



# An Investigation of Simultaneous Multithreading on HPCx

A. Gray<sup>1</sup>, J. Hein<sup>1</sup>, M. Plummer<sup>2</sup>, A. Sunderland<sup>2</sup>, L. Smith<sup>1</sup>,  
A. Simpson<sup>1</sup>, A. Trew<sup>1</sup>

<sup>1</sup>*EPCC, The University of Edinburgh, James Clerk Maxwell Building,  
Mayfield Road, Edinburgh, EH9 3JZ, UK*

<sup>2</sup>*CCLRC Daresbury Laboratory, Warrington WA4 4AD, UK*

April 25, 2006

## Abstract

The use of Simultaneous Multithreading (SMT) on HPCx is investigated to determine if any performance improvement is obtainable for user applications. Various codes are benchmarked with and without the use of SMT. SMT is seen to boost the performance, at low and moderate processor counts, of the H2MOL application and the classical molecular dynamics codes NAMD, MDcask and DL\_POLY, with up to a factor of 1.4 improvement observed. No performance improvement is seen when using SMT with the materials science CASTEP code, the ScaLAPACK eigensolver routines or the STREAMS memory bandwidth benchmark.

**This is a Technical Report from the HPCx Consortium**

**© HPCx UoE Ltd 2005**

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

## 1 Introduction

In general, the execution of real applications results in the functional units of the processor remaining idle for a high percentage of the total number of processor cycles. Simultaneous Multithreading (SMT) aims to reduce this inefficiency by allowing multiple threads (e.g. MPI tasks or OpenMP threads) to simultaneously access the resources of one physical processor. The processors on HPCx support SMT with 2 threads per processor.

An overview of the architecture of HPCx is given in Section 2, SMT is discussed in Section 3 and the effects of SMT are investigated for various benchmark applications in Section 5.

## 2 Architecture

The current HPCx Phase 2a service utilises IBM Power5 technology. Featured are 96 IBM eServer 575 compute nodes, each containing 16 1.5GHz IBM Power5 64-bit RISC processors. The 16 processors in a compute node are packaged into 8 Dual Chip Modules (DCMs). Each DCM comprises a chip with 2 processing cores and an off chip L3 cache. Each core has private on chip 64KB L1 instruction cache and 32KB L1 data cache. Also on chip is a L2 combined data and instruction cache of 1.9MB shared between the two processors. The off chip L3 cache, again shared between the two processors, is 36MB. DCMs communicate via shared memory. Communication between nodes is provided by an IBM High Performance Switch (HPS). A total of 4 inter-node links are available: each node has 2 network adapters, each with 2 links.

## 3 SMT

The theoretical peak floating point performance of microprocessors has steadily risen in recent years. However, the actual performance obtained by real applications, relative to the theoretical peak, has dropped substantially. In other words, the number of processor cycles for which the floating point units are idle is rising. This is due to the latencies involved with operations performed by the processor. For example, for an HPCx Power5 processor, it takes around

- 3 cycles to retrieve data from L1 cache
- 15 cycles to retrieve data from L2 cache
- 80 cycles to retrieve data from L3 cache
- 350 cycles to retrieve data from main memory

Other sources of latencies include arithmetic operations, branch misprediction and TLB misses.

The compiler tries to schedule instructions in such a way as to minimise the number of idle cycles. However, the effectiveness of this is limited due to dependencies between instructions, and many idle cycles remain.

This situation of inefficiency has been magnified by the increased use of superscalar processors which have multiple functional units. The Power5 has 2 floating point units, and the theoretical peak performance can only be achieved if independent instructions can be issued to both of these units during each cycle. In practice, however, in addition to the idle processor cycles mentioned above (“vertical waste”), there exist cycles for which the processor is not idle but does not utilise all functional units (“horizontal waste”).

The concept of Multithreading (MT) aims to reduce such waste. The idea is that multiple threads (e.g. MPI tasks or OpenMP threads) can run on one physical processor. This provides a greater number of independent instructions which should therefore be able to more fully utilise the resources. The total execution time should be less than the time to run one thread multiplied by the number of threads. The main three types of multithreading are coarse-grain multithreading, fine-grain multithreading and simultaneous multithreading.

