Application performance on the UK's new HECToR service

Fiona Reid[1,2], Mike Ashworth[1,3], Thomas Edwards[2],
Alan Gray[1,2], Joachim Hein[1,2], Peter Knight[4],
Alan Simpson[1,2], Kevin Stratford[1,2], Michele Weiland[1,2]
[1]*HPCX Consortium*
[2]*EPCC, The University of Edinburgh, James Clerk Maxwell Building, Mayfield Road, Edinburgh, EH9 3JZ, UK*
[3]*STFC Daresbury Laboratory, Warrington WA4 4AD, UK*
[4]*EURATOM/UKAEA Fusion Association, Culham Science Centre, Abingdon, Oxfordshire, OX14 3DB, UK*

Abstract

HECToR is the UK's new high-end computing resource available for research funded by the UK Research Councils. The HECToR Cray XT4 system began user service in October 2007 and comprises 5,564 dual core 2.8GHz AMD Opteron processors. The results of running a number of synthetic benchmarks and popular application codes which are used by the UK academic community are presented. The synthetic benchmarks include STREAMS and MPI benchmarks. The application benchmarks include fluid dynamics, molecular dynamics, fusion, materials science and environmental science codes. The results are compared with those obtained on the UK's HPCx service which comprises 160 IBM e-Server p575 16-way SMP nodes each containing 8 dual core 1.5GHz IBM Power5 64-bit RISC chips. Where appropriate, results are also included from the HECToR Test and Development system which has a different memory structure from the main system. It is found that there is not much difference between the systems in terms of comparing similar numbers of processing cores, but HECToR is a much larger system with many more cores and a more scalable interconnect. Memory bandwidth is seen to be a bottleneck for certain applications on both systems, with HECToR more seriously affected.

1 Introduction

HECToR is the UK's new high-end computing resource available for research funded by the UK Research Councils. The HECToR service began in October 2007, and will overlap with the established HPCx service in a complementary fashion. This paper gives, a performance comparison of the two services, via a range of synthetic and application benchmarks, with the aim of providing users with guidance on which system is better suited to their computational requirements.

In Section 2, the architectures of the machines are described, and the processing capabilities of the services are qualitatively compared. In Section 3 results from synthetic benchmarks are given and results from real application benchmarks are presented in Section 4.

2 Architecture

The HPCx (Phase 3) system [15] features 160 IBM eServer 575 compute nodes, each containing 8 dual core 1.5GHz IBM Power5 64-bit RISC chips, i.e. a total of 2560 cores are available. Each core has a private L1 instruction cache of 64KB and L1 data cache of 32KB. The L2 combined data and instruction

cache of 1.9MB, and L3 cache of 36MB, are each shared between the two cores in a chip. Within a node, cores can communicate via shared memory, and communication between nodes is provided by an IBM High Performance Switch (HPS).

The HECToR (Phase 1) system [14] comprises 5664 nodes, each with one dual core AMD Opteron 2.8GHz chip, i.e. a total of 11,328 cores. Each core has access to a private 64KB L1 instruction cache, 64KB L1 data cache and 1MB L2 cache. The two cores within a chip (or node) share 6GB of main memory. Each node controls a Cray Seastar2 network chip. Each Seastar2 has six links and the network is configured in a 3D toroidal topology.

Table 1 compares the processing resources available on the two systems. The processing core clock frequency on HECToR is almost a factor of two higher than HPCx. However, the HPCx cores can perform two fused multiply-add operations, i.e. four floating point operations, per cycle, whereas the HECToR cores can perform one multiplication plus one addition, i.e. two floating point operations, per cycle. This leads to a very similar peak performance per core. HECToR has around a factor of four more cores available, and therefore around a factor of four higher peak (and Linpack) performance than HPCx. Of course, in general real applications can only utilise a fraction of the total cycles available [1].

3 Synthetic Benchmark Results

3.1 STREAMS

Figure 1 shows the results of the STREAMS Triad benchmark [6] which gives a measure of the memory bandwidth performance. To produce these results, the STREAMS code was modified to loop over multiple array sizes.

For each system, the cache structure can be seen through the discontinuities in bandwidth with increasing array size. At low array sizes, the HECToR L1 cache is seen to outperform the corresponding HPCx L1 cache. This can be attributed to the Streaming SIMD extensions on HECToR, which allow double element "vectors" to be loaded from L1 to the registers (or stored from the registers to L1), instead of single element scalars. At those arrays of intermediate size which fit into the lower levels of cache, HPCx has a clear advantage over HECToR. HECToR has a slight advantage for larger arrays which do not fit into the cache and are therefore streamed from main memory.

The benchmarks were repeated while only utilising one core per dual core chip on both systems. Any difference in these results compared with the fully utilised results indicate that the memory bandwidth cannot sustain both cores on the chip. There is no difference on L1 cache, as expected since the L1 cache is private to the core. On HPCx, there is a slight difference on L2 and a significant difference on L3 and main memory. The discontinuities seen within the range of the HECToR L2 cache (which is private to the core) can be attributed to a compiler issue, and disappear when the Streaming SIMD extensions (mentioned above) are disabled. On main memory, the differences between the fully utilised and under utilised results are more pronounced for HECToR than for HPCx. This indicates that the HPCx memory system can better sustain the cores than HECToR.

	HPCx	HECToR
Chip	IBM Power5 (dual core)	AMD Opteron (dual core)
Clock	1.5 GHz	2.8GHz
FPUs	2 FMA	1M, 1A
Peak Perf/core	6.0 GFlop/s	5.6 GFlop/s
cores	2560	11328
Peak Perf	15.4 TFLOP/s	63.4 TFLOP/s
Linpack	12.9 TFLOP/s	54.6 TFLOP/s

Table 1: A comparison of the processing resources available on HPCx and HECToR.

