Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Performance analysis of parallel applications on modern multithreaded processor architectures

Maciej Cytowski*[a], Maciej Filocha [a], Jakub Katarzyński [a], Maciej Szpindler [a]

[a]*Interdisciplinary Centre for Mathematical and Computational Modeling (ICM), University of Warsaw, Poland*

Abstract

In this whitepaper we describe the effort we have made to measure performance of applications and synthetic benchmarks with the use of different simultaneous multithreading (SMT) modes. This specific processor architecture feature is currently available in many petascale HPC systems worldwide. Both IBM Power7 processors available in Power775 (IH) and IBM Power A2 processors available in Blue Gene/Q are built upon 4-way simultaneous multithreaded cores. It should be also mentioned that multithreading is predicted to be one of the leading features of future exascale systems available by the end of next decade [1].

1. Introduction

Performance of today's general purpose processor architectures is driven by three main components: clock speed, number of computational cores and number of double precision operations per cycle. The combination of those three is widely used as a basic measure of processors performance known as FLOPs – number of floating point operations per cycle. Since further increasing clock speed and core count is technologically still very difficult, hardware vendors continue to develop different ways to increase single core application's performance i.e. vector processing units, support for fused multiply and add operations and hardware support for simultaneous processing of multiple threads, so-called multithreading. One of the most appropriate ways to measure real performance of a given processor architecture is to measure its efficiency when used for chosen classes of scientific algorithms and applications.

In this whitepaper, we describe the effort we have made to measure performance of applications and synthetic benchmarks with the use of different simultaneous multithreading (SMT) modes. It should be stated that SMT mechanism does not increase the maximum number of FLOPs, however it might influence the performance of chosen algorithms and applications. This specific processor architecture feature is currently available in many petascale HPC systems available worldwide. Both IBM Power7 processors available in Power775 (IH) and IBM Power A2 processors available in Blue Gene/Q are built upon 4-way simultaneous multithreaded cores. It should be also mentioned that multithreading is predicted to be one of the leading features of future exascale systems available by the end of next decade [1].

This work was motivated by results presented in [2] which show that the performance gain from SMT varies depending on the program execution and its execution model, the threading mode being used on the processor, and the resource utilization of the program. The gains from using SMT modes with chosen algorithms where measured with the use of few well known benchmarks: SPEC CFP2006, NAS Parallel Benchmark Class B (OpenMP) and NAS Parallel Benchmark Class C (MPI). One of the conclusions of the study presented in [2] was that throughput type workloads are best suited to see gains from using higher SMT modes. On the other hand high memory traffic codes will most likely not perform well when executed in SMT2 or SMT4 mode.

Through all of this paper we will extensively use the formulation that a specific application is using SMT2/SMT4 mode. By saying this we will refer to parallel codes which are executed with number of processes and/or threads that exceed the physical number of cores available in the system. This may be achieved by:

- executing an application with 2x or 4x more MPI processes,
- executing an OpenMP/Pthreads code with 2x or 4x more threads,
- mixing those two MPI and multithread execution modes (e.g. in the case of hybrid MPI/OpenMP codes).

Results presented in this whitepaper show that SMT mechanism available in modern processor chips can be efficiently use to

* Corresponding author. E-mail address: m.cytowski@icm.edu.pl

increase performance of chosen applications and algorithms. On the other hand, there is a class of algorithms and applications that does not benefit from mutithreading. In-depth investigation of the reasons of such divergence is described as future work.

In Section 2 we describe the effort we have made to measure performance of chosen scientific applications with different SMT modes. Our selection of applications was based on previous work of the PRACE project [3][4][5]. This benchmarks where executed on a Power750 system with two Power7 processors (each with 8 cores) operating at 3.5 GHz with total of 128 GB RAM. In Section 3 we present an in-depth analysis of one chosen application – GADGET2. We have measured performance of different algorithms used in GADGET2 code. In Section 4 we describe the work which was motivated by results of previous investigations. We have measured the performance of chosen scientific algorithms available in BOTS benchmark. BOTS benchmark was executed on computational nodes of PRACE Tier-1 Boreasz system available at ICM, University of Warsaw. Boreasz is a IBM Power775 (IH) supercomputer whose computational nodes (called octants) are composed of four Power7 processors (each with 8 cores) operating at 3.8 GHz with total of 128 GB RAM. This gave us additional information on the type of algorithms whose performance might benefit from SMT mechanism. At last we discuss our plans for future work.