With coarse-grain multithreading, a single thread is executed until it encounters a long latency. Then execution is switched to a different thread until it encounters a long latency, and so on. Thus the number of latency induced idle cycles (vertical waste) is reduced. However, the switching between threads is not instantaneous but takes several cycles.

With fine grain multithreading, again a single thread issues instructions on each cycle, but thread switching can be done in a single cycle, further reducing the sources of vertical waste.

Simultaneous Multithreading (SMT) has the ability to reduce not just vertical but also horizontal waste [1]. With SMT, *multiple* threads can issue instructions to the multiple functional units on a single cycle. The Power5 processors in HPCx support SMT with 2 threads, which means that each thread is able to issue an instruction to one of the floating point units in each cycle. This aims to reduce the number of idle cycles as experienced by both floating point units, and hence improve performance of real applications.

## 4 Enabling SMT on HPCx

In order to use SMT on HPCx users must edit their Loadleveler scripts accordingly, i.e.:

- Include the requirements/Feature keyword
- Specify 32 tasks per node<sup>1</sup>.

Below is a loadleveler script that places a job in the 256\_1 processor queue. Note that the `cpus` keyword specifies the number of tasks to be run, i.e. one core with SMT running is equivalent to 2 tasks (and this is how the machine sees it), thus this setting enables 512 tasks to run on 256 processors in smt mode.

```
#@ shell = /bin/ksh
#
#@ job_type = parallel
#@ job_name = smt_test
```

---

<sup>1</sup>SMT will also be enabled for any number of tasks per node if the Feature keyword is set

```

#
#@ cpus = 512
#@ tasks_per_node = 32
#
#@ requirements = ( Feature == "SMT" )
#@ node_usage = not_shared
#
#@ wall_clock_limit = 00:20:00
#@ account_no = z001
#
#@ output = $(job_name).$(schedd_host).$(jobid).out
#@ error = $(job_name).$(schedd_host).$(jobid).err
#
#@ queue
#

export MP_EAGER_LIMIT=65536
export MP_SHARED_MEMORY=yes
export MEMORY_AFFINITY=MCM
export MP_TASK_AFFINITY=MCM
export MP_USE_BULK_XFER=yes

poe ./a.out

```

Use of SMT is restricted to non-shared nodes, therefore SMT-enabled jobs can only be run in the parallel batch queues. Evidently, the memory limit per task becomes half that of an equivalent non-SMT job.

## 5 Benchmark Results

Several codes were benchmarked both without and with the use of SMT. Without SMT, the number of MPI tasks is equal to the number of physical processors used. With SMT, 2 tasks (threads) are placed per processor. Plots will usually show the time multiplied by the number of physical processors (i.e. the cost of the job) against the number of physical processors used. This will give a indication of whether the use of SMT boosts the performance of the code for a given resource. In all the figures shown here, ideal scaling would be represented by a flat line plot. Investigated are the classical molecular dynamics codes NAMD, MDcask and DL\_POLY, plus the CASTEP and H2MOL applications, ScaLAPACK eigensolvers and STREAMS benchmarks.

### 5.1 Classical Molecular Dynamics

Classical molecular dynamics codes have been observed not to perform as well as expected on HPCx Phase 2a, likely due to sensitivity to an increased latency in some part of the memory subsystem [2]. As explained in section 3, SMT aims to hide such latencies. In