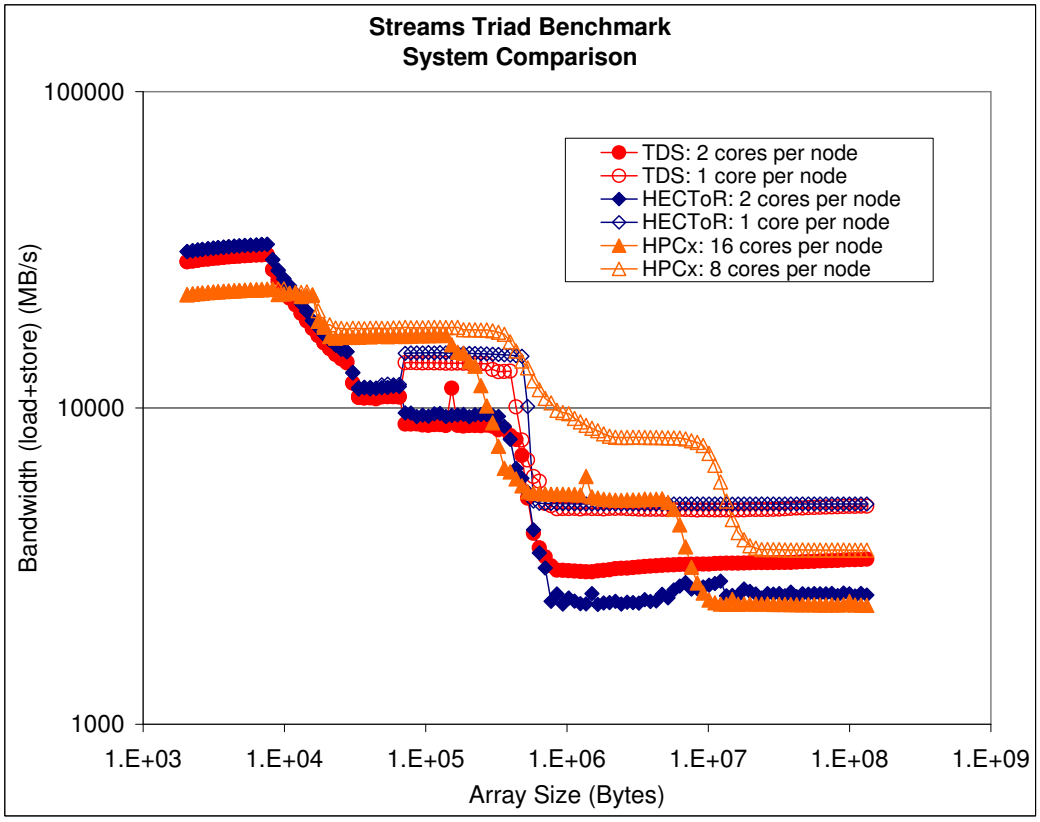


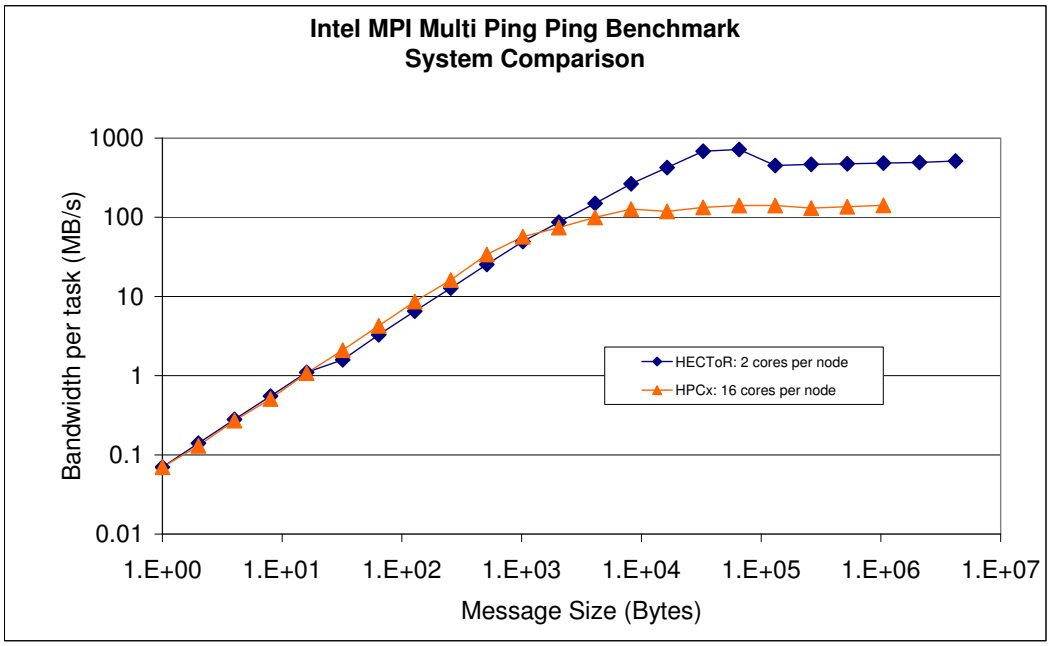
Figure 1: The dependence of the bandwidth on the array size for the per chip.

Results are also presented for the HECToR Test and Development system (TDS) which has an identical hardware and software to the main HECToR machine except for slightly slower cores (2.6GHz cf 2.8GHz), and less memory (4GB/node cf 6GB/node). At first glance it seems confusing that the TDS memory bandwidth is higher than the main HECToR system, despite the lower clock rate. However, this can be explained by the fact that the asymmetrical memory layout on the main system (one 4GB bank plus one 2GB bank), does not allow full interleaving, restricting the memory bandwidth. The TDS has a 2GB + 2GB memory layout which allows full interleaving.

3.2 INTEL MPI Benchmarks

The Intel MPI Benchmarks [16] give a measure of MPI communication performance. The top of Figure 2 shows results from the Multi PingPing benchmark. This is based on the PingPing benchmark, where pairs of processes simultaneously send messages to each other, but comprises multiple pairs of processes such that the resource is fully utilised to give a more realistic simulation of application behaviour. Each pair is split between nodes, forcing all communication to go through the network rather than shared memory. It can be seen that at small message sizes there is no significant difference between the systems, but HECToR performs better at large message sizes: the HECToR toroidal network has a higher saturation point than the HPCx switch.

The bottom of Figure 2 shows the results of the AlltoAll benchmark, where each process sends to each other process. For small message sizes (<100 bytes) the HPCx network has a slight edge over HECToR. For message sizes larger than 100 bytes the HECToR network outperforms that of HPCx. Further details on this work are available in [3].



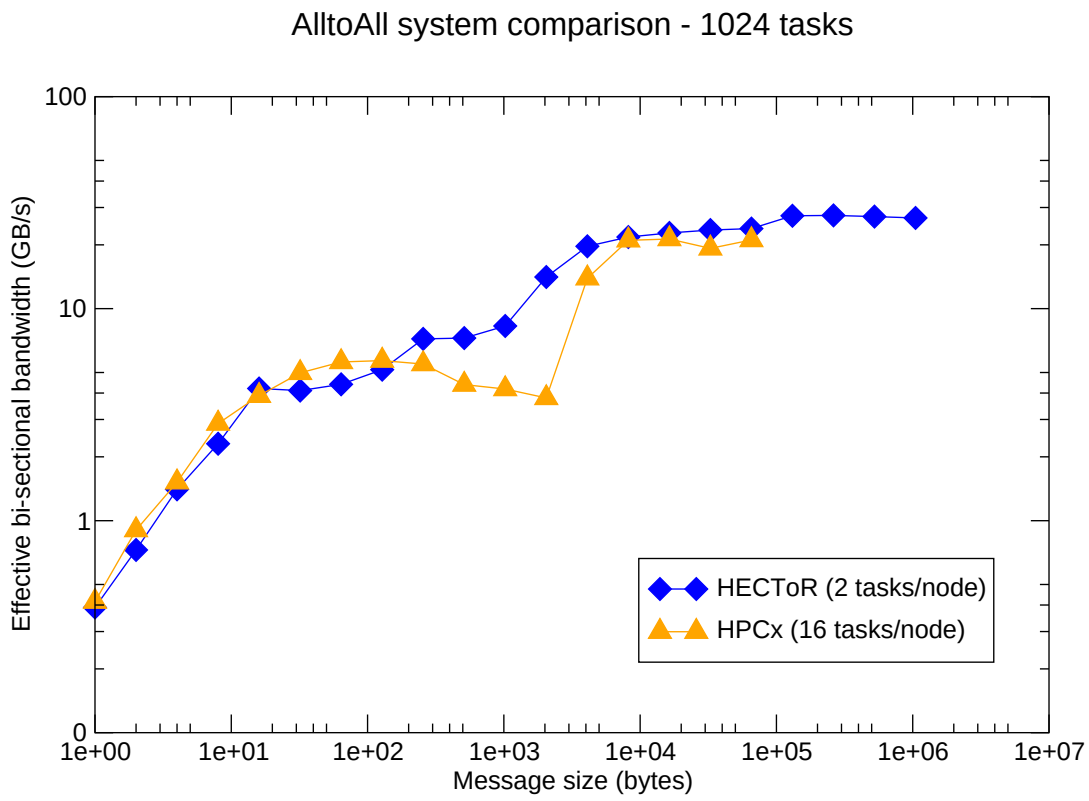


Figure 2: Top: the dependence on the bandwidth (per task) of the message size for the Intel MPI Multi Ping-Ping Benchmark. Bottom: the dependence of the effective bisectional bandwidth on the message size for the Intel MPI AlltoAll Benchmark. Diamonds and triangles denote HECToR and HPCx results respectively.

4 Application Benchmark Results

4.1 Fluid Dynamics: PDNS3D

The PDNS3D (formerly known as PCHAN) code is designed to simulate the flowing of fluids to study turbulence [17]. This is achieved by solving the governing equations of motion from first principles.

Since HECToR is a new system, it is of value to explore the range of compiler optimisations available. The bottom of Figure 3 compares a variety of benchmark times gained while varying the compilation options to the Portland Group fortran compiler [18]. The `-O<n>` macros perform a range of optimisations, with the level of optimisation increasing with `<n>`, and `-fast` specifies generally optimal flags for the target platform. The `-Mipa=fast,inline` flag directs the compiler to, using inter procedural analysis, inline suitable routines. It can be seen that, for this code, there is advantage to be gained with the `-O2` optimisation level (or higher), and further performance improvement is observed when the `-fast` macro is invoked. Inlining has no effect in this case. These results may vary from code to code, and users should benchmark their own code. The `-fast -O3` option was used in the below PDNS3D system comparison results, and in general for the other applications in this paper.

The top of Figure 3 compares HECToR to HPCx in terms of dependence of the "total core time", i.e. the number of cores multiplied by benchmark time, on the number of cores used. This metric indicates the cost of the job (i.e. low is good), and ideal scaling would be represented by a horizontal line. The negative gradient of the HPCx results indicate better than ideal scaling, and this can be explained by the fact that the problem size per core is becoming more suited to the Power5 cache structure as the number of cores increases. The main point to take from this graph is that HECToR performs almost a factor of 2 worse than HPCx for this code. It is strongly suspected that this is due to a memory bandwidth bottleneck. PDNS3D streams data from memory, and is thus very sensitive to the memory bandwidth.

Figure 4 shows results including runs for which the nodes have been underutilised. For each of HPCx and HECToR, runs were repeated but this time only using 1 core per dual core chip, thus making available all the chip's memory bandwidth to that core (but obviously having to utilise twice the resource for the same size of run). While the HPCx results improve by around a factor of 1.2, the HECToR results improve by a factor of 1.6. This is consistent with the results from the STREAMS benchmark (Section 3.1), and indicates that the HPCx cores can cope better with data streaming to both cores in a chip, for this problem at least.

