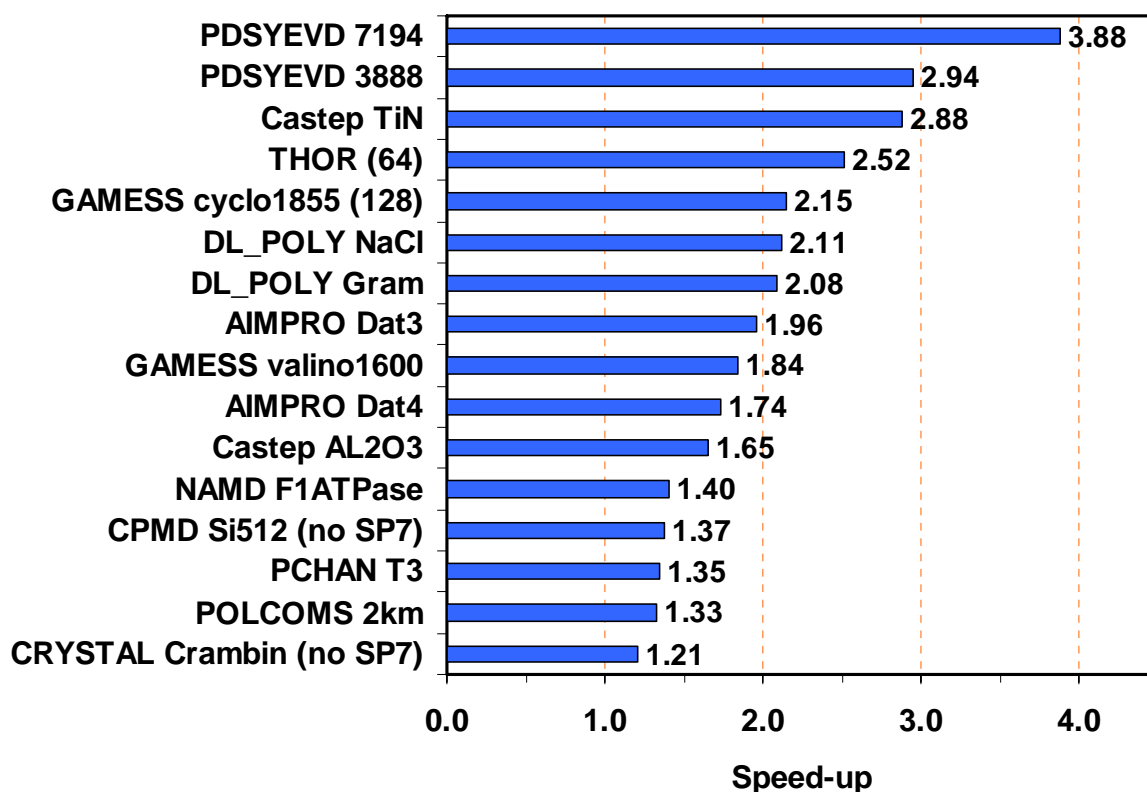


Figure 17 Application Performance: Speed-up of the Phase2 p690+ SP7 system relative to the Phase1 p690 system at 256 processors across a variety of applications and associated data sets. (For THOR the comparison is made at 64 processors, for GAMESS cyclosporin 1855 GTOs it is at 128 processors. The measurements for CPMD and CRYSTAL are prior to the SP7 upgrade).

Whereas all codes performed better on the Phase2 system, the extent of the improvement is highly application dependent. Those which already showed excellent scaling on the Phase1 system, CPMD, CRYSTAL, NAMD, PCHAN and POLCOMS, in some cases following significant optimisation effort by the HPCx Terascaling Team, showed a speed-up which was close to the increase in processor clock speed of 1.31.

Other applications, AIMPRO, CASTEP, DL_POLY, GAMESS, and THOR, for which the performance of the interconnect was a critical issue, showed much greater performance increases ranging from 1.65 to 2.88. The PDSYEVD-based matrix diagonalization benchmarks showed even greater improvement, up to a factor of 3.88 for the larger matrix size.

Many of the application codes that exhibit enhanced performance on the Phase2 system 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” [9].
- The incorporation of a somewhat faster, and more numerically stable version of the parallel Jacobi diagonalizer [13] in CRYSTAL 2003, and the rationalization of the memory management within the code
- The introduction of a parallel 3D FFT [22] 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.

It is the applications with unavoidable large-scale global communications (from this paper DL_POLY, GAMESS and NAMD, but also H2MOL and PFARM [32]) which provide the most severe test for the communications sub-system of the IBM p690 clusters and other modern MPP systems. 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.

References

- [1] IBM Red Book “An Introduction to the New IBM eServer pSeries High Performance Switch”
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246978.pdf>
- [2] IBM Red Book “Performance and Tuning Considerations for the p690 in a Cluster 1600” <http://www.redbooks.ibm.com/redbooks/pdfs/sg246841.pdf>
- [3] IBM Red Book: “An Introduction to CSM 1.3 for AIX 5L”
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246859.pdf>
- [4] *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.
- [5] *Computational Chemistry Applications: Performance on High-End and Commodity-class Computers*, M.F. Guest and P. Sherwood, Proceedings of HPCS 2002, Moncton, Canada, 2002.
- [6] *HPCx: a new resource for UK Computational Science*, M. Ashworth, I.J. Bush, M.F. Guest, M. Plummer, A.G. Sunderland, S.P. 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)