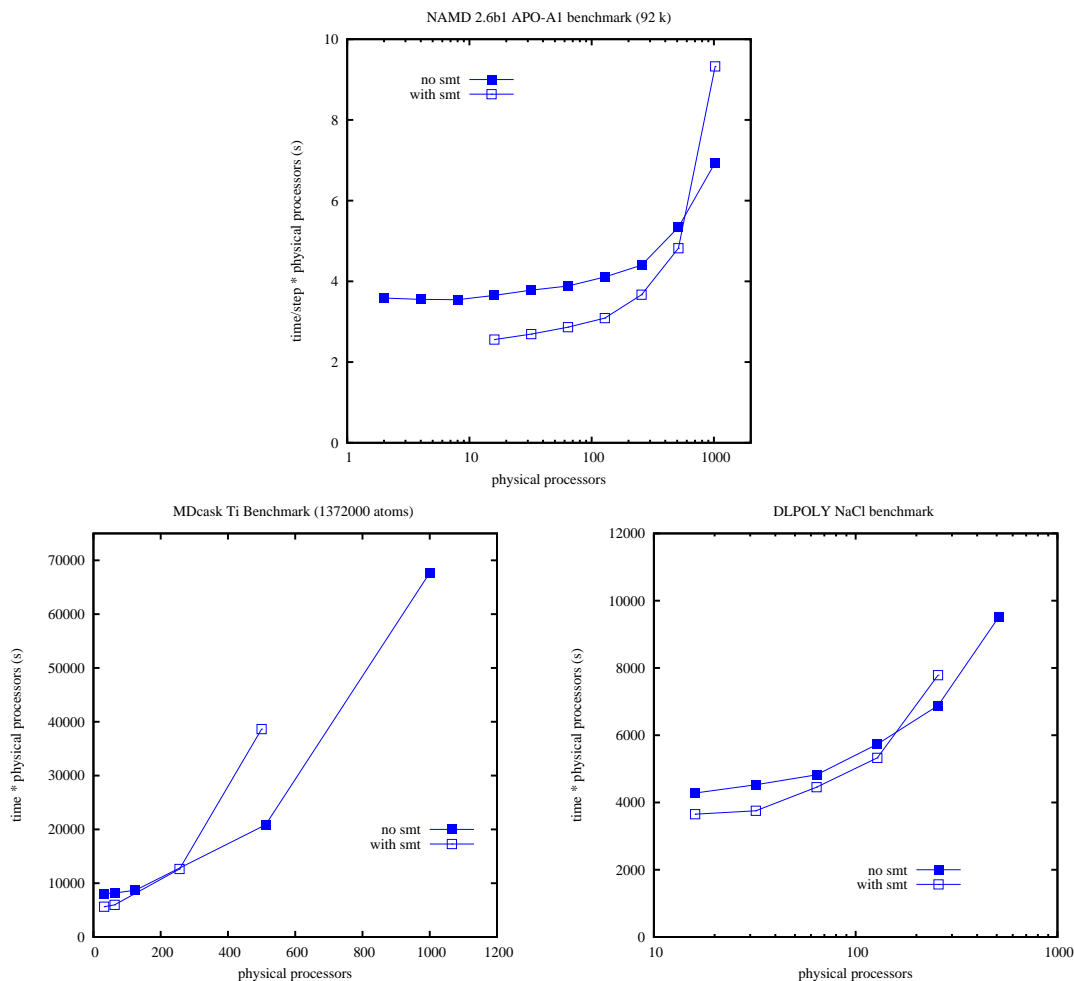


Figure 1: The dependence of total processor time on the number of processors for the NAMD ApoA1 (top), MDcask Ti (bottom left) and DLPOLY NaCl (bottom right) benchmarks. Closed and open shapes represent results without and with the use of SMT respectively.

this section, it is investigated whether the use of SMT can improve the performance of the NAMD, MDcask and DL\_POLY classical molecular dynamics codes, where

- The NAMD code is designed to simulate biomolecular systems such as proteins [3]. Results are given for the ApoA1 benchmark, which involves 92000 atoms.
- MDCASK was originally developed to study radiation damage in metals [4]. It operates by calculating the energies of and forces on, and determining the motions of, the atoms in the system which is characterised by specific interatomic potentials and boundary conditions. Here, results are given for the MDcask Ti Benchmark.
- DL\_POLY is a code which can be used to simulate systems with very large numbers of atoms [5]. Here, DL\_POLY3, which is parallelised by domain decomposition and is suitable for large numbers of processors (as opposed to DL\_POLY2 which uses a Replicated Data strategy and is suitable for of order 100 processors), was run for a system of 216000 NaCl ions.