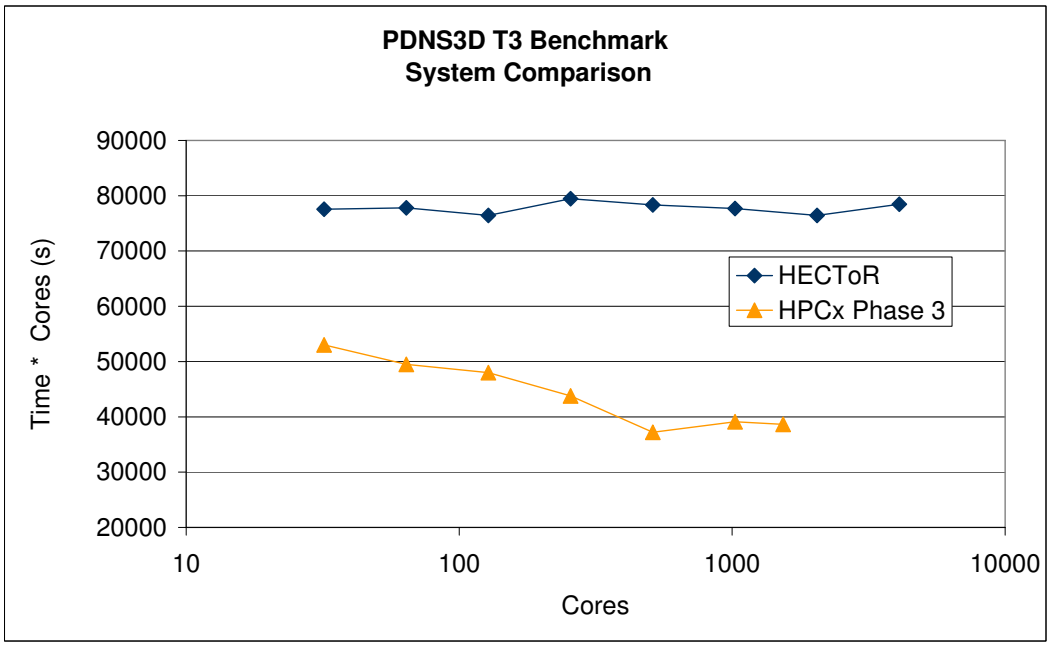
These results are reinforced by the results on the HECToR TDS machine. As is observed for STREAMS in Section 3.1, this memory bandwidth sensitive application is performing better on the lower spec TDS due to the fact that the memory setup on the main machine is asymmetric. This memory bandwidth restriction on the main system may contribute to the performance difference between HPCx and HECToR.

An optimised version of PDNS3D has recently been made available. The main kernels have been restructured and as a result the code is less sensitive to the memory bandwidth. Figure 5 demonstrates the dramatic effect this has on the HECToR results: these are now in line with those from HPCx which are not significantly affected by the optimisation. The effect of only utilising one core per chip on HECToR is also seen to be reduced in magnitude for the optimised code.

4.2 Fluid Dynamics: Ludwig

Ludwig is a lattice Boltzmann code for solving the incompressible Navier-Stokes equations and is used to study complex fluids. In the examples presented in this paper, we consider a binary fluid mixture both alone (i.e. no shear flow) and with shear flow included. Further details of this fluid mixture are given by [11]. Figure 6 shows the performance of the fluid only and fluid including shear flow for both HPCx and HECToR for a range of processor counts. Also shown are HPCx results where Simultaneous Multithreading (SMT) [12] has been activated, which is known to offer performance advantages for this type of application on HPCx [2].

The Ludwig code uses a regular domain decomposition and requires boundary exchanges between subdomains at each iteration. For the shear flow problem, an extra episode of communication is required in one



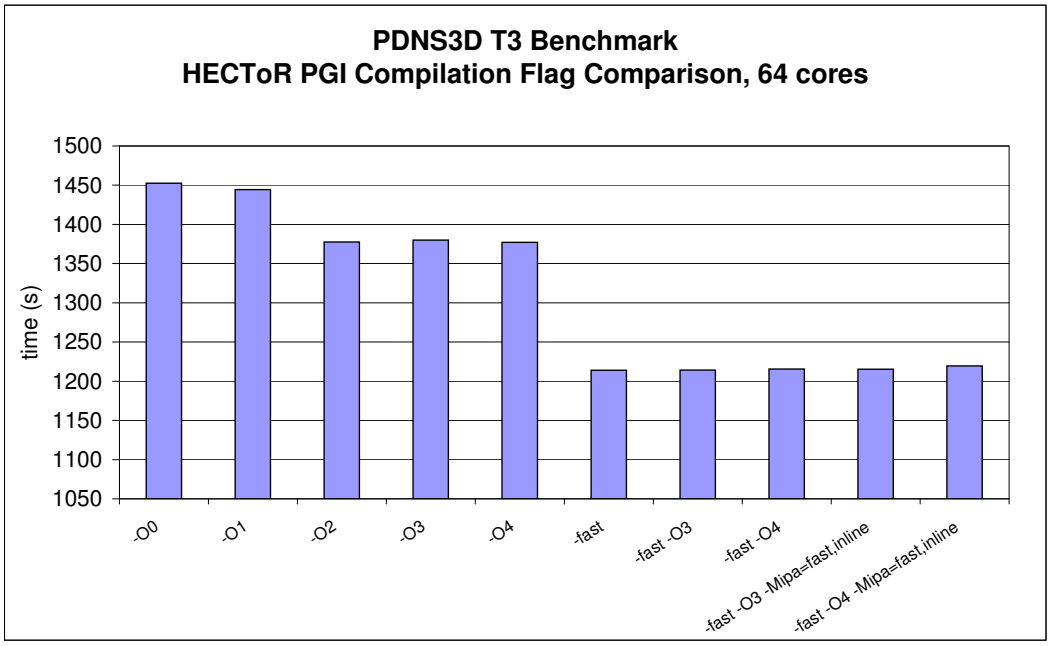
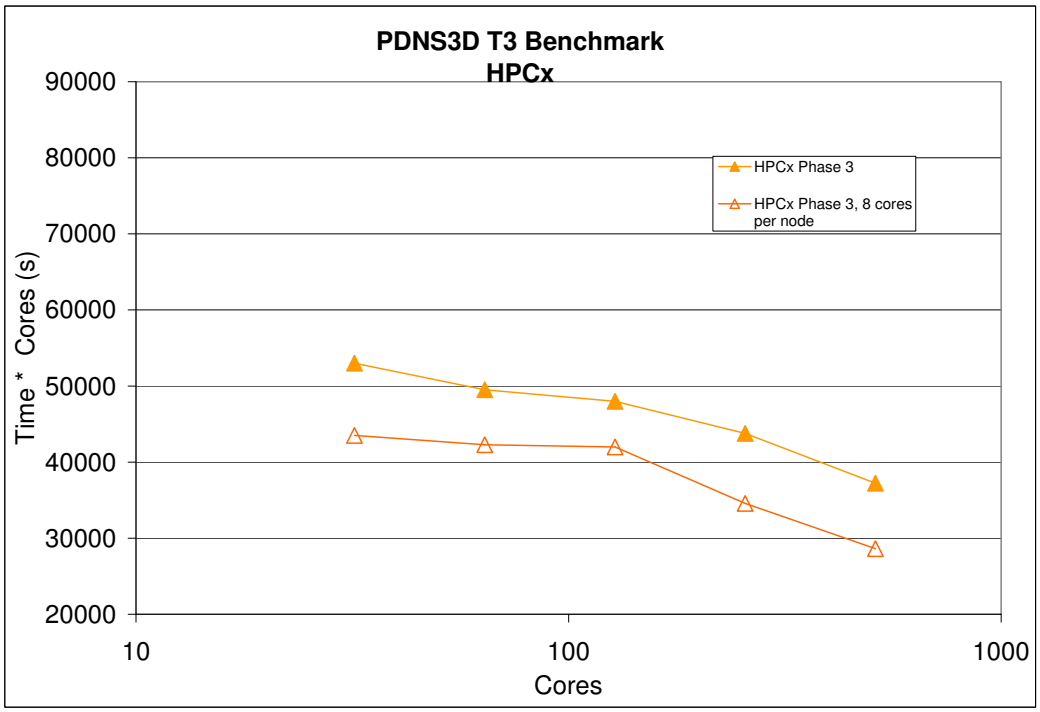


Figure 3: Top: the dependence of the total core time on the number of cores for the PDNS3D T3 benchmark, where diamonds and triangles denote HECToR and HPCx results respectively. Bottom: the run time dependence on the compilation option used for a 64 core HECToR PDNS3D T3 benchmark.



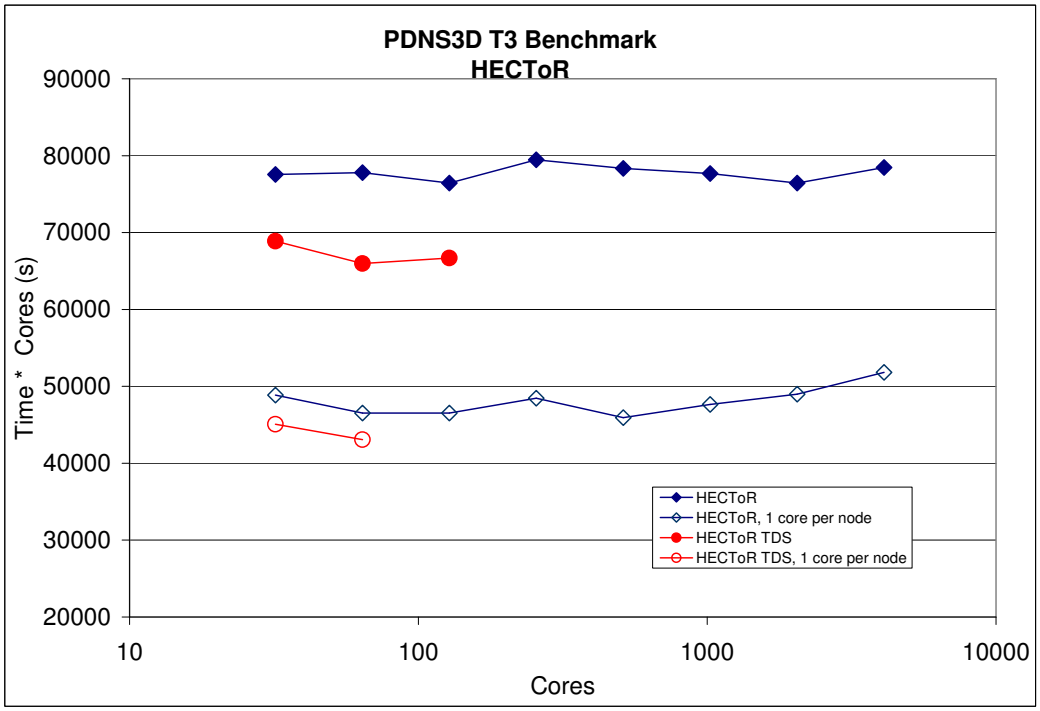


Figure 4: The dependence of the total core time on the number of cores for the PDNS3D T3 benchmark. Fully utilised HPCx and HECToR nodes (closed symbols) are compared, on the top and bottom plots respectively, to runs where only a single core has been utilised per chip (open symbols). Also included are results from the HECToR TDS system.