- [7] *Single Node Performance of Applications and Benchmarks on HPCx*, Mark Bull, HPCx Technical Report HPCxTR0416, 2004
http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0416.pdf
- [8] *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 (the authors together with K Refson represent the Castep Developers' Group)
<http://www.accelrys.com>
- [9] *"An LPAR-customized MPI_AllToAllV for the Materials Science code CASTEP"*, M. Plummer and K. Refson, HPCx Technical Report HPCxTR0401, 2004
http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0401.pdf
- [10] M. Plummer and K. Refson, in preparation
- [11] *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>
- [12] *Ab Initio Modelling in Solid State Chemistry*, European Summer School, MSSC2002, 8-13 September 2002, Torino, Italy
<http://www.chimifm.unito.it/teorica/mssc2002>.
- [13] *A New Implementation of a Parallel Jacobi Diagonalizer*, I.J. Bush
<http://www.cse.clrc.ac.uk/arc/bfg.shtml>
- [14] *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* **97** (2000) 3171
- [15] *DL_POLY: A general purpose parallel molecular dynamics simulation package*, W Smith and T R Forester, *J. Molec. Graphics* **14** (1996) 136.
- [16] *DL_POLY: Applications to Molecular Simulation*, W. Smith, C. Yong and M. Rodger, *Molecular Simulation* **28** (2002) 385.
- [17] *Application Performance on High-end and Commodity-class Computers*, M.F. Guest, <http://www.ukhec.ac.uk/publications/reports/benchmarking.pdf>
- [18] *The DL-POLY Molecular Simulation Package*, W. Smith,
http://www.cse.clrc.ac.uk/msi/software/DL_POLY/
- [19] *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.
- [20] *Computer Simulation of Liquids*, M.P. Allen and D.J. Tildesley, Clarendon Oxford 1987.
- [21] *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/>
- [22] *A Parallel Implementation of SPME for DL_POLY 3*, I.J. Bush and W. Smith,
<http://www.cse.clrc.ac.uk/arc/fft.shtml>

- [23] 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.
- [24] 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.
- [25] *Parallel inverse iteration with reorthogonalization*, G. Fann and R.J. Littlefield in: *Sixth SIAM Conference on Parallel Processing for Scientific Computing (SIAM)*, (1993) 409-413
- [26] R.J. Littlefield, K.J. Maschhoff, Investigating the Performance of Parallel Eigensolvers for Large Processor Counts, *Theoretica Chimica Acta* **84** (1993) 457-473
- [27] *PeIGS - Parallel Eigensystem Solver*, G. Fann,
<http://www.emsl.pnl.gov/docs/nwchem/doc/peigs/docs/peigs.html>.
- [28] 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) 273-289.
- [29] *A Parallel Second-Order Møller Plesset Gradient*, G. D. Fletcher, A. P. Rendell and P. Sherwood. *Molecular Physics*. **91** (1997) 431.
- [30] *Optimization of Gaussian-type Basis Sets for Local Spin Density Functional Calculations. Part I. Boron through Neon, Optimization Technique and Validation*, N. Godbout, D. R. Salahub, J. Andzelm and E. Wimmer, *Can. J. Chem.* **70**, (1992) 560.
- [31] *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.
- [32] *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*, (2005) in press.
- [33] *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>
- [34] *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. Zwiefelhofer and N. Kreitz, 2001, 150-163, (World Scientific).

- [35] *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.
- [36] 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>
- [37] *The structures and properties of tetrafluoromethane, hexafluoroethane, and octafluoropropane using the AIMPRO density functional program*, R.W. Zoellner, C.D. Latham, J.P. Goss, W.G. Golden, R. Jones and P.R. Briddon, *Journal of Fluorine Chemistry*, **121** (2) (2003) 193-199
- [38] *Unified Density-Functional-Theory and Molecular Dynamics*, R. Car and M. Parrinello, *Phys. Rev. Lett.* **55** (1985) 2471-2474
- [39] *ab-initio Molecular Dynamics: Theory and Implementation*, D. Marx and J. Hutter, in "Modern Methods and Algorithms of Quantum Chemistry", J. Grotendorst (Ed.), NIC Series, Vol. 1, FZ Jülich, Germany, 2000; see also <http://www.fz-juelich.de/nicseries/Volumel>
- [40] CPMD V3.8, Copyright IBM Corp. 1990-2003, Copyright MPI für Festkörperforschung Stuttgart 1997-2001. See also <http://www.cpmc.org>.
- [41] *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).
- [42] *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.
- [43] *The performance of NAMD on HPCx*, J. Hein, HPCx technical report, 2003, HPCxTR0310
http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0310.pdf
- [44] NAMD Performance, Theoretical and Computational Biophysics Group, NIH Resource for Macromolecular Modeling and Bioinformatics,
<http://www.ks.uiuc.edu/Research/namd/performance.html>
- [45] *MD Benchmarks for Amber, CHARMM and NAMD*,
<http://amber.scripps.edu/amber8.bench2.html>
- [46] *On the performance of molecular dynamics applications on current high-end systems*, J. Hein, F. Reid, L. Smith, I.J. Bush, M.F. Guest, P. Sherwood, preprint (2004) to appear in: *Philosophical Transactions of the Royal Society*
- [47] *ScaLAPACK Home Page*,
http://www.netlib.org/scalapack/scalapack_home.html
- [48] *A Parallel Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem on Distributed Memory Architectures*, F. Tisseur and J. Dongarra, , *SIAM J. Sci. Comput.* Vol. 20, No. 6, pp. 2223-2236.

- [49] *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
- [50] *Linear Algebra*, Wilkinson and Reinsch, Vol. 2 of Handbook for Automatic Computation (1971)
- [51] *THOR-2D: A two-dimensional computational fluid dynamics code*, X. J. Gu & D. R. Emerson, Technical Report, Computational Science and Engineering Department, CCLRC Daresbury Laboratory, June 2000.