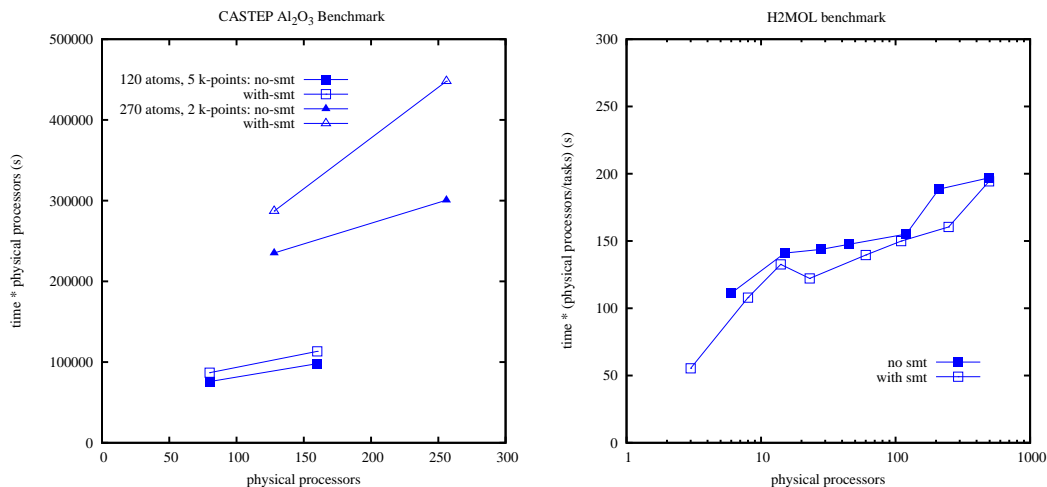


Figure 2: The dependence of total processor time on the number of processors for the CASTEP  $Al_2O_3$  benchmark (left) and the normalised time on the number of processors for the H2MOL benchmark (right). Closed and open shapes represent results without and with the use of SMT respectively.

It is seen from Figure 1 that for each of these codes the use of SMT significantly boosts performance at the low and moderate processor counts, by up to a factor of 1.4. These observed improvements indicate that the use of SMT is successfully hiding the memory latencies thought to hinder the performance of these applications.

As the number of processors increases the SMT advantage decreases to a crossover point, above which the code performs better without the use of SMT. This can likely be attributed to scaling effects. With the use of SMT, there are 2 tasks per processor so the total number of tasks is double that of the no-SMT results for a given resource. Therefore, with SMT, any communication bottlenecks are encountered at lower physical processor counts. This behavioural shift can be easily seen, e.g., in the MDcask high processor count results. The location of the crossover point depends on the code: it is around 512, 256 and 128 processors for NAMD, MDcask and DL-POLY respectively.

## 5.2 CASTEP

The materials science software package CASTEP can be used to perform molecular dynamics simulations and provide an atomic-level description (including information regarding energies, forces, and stresses, and calculations of optimum geometries, structures and spectra) of a wide range of materials and molecules [6]. CASTEP is known to perform well on HPCx Phase 2a [2]. Here results are given for two Aluminium Oxide ( $Al_2O_3$ ) systems: a 120 atom slab cell sampled with 5 k-points and a 270 atom cell sampled with 2 k-points.

From the left of Figure 2 it is seen that the use of SMT degrades performance in all cases. This degradation is not great for the 120 atom case but is more significant for the 270 atom case, especially at 256 processors.

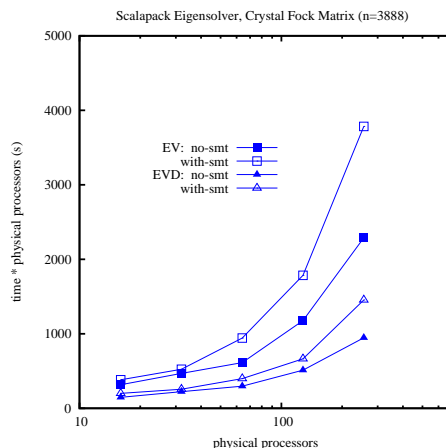


Figure 3: The dependence of total processor time on the number of processors for the processing of a  $3888^2$  Crystal Fock Matrix using the Scalapack EV and EVD eigensolvers. Closed and open shapes represent results without and with the use of SMT respectively.