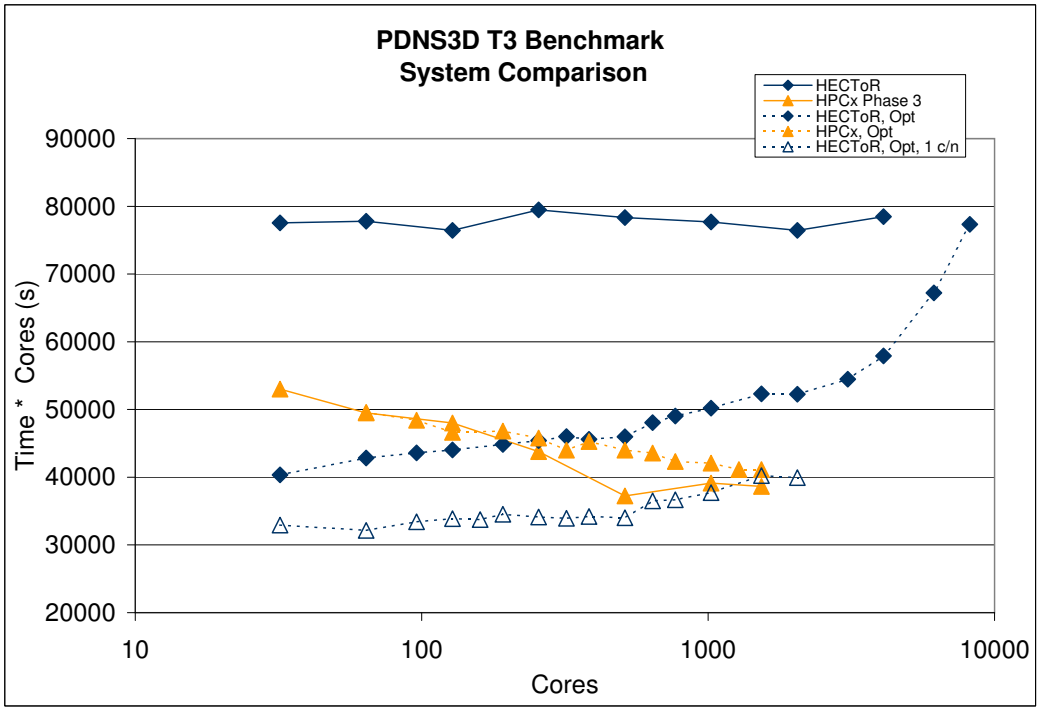


Figure 5: The dependence of the total core time on the number of cores for the PDNS3D T3 benchmark, where results from the new optimised version of the code are included (broken lines). Diamonds and triangles denote HECToR and HPCx results respectively. Runs where only a single core per chip has been utilised on HECToR are included (open symbols).

coordinate direction to implement the shear boundary conditions. This extra communication is apparent from Figure 6 (compare the solid and dashed lines).

The lattice Boltzmann algorithm essentially consists of two basic stages: a floating point intensive *collision* stage which introduces the appropriate physics, and a *propagation* stage which involves only memory movement. Computationally, the latter is then purely a probe of memory bandwidth. The relative strength of memory bandwidth on each machine means that propagation is about 15% of the total time on HECToR and 5-10% of the total time on HPCx.

The absolute performance of Ludwig is also reasonable: about 15% of peak serial performance can be obtained on HPCx [1]. The local nature of the lattice Boltzmann algorithm means that scaling is generally good. In practice, SMT would always be employed on HPCx for these calculations which means that HECToR performs up to 1.33 times faster than HPCx.

4.3 Nuclear Fusion: CENTORI and GS2

CENTORI and GS2 are codes which simulate plasma physics inside tokamak nuclear fusion devices. CENTORI simulates the plasma fluid flow, and was developed by UKAEA Fusion in collaboration with EPCC. GS2 [5] is an open source code which focuses on the physics at a smaller scale: it performs gyrokinetic simulations of low-frequency turbulence in tokamaks.

From the top of Figure 7, it is seen that HECToR and HPCx perform very similarly for CENTORI, with the scaling slightly better on HECToR. The TDS is around 10% faster than the main HECToR system. These results indicate that this code is not as sensitive to the memory bandwidth as PDNS3D (see Section 4.1).

GS2 is seen, from the bottom of Figure 7, to scale incredibly well on both HECToR and HPCx. The

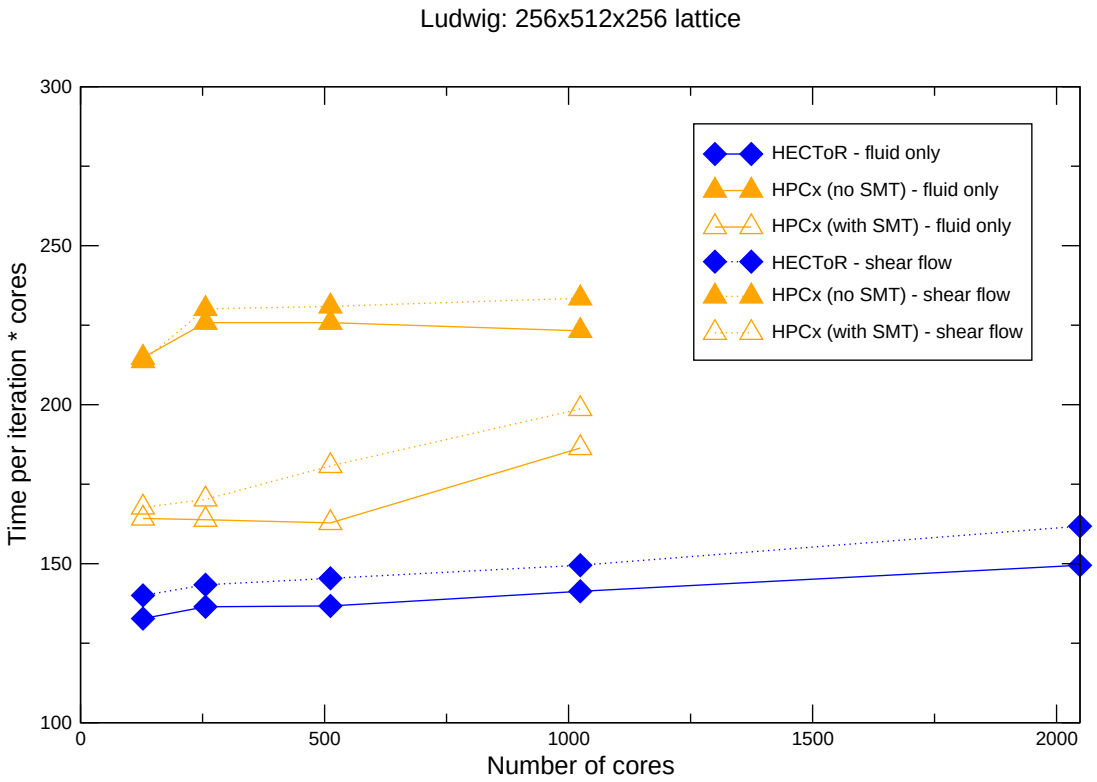


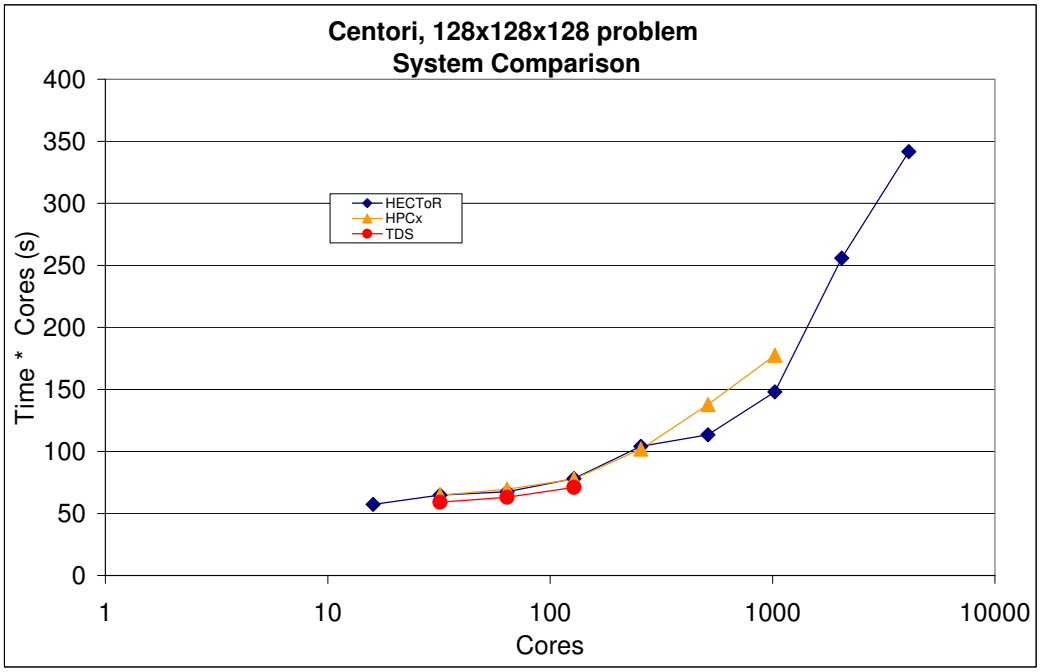
Figure 6: The dependence of total core time on the number of cores for a binary fluid mixture run on Ludwig. Solid lines represent a fluid mixture without shear flow and dashed lines represent represent the same fluid with shear flow included. The fluid has a lattice size of 256x512x256. The HECToR results are represented by diamonds and the HPCx results with and without SMT are represented by the filled and open triangles respectively.

superscaling observed can likely be explained by cache effects: the problem may become more suited to the cache structure as the number of cores increases. The effect is more pronounced on HPCx, which can possibly be attributed to the L3 cache. HECToR outperforms HPCx at low core counts, but HPCx has the advantage at higher core counts. Similarly to PDNS3D (Section 4.1), the TDS is outperforming the main HECToR system when using two cores per chip. When only utilising one core per chip on HECToR this difference vanishes as is seen for the STREAMS benchmark (see Section 3.1). This indicates that this code is sensitive to the memory bandwidth, and the memory bandwidth to the HECToR chips is not able to sustain both cores. It should be noted that the advantage gained when only utilising one of the two cores per chip is not large enough to justify leaving the second core empty: users are charged for the total number of nodes used regardless of the number of cores utilised.

4.4 POLCOMS

The Proudman Oceanographic Laboratory Coastal Ocean Modelling System (POLCOMS) has been developed to tackle multi-disciplinary studies in coastal/shelf environments and optimised to make use of high-performance parallel computers [19]. In order to improve simulations of marine processes, we need accurate representation of eddies, fronts and other regions of steep gradients. The current generation of models includes simulations which cover large regions of the continental shelf at approximately 1km resolution.

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 advection routines using the Piecewise Parabolic Method are used to



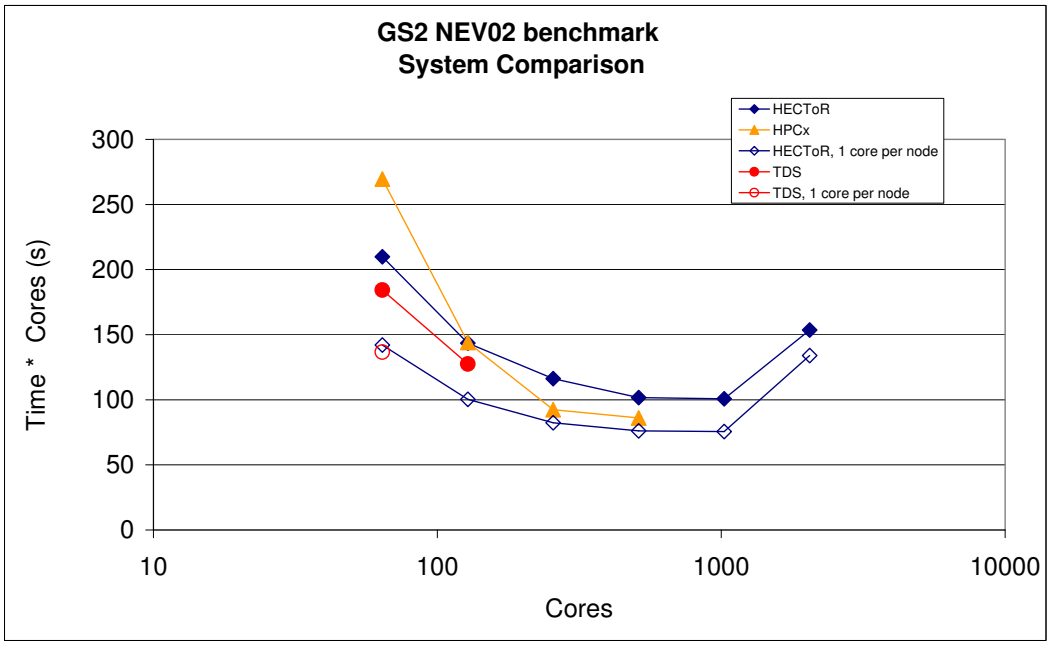


Figure 7: The dependence of the total core time on the number of cores for CENTORI (top) and GS2 (bottom) benchmarks. Diamonds, triangles and circles denote HECToR, HPCx and TDS results respectively. The GS2 plot includes runs where only a single core has been utilised per chip (open symbols).

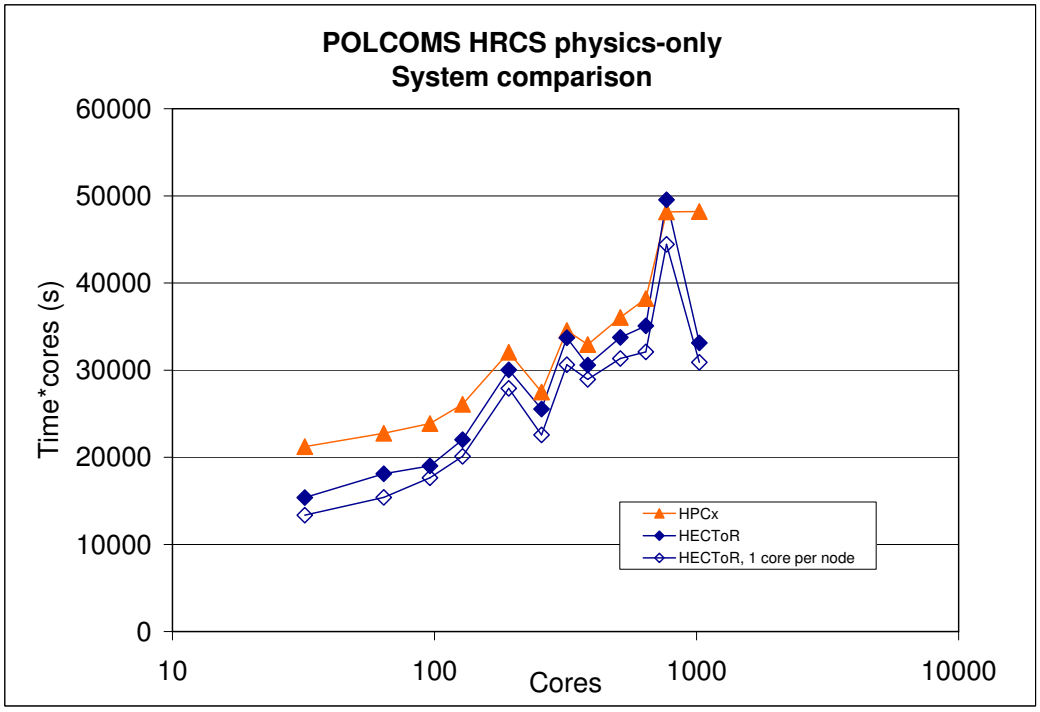


Figure 8: the dependence of the total core time on the number of cores for the POLCOMS HRCS physics-only code, where diamonds and triangles denote HECToR and HPCx results respectively. Runs where only a single core per chip has been utilised on HECToR are included (open symbols).

ensure strong frontal gradients. Vertical mixing is through turbulence closure (Mellor-Yamada level 2.5).

POLCOMS has been parallelised using grid partitioning in the two horizontal dimensions. A recursive bi-section algorithm is employed which balances the number of active sea-points across the processors in order to optimise the load balance. Communications are managed by the well-known method of maintaining halo regions surrounding each sub-domain. The vertical dimension is kept local to each processor as this has performance advantages for processes which take place within the water column, such as convection, downwelling and vertical diffusion. As with many finite difference codes the computational load is heavy on accesses to main memory with limited cache re-use.

The benchmark case is the High-Resolution Continental Shelf (HRCS) model. This is a high-resolution grid of size 1001 x 801 x 34 points representing a resolution of 1/40 degree x 1/60 degree covering the north-west European shelf seas. This is a research model used for investigations of the circulation of the north-west European continental shelf [4].

The jumpy results in Figure 8 indicate that the performance is particularly sensitive to the decomposition (which in turn is dependent on the number of cores), but HECToR is seen to have a significant performance advantage over HPCx. The effect of only utilising one core per chip on HECToR is not so pronounced as for some of the codes in this study. These results indicate that this code is not particularly sensitive to the memory bandwidth.