### 5.3 H2MOL

H2MOL uses the Schrödinger equation to calculate the redistribution of energy between electrons and nuclei when hydrogen molecules are heated by short intense laser pulses [7]. Such calculations can be compared with experiment to help to develop the understanding of such molecular behaviour in general. A cylindrical computational grid is defined with Phi, Rho and Z co-ordinates, with the Z domain distributed amongst an array of processors arranged logically in a triangular grid (to take advantage of symmetry). Here the grid size is set to Phi points = 11, Rho points = 30 with 15 Z points per task. With this code the work increases as the number of tasks increases, so plotted on the right of Figure 2, to allow comparison, is the time normalised by the number of physical processors divided by the number of tasks, against the number of physical processors. Note that this comparison is complicated by the fact that H2MOL can only operate at specific task counts, and is very sensitive to under-filled nodes. However, the use of SMT is seen to have a significant advantage for all processor counts up to 496, at which point there is no significant difference from the non SMT result.

### 5.4 ScaLAPACK Eigensolver

Many codes utilise the ScaLAPACK [8] EV and EVD routines to compute eigenvalues and/or eigenvectors of large matrices, where the EV routine uses the QR method and the EVD routine the divide and conquer algorithm [9].

Figure 3 shows the results of benchmarking these routines for a  $3888^2$  Crystal Fock Matrix [10]. It is seen that the use of SMT is never seen to help: at low processor counts the use of SMT reduces performance very slightly, and this reduction in performance grows as the number of processors increases. This behaviour is explained by the ScaLAPACK routines' usage of highly optimised BLAS routines, thus ensuring little 'horizontal' or 'vertical waste' and a relatively high percentage of peak performance per processor. Consequently, SMT runs here yield little benefit at low processor counts and encounter communication bottlenecks at lower processor counts than non-SMT runs.

## 5.5 STREAMS

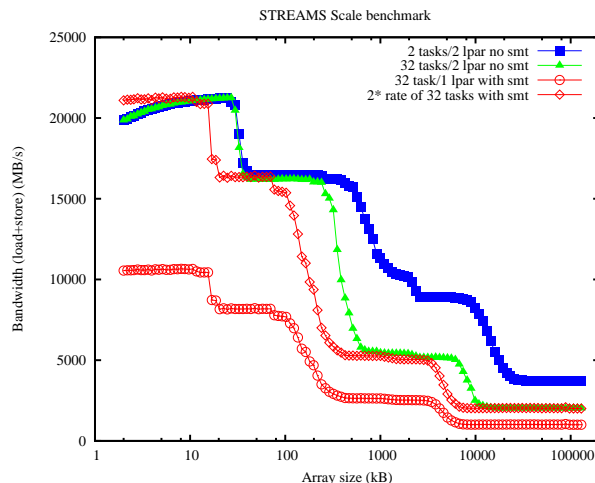


Figure 4: The dependence of the bandwidth on the array size for the STREAMS scale benchmark. The closed symbols represent results without the use of SMT: squares represent a single task per node (measured when using 2 nodes), and triangles 16 tasks per node asking for data simultaneously. The open symbols represent results with the use of SMT: circles represent 32 tasks on a 16-way node with SMT engaged and diamonds represent the circle denoted results multiplied by a factor of 2 to allow comparison.

The STREAMS benchmark suite enables measurements of memory bandwidth [11]. Shown in Figure 4 are resulting bandwidths, as a function of array size, for the “scale” benchmark (where the array is multiplied by a constant factor).

For results without the use of SMT, shown are bandwidths for both a single task per node and for fully saturated nodes (1 task per processor). Clearly seen from the steep transitions are the cache levels. The 32kB L1 cache is private to the processor and hence is seen at this array size in both the single and multi task cases. The 1.9M L2 cache is shared between 2 processors, so is seen at this array size for the single task case, and half this array size for the multi task case. Similarly, the 36MB L3 cache is shared between 2 processors. The main memory bandwidth is seen to be down, for a fully saturated node, by a factor of 1.8 from the single task case.