4.5 Molecular Dynamics

Molecular dynamics simulations account for a large fraction of the HPCx service allocation, and this may well also be the case on HECToR. In this section, we present performance results from the DL_POLY, LAMMPS, GROMACS, NAMD and AMBER applications.

DL_POLY is a classical molecular dynamics code which can be used to simulate systems with very large

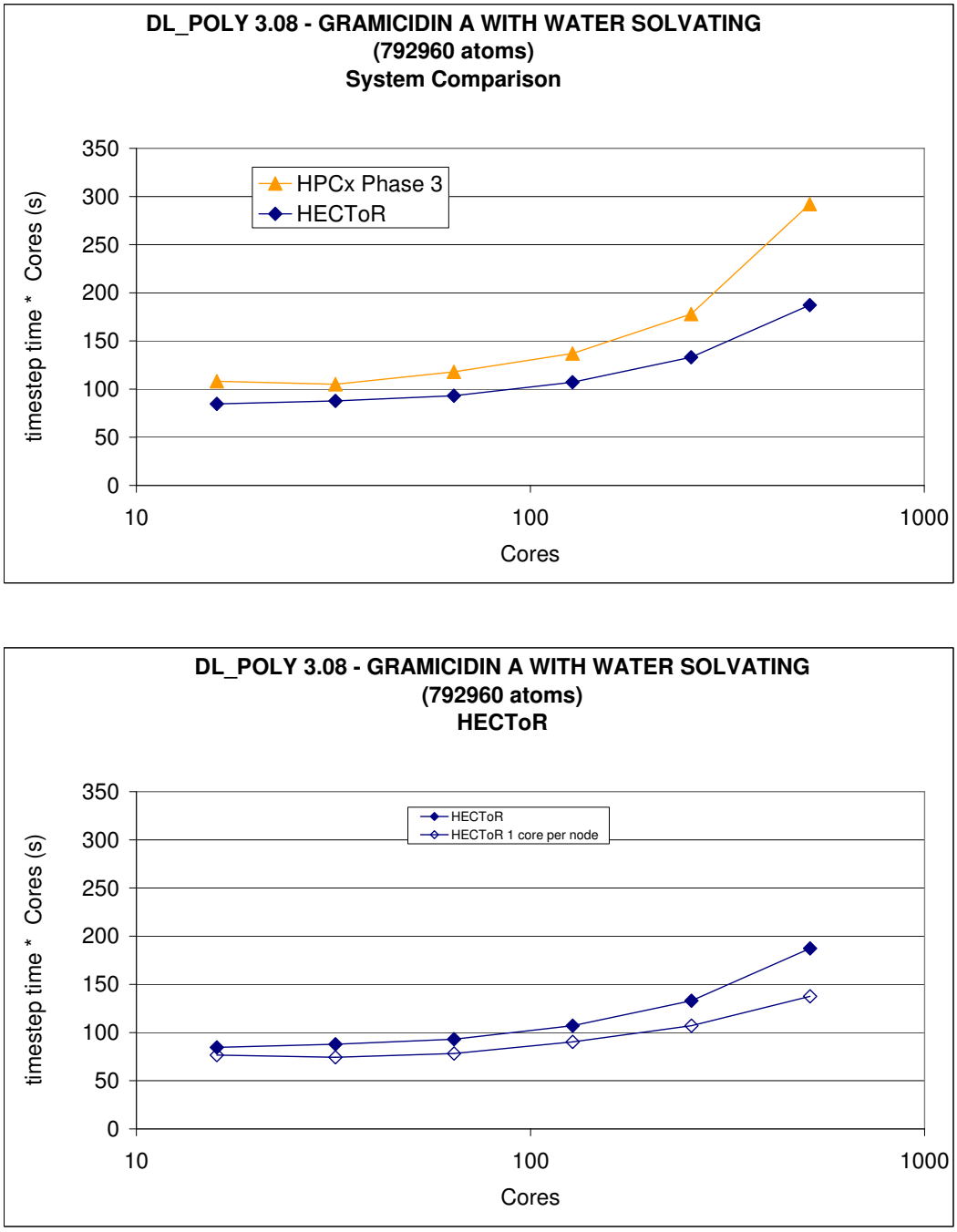
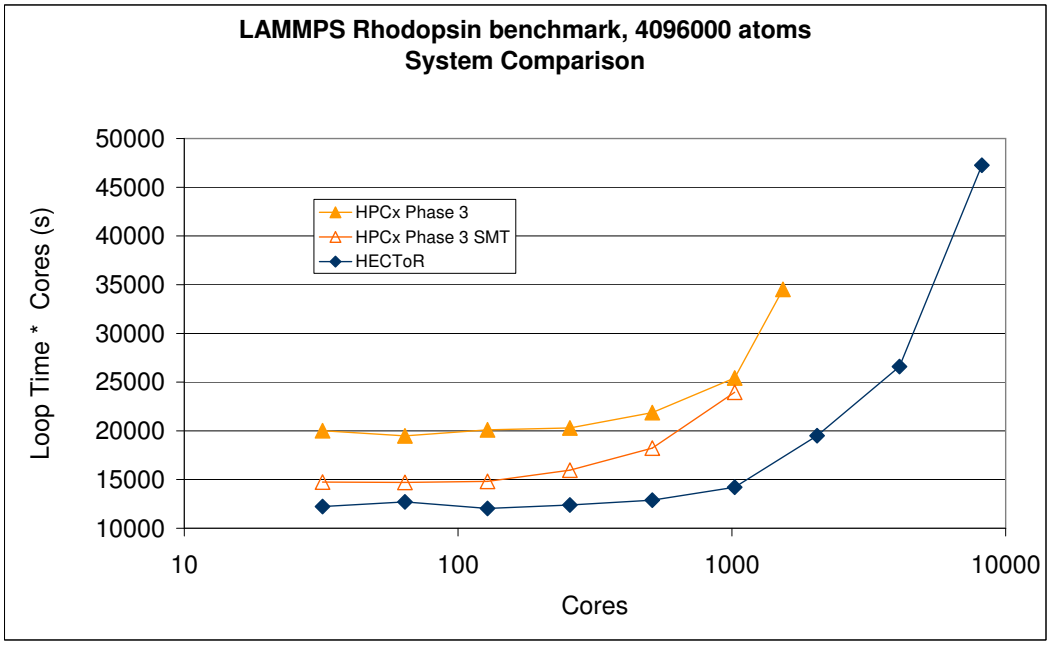


Figure 9: Top: the dependence of the total core time on the number of cores for the DL_POLY Gramicidin benchmark, where diamonds and triangles denote HECToR and HPCx results respectively. Bottom: a comparison of runs on HECToR using fully populated nodes (closed symbols) to those where only a single core has been utilised per chip (open symbols).



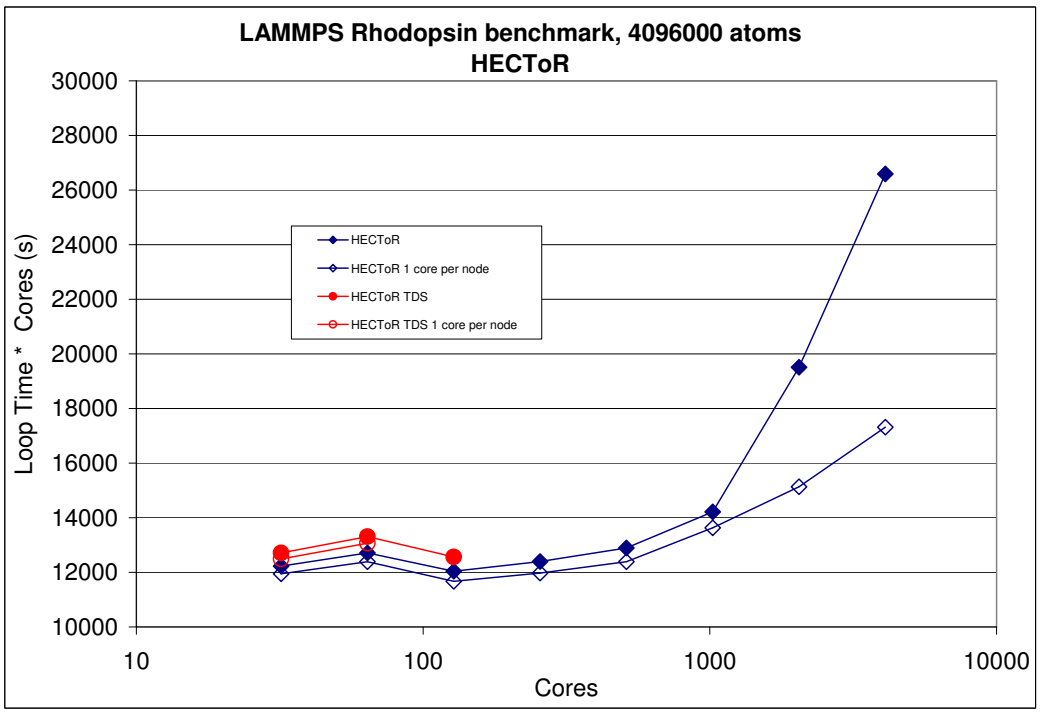


Figure 10: Top: The dependence of the total core time on the number of cores for the LAMMPS Rhodopsin benchmark, where diamonds and triangles denote HECToR and HPCx results respectively. The Open triangles denote the utilisation of SMT on HPCx. Bottom: A comparison of runs on HECToR (diamonds) and TDS (circles) systems using fully populated nodes (closed symbols) to those where only a single core has been utilised per chip (open symbols).

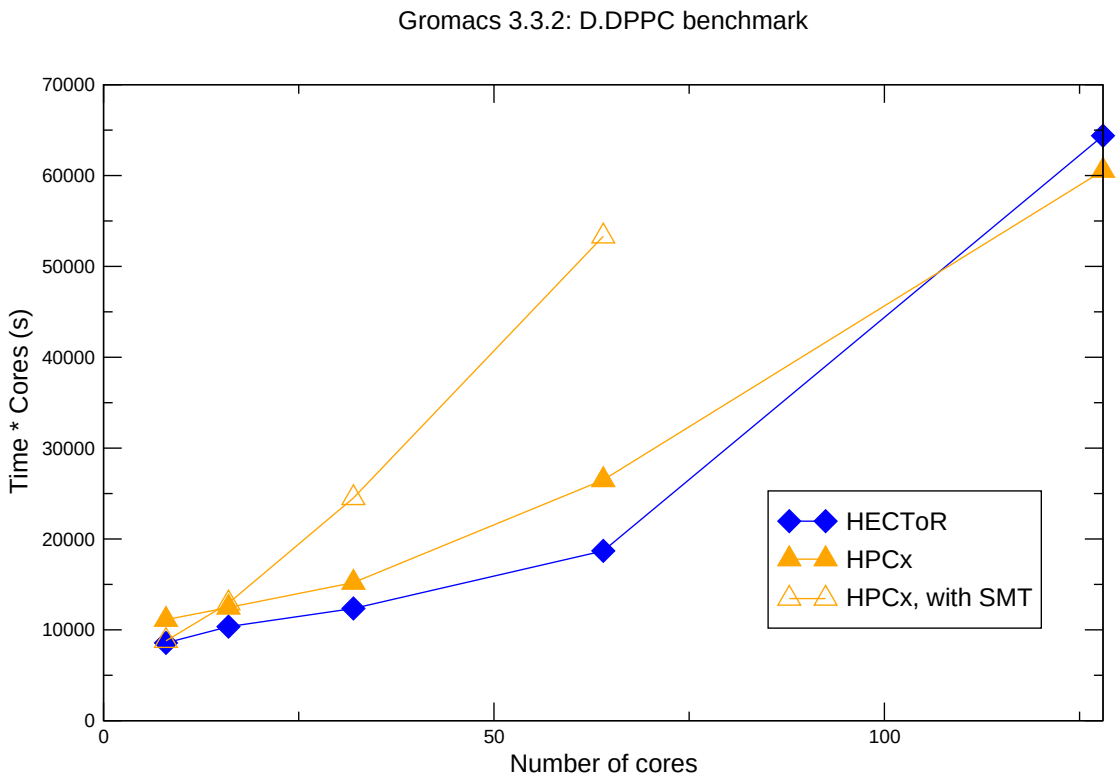


Figure 11: Time taken to run the DPPC benchmark on GROMACS 3.3.2 plotted against the number of cores where diamonds and triangles denote the HECToR and HPCx results respectively. The triangles denote the utilisation of SMT on HPCx.

numbers of atoms [10]. Here, DL_POLY3, which is parallelised by domain decomposition and is suitable for large numbers of cores (as opposed to DL_POLY2 which uses a Replicated Data strategy and is suitable for of order 100 cores), was run with a system of gramicidin molecules in water containing a total of 792,960 atoms.

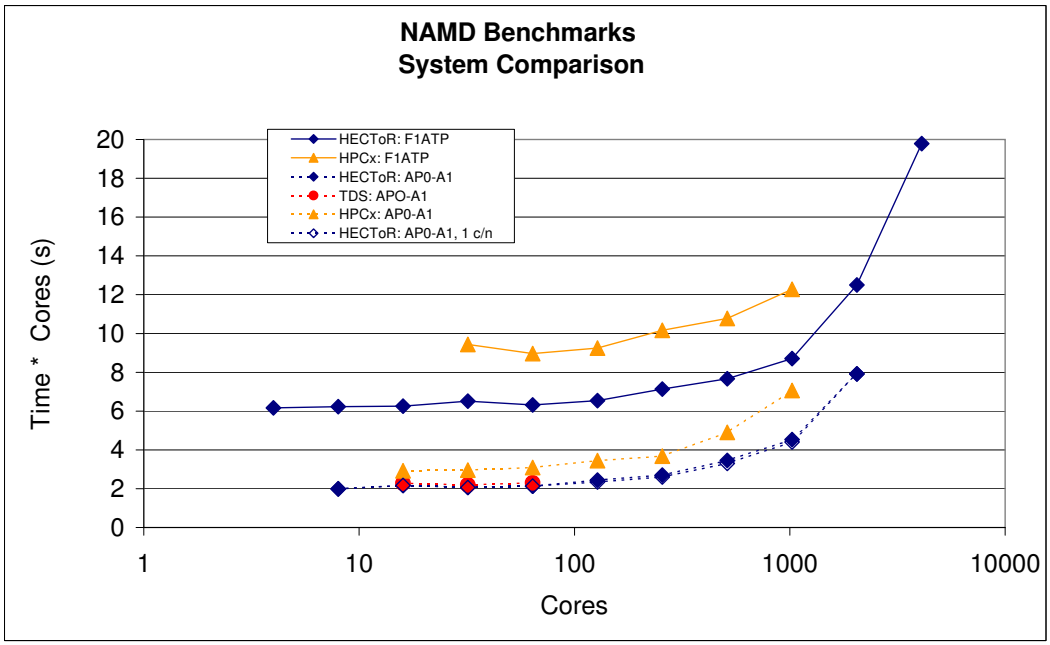
LAMMPS is a molecular dynamics package which solves classical physics equations and is able to simulate a wide range of materials including atomic, molecular, metallic and hybrid systems [9]. Here a rhodopsin system, with 4,096,000 atoms, is benchmarked. The system consists of a rhodopsin protein solvated in a lipid bilayer with a CHARMM [7] force field applied and is created by scaling up (i.e. replicating) the 32,000 atom benchmark supplied with the code. The replication factors used to generate the 4,096,000 atom system were respectively 8, 4, 4 in the X, Y and Z directions.

GROMACS is a general purpose molecular dynamics package which solves classical (Newtonian) equations of motion for systems containing hundreds to millions of particles. It can be used to simulate a wide range of materials but is designed primarily for use with biochemical molecules, e.g. proteins and lipids [13]. Here one of the biochemical systems supplied with the code, DPPC, is benchmarked using GROMACS version 3.3.2. The DPPC system consists of a phospholipid membrane with 1024 dipalmitoylphosphatidylcholine (DPPC) lipids in a bilayer configuration. Each of the lipids has 23 water molecules associated with it. The entire system contains a total of 121,856 atoms.

The NAMD code is designed to simulate biomolecular systems such as proteins [8]. Results are given for the ApoA1 (92,224 atom) and F1ATP (327,506 atom) benchmarks.

AMBER is a suite of molecular dynamics applications, in which PMEMD (Particle mesh Ewald molecular dynamics) is the application which demonstrates the best scaling. Here, results are given for the PMEMD Factor IX benchmark, which involves 90906 atoms.

The top of Figure 9 shows that DL_POLY is performing and scaling significantly better on HECToR than



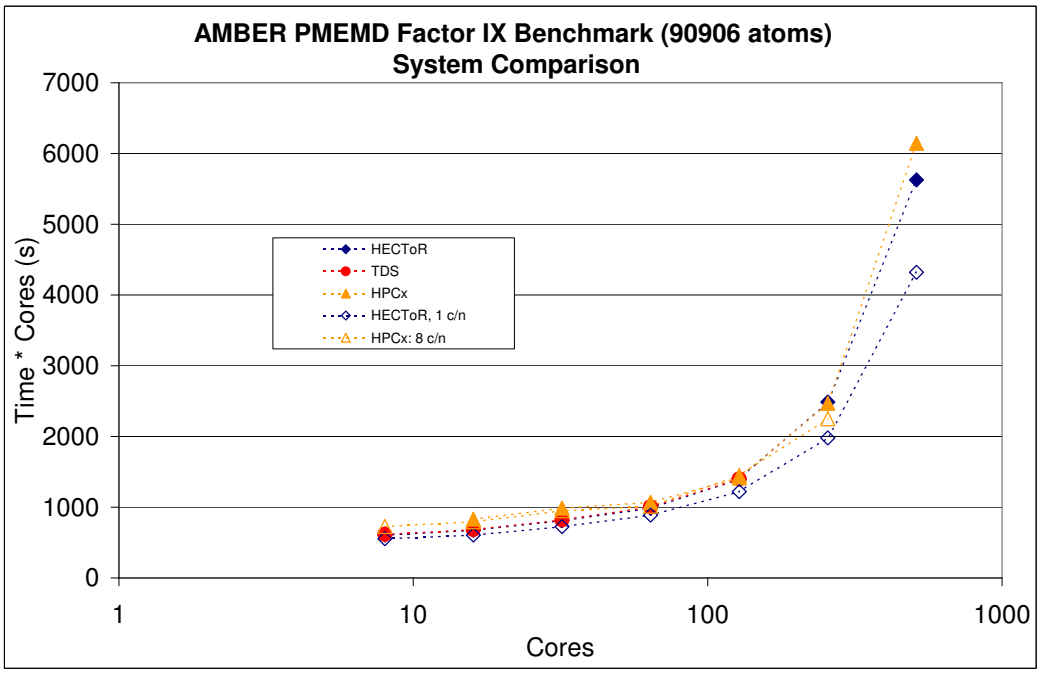


Figure 12: The dependence of the total core time on the number of cores for the NAMD F1ATP and AP0-A1 (top) and AMBER PMEMD Factor 1X (bottom) benchmarks, where diamonds, triangles and circles denote HECToR, HPCx and TDS results respectively. Included are runs where only a single core has been utilised per chip (open symbols).