Results from runs using SMT are given for a fully saturated node with 2 tasks per processor. For comparison with the no-SMT results, also shown are these bandwidths multiplied by a factor of two. The use of SMT has no effect on the total bandwidth obtained from each memory level (and, as expected, the cache size is seen to half with SMT). This reinforces the belief that any observed increase in application performance with SMT is related to memory latency rather than bandwidth.

## 6 Conclusions

The use of SMT, which aims to improve the performance of applications by hiding latencies, was investigated on HPCx and was found to significantly boost performance in certain situations. The classical molecular dynamics codes, known to perform more poorly than expected on Phase 2a likely due to sensitivity to latency, were seen to benefit

from the use of SMT. The H2MOL and CASTEP codes already perform well on Phase 2a, and H2MOL was seen to benefit from SMT while CASTEP was seen not to.

The performance improvements with SMT were seen to decline with increasing processor counts to points where the effect on performance was insignificant or negative. This behaviour is likely due to scaling effects and is code specific.

No performance improvements were seen with SMT for the ScaLAPACK eigensolver EV and EVD routines, or for the STREAMS memory bandwidth scale benchmark.

It is recommended that the user benchmarks their own application to determine whether it benefits from the use of SMT.

## References

- [1] Dean M. Tullsen, Susan J. Eggers and Henry M. Levy, Simultaneous Multithreading: Maximising on Chip Parallelism, Proceedings of 22nd Annual Symposium on Computer Architecture, 1995.
- [2] Alan Gray *et al.*, A Performance Comparison of HPCx Phase 2a to Phase 2, *HPCx Technical Report 0602*,  
[http://www.hpcx.ac.uk/research/hpc/technical\\_reports/HPCxTR0602.pdf](http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0602.pdf)
- [3] 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/>
- [4] MDCASK README  
<http://www.llnl.gov/asci/platforms/purple/rfp/benchmarks/limited/mdcask/mdcask.readme.html>
- [5] Smith, W. & Forester, T. 1996, DL\_POLY: A General Purpose Parallel Molecular Dynamics Simulation Package. *J. Molec. Graphics* **14**, 136; Smith, W., Yong, C. & Rodger, M. 2002, DL\_POLY: Applications to Molecular Simulation. *Molecular Simulation* **28**, 385; Todorov, I. & Smith, W. 2004, DL\_POLY\_3: The CCP5 National UK Code for Molecular Dynamics Simulations. *Phil. Trans. R. Soc. Lond. A* **362**, 1835;  
[http://www.cse.clrc.ac.uk/msi/software/DL\\_POLY/](http://www.cse.clrc.ac.uk/msi/software/DL_POLY/)
- [6] M D Segall *et al* 2002, First-principles simulation: ideas, illustrations and the CASTEP code. *J. Phys.: Condens. Matter* **14** 2717-2744;  
<http://www.tcm.phy.cam.ac.uk/castep/>
- [7] D. Dundas, K.J. Meharg, J.F. McCann and K.T. Taylor, 2003, Dissociative ionization of molecules in intense laser fields *Eur. Phys. J. D* **26**, 51-57;  
<http://www.hpcx.ac.uk/research/atomic/h2mol.html>
- [8] ScaLAPACK documentation and downloads available at  
[http://www.netlib.org/scalapack/scalapack\\_home.html](http://www.netlib.org/scalapack/scalapack_home.html)
- [9] Ian Bush, Andrew Sunderland and Gavin Pringle, Scalable Eigensolvers on HPCx: Case Studies, *HPCx Technical Report 0510*,  
[http://www.hpcx.ac.uk/research/hpc/technical\\_reports/HPCxTR0510.pdf](http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0510.pdf)

- [10] CRYSTAL Website  
<http://www.hpcx.ac.uk/research/materials/crystal.html>
- [11] J.D. McCalpin, A Survey of Memory Bandwidth and Machine Balance in Current High Performance Computers, IEEE Technical Committee on Computer Architecture (TCCA) Newsletter, 1995.