HPCx. This indicates that this code is more suited to the HECToR cores and interconnect, and does not suffer from the memory bandwidth sensitivity experienced by several of the previously studied applications. This is not surprising since molecular dynamics applications are typically characterised by strided memory accesses, and are hence more sensitive to the memory latency than bandwidth. The bottom of Figure 9 shows that, when only utilising 1 core per chip on HECToR, there is no significant difference in the performance, from full utilisation of cores, at low core numbers, but the scaling is significantly better. This could be because with only one core per chip there is no contention for the network resources.

The LAMMPS results (Figure 10) show the same characteristics as for DL_POLY, but with HECToR having more of an advantage. On HPCx, the use of simultaneous multithreading is clearly advantageous due to the type of application [2]. Comparing the results obtained on the main HECToR system with those from the TDS (bottom of Figure 10) we see that the non symmetric memory distribution of HECToR is not an issue for LAMMPS.

The GROMACS results (Figure 11) show similar characteristics to the other molecular dynamics codes. The scaling is found to be poor on both systems with little benefit gained from using more than 64 processors. Due to the poor scaling, simultaneous multithreading is only beneficial up to 32 cores on HPCx.

The NAMD and PMEMD results are presented in Figure 12, and show similar trends to the above Molecular Dynamics applications, with the performance differences between HPCx and HECToR dependent on the specific benchmark. NAMD on HECToR is particularly insensitive to only utilising one core per chip.

5 Conclusions

A number of synthetic and application benchmarks were used to compare the performance of the two current UK academic supercomputing services: HPCx and HECToR.

The synthetic STREAMS benchmark showed that the memory bandwidth on both systems cannot sustain both cores on a chip, with HECToR being more adversely affected. Comparing the HECToR and HPCx results depends on the array size, with HPCx having the advantage over most of the cache structure. The synthetic Intel MPI benchmarks indicate that the HECToR network has the advantage.

On a core by core basis, the application benchmark results were dependent on the type of application: some favoured HECToR while others favoured HPCx: on the whole there is not much difference between the systems. However, HECToR is a much larger system with more cores and a more scalable interconnect. It should also be noted that several of these applications may have received optimisation tailored to HPCx over the last few years, and there may still be room for improvements on HECToR. The application scaling was seen to be better on HECToR at large core counts.

It was observed that the memory bandwidth cannot sustain fully populated nodes on either system. This is a general problem for HPC systems these days, but here it was observed that, especially with the current asymmetric memory setup, the problem is significantly worse on HECToR than HPCx.

6 Acknowledgements

We thank Colin Roach (UKAEA) and Bill Dorland (University of Maryland) for assistance in porting GS2 to HECToR. Colin also supplied us with the NEV02 benchmark. Sarah Anderson (Cray), Jim Phillips (UIUC) and Ghengbin Zheng (UIUC) provided support in installing NAMD on HECToR.

7 About the Authors

Fiona Reid is an Applications Consultant at EPCC and can be reached at f.reid@epcc.ed.ac.uk. Mike Ashworth leads the Advanced Research Computing Group at STFC Daresbury Laboratory and can be reached at m.ashworth@dl.ac.uk. Thomas Edwards is a PhD student at EPCC and can be reached at t.edwards@epcc.ed.ac.uk. Alan Gray is a Principal Consultant at EPCC and can be reached at a.gray@epcc.ed.ac.uk. Joachim Hein is a Computing Architect at EPCC and can be reached at j.hein@epcc.ed.ac.uk. Peter Knight

is a Computational Physicist in the Theory and Modelling Department at EURATOM/UKAEA and can be reached at peter.knight@ukaea.org.uk. Alan Simpson is the Service Director of the HPCX consortium and the Technical Director of EPCC and can be reached at a.simpson@epcc.ed.ac.uk. Kevin Stratford is a Principal Consultant at EPCC and can be reached at k.stratford@epcc.ed.ac.uk. Michele Weiland is an Applications Consultant at EPCC and can be reached at m.weiland@epcc.ed.ac.uk.

References

[1] Bull J.M. 2007. Single Node Performance Analysis of Applications on HPCx. *HPCx Technical Report* **0703**, http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0703.pdf

[2] Gray A., *et al.*, 2006. An Investigation of Simultaneous Multithreading on HPCx. *HPCx Technical Report* **0604**, http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0604.pdf

[3] Hein, J., *et-al* 2008. Parallel 3D-FFTs for Multi-processing Core Nodes on a Meshed Communication Network. *Proceedings of the 2008 Cray User Group (CUG) Meeting, Helsinki, Finland.*

[4] Holt J., Proctor R., and Ashworth M. 2007. High-Resolution Modelling of the European Shelf Seas using POLCOMS. *Capability Computing Article*, Issue **10**, http://www.hpcx.ac.uk/about/newsletter/HPCxNews10.pdf

[5] GS2 Homepage, http://gs2.sourceforge.net; Joiner, N., Applegate, D., Cowley, S.C., Dorland, W., and Roach, C.M. 2006. Electron Temperature Gradient Driven transport in a MAST H-mode Plasma, *Plasma Phys. Control. Fusion* **48** 685-697.; Roach, C.M., Applegate, D.J., Connor, J.W., Cowley, S.C., Dorland, W.D., Hastie, R.J., Joiner, N., Saarelma, S., Schekochihin, A.A., Akers R.J., Brickley, C., Field, A.R., and Valovic, M. 2005. Microinstability Physics as Illuminated in the Spherical Tokamak, *Plasma Physics and Controlled Fusion* **47**, B323.

[6] McCalpin, J.D. 1995. A Survey of Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Technical Committee on Computer Architecture (TCCA) Newsletter.*

[7] MacKerell, Jr., A. D., Bashford, D., Bellott, M., Dunbrack Jr., R.L., Evanseck, J.D., Field, M.J., Fischer, S., Gao, J., Guo, H., Ha, S., Joseph-McCarthy, D., Kuchnir, L., Kuczera, K., Lau, F.T.K., Mattos, C., Michnick, S., Ngo, T., Nguyen, D.T., Prodhom, B., Reiher, III, W.E., Roux, B., Schlenkrich, M., Smith, J.C., Stote, R., Straub, J., Watanabe, M. Wiorkiewicz-Kuczera, J., Yin, D., and Karplus, M. 1998. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *Journal of Physical Chemistry B*, **102**, 3586–3616.

[8] Phillips, J., Zheng, G., Kumar S. & Kalé L. 2002. NAMD Biomolecular Simulation on Thousands of Processors. In *Proceedings of the SC2002 Conference.* IEEE Press; http://www.ks.uiuc.edu/Research/namd/

[9] Plimpton, S. 1995. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comp. Phys.* **117**, 1–19; http://www.cs.sandia.gov/ sjplimp/lammps.html

[10] Smith, W., and Forester, T. 1996. DL_POLY: A General Purpose Parallel Molecular Dynamics Simulation Package. *J. Molec. Graphics* **14**, 136; Smith, W., Yong, C., and Rodger, M. 2002. DL_POLY: Applications to Molecular Simulation. *Molecular Simulation* **28**, 385; Todorov, I., and Smith, W. 2004. DL_POLY_3: The CCP5 National UK Code for Molecular Dynamics Simulations. *Phil. Trans. R. Soc. Lond. A* **362**, 1835; http://www.cse.clrc.ac.uk/msi/software/DL_POLY/

[11] Stratford, K., Desplat, J-C., Stansell, P. and Cates, M. 2007. Binary Fluids Under Steady Shear in Three Dimensions. *Phys. Rev. E.*, **76**, 030501(R).

[12] Tullsen, D.M., Eggers S.J., and Levy, H.M. 1995. Simultaneous Multithreading: Maximising on Chip Parallelism. *Proceedings of 22nd Annual Symposium on Computer Architecture*,

[13] GROMACS home page, http://www.gromacs.org/

[14] HECToR web page, http://www.hector.ac.uk

[15] HPCx web page, http://www.hpcx.ac.uk

[16] INTEL MPI benchmark code and documentation available from http://www.intel.com/software/products/cluster/downloads/mpi_benchmarks.htm

[17] PDNS3D web page, http://www.cse.scitech.ac.uk/arc/pdns3d.shtml

[18] PGI User's Guide, http://www.pgroup.com/doc/pgiug.pdf

[19] POLCOMS home page, http://www.pol.ac.uk/home/research/polcoms/