2. Performance analysis of scientific applications

2.1 Computational cosmology

GADGET2 package was developed for large scale cosmological simulations on massively parallel computers with distributed memory [6]. The GADGET2 code employs a variety of computational algorithms: hierarchical tree and particle mesh algorithms for gravitational computations and Smoothed Particle Hydrodynamics (SPH) for modeling hydrodynamic of barionic content of the Universe. GADGET2 code uses few external libraries: Message Passing Interface (MPI) for parallelization, FFTW v.2.1.5 library for particle mesh computations and GNU Scientific Library for mathematical algorithms.

The test case used in this study consisted of almost 28 million particles from which the half were gas particles. The PMGRID parameter was set to the size of 1024. The following compiler options were used during the compilation phase on the POWER7 system: *-q64 -qarch=pwr7 -qtune=pwr7 -O3 -qhot -qaltivec -qsimd=auto*. Results of the measurements are shown in Figure 1.

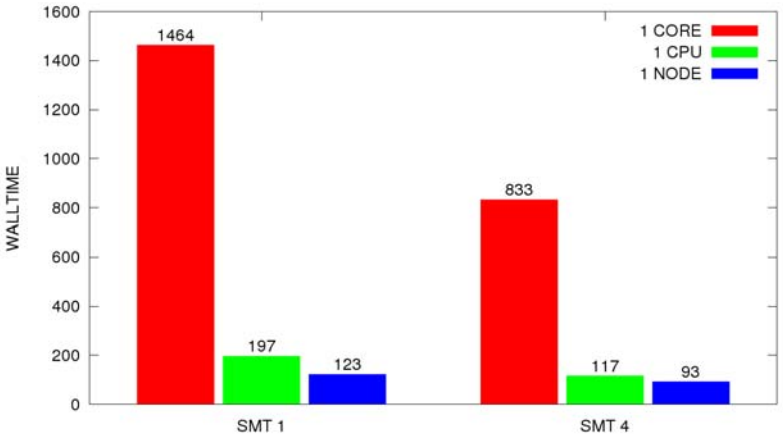

Figure 1 *Performance of GADGET2 application versus different SMT modes.*

During the preliminary testing phase we have noticed a very unusual low performance on POWER7 processor. The domain decomposition of GADGET2 code was more than 50x slower than on modern x86-64 architecture. Thanks to detailed analysis of the source code we have found very poor performing program fragment. Domain construction and decomposition algorithm implemented in GADGET2 code used *qsort* for sorting and for unspecified reason this program was responsible for generating such a poor time performance. We have corrected this issue by replacing the *qsort* call by *gsl_heapsort* procedure of GNU Scientific Library. The performance level achieved thanks to this minor change in the code was significant (around 46x faster when compared with initial version).

2.2 Climate modeling

WRF [7] is a numerical weather prediction model used for both operational forecasting and atmospheric research in many leading meteorological institutes around the world. WRF code was prepared for execution on highly parallel computer architectures and is well known from its ability to scale using significant amount of processes and computing cores.

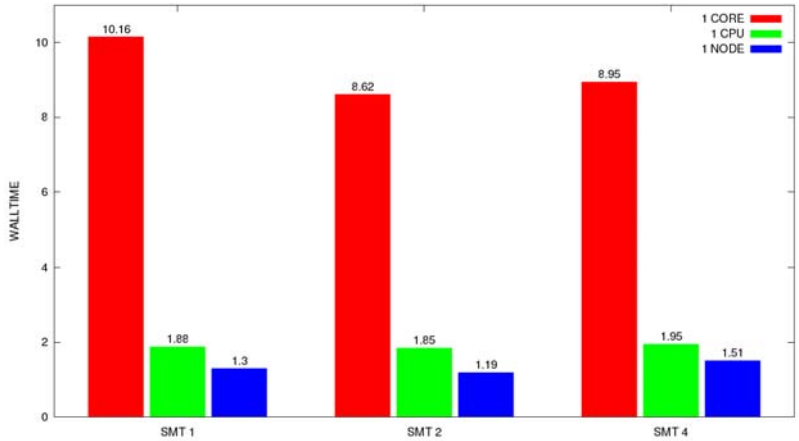

Figure 2 *Performance of WRF code measured against different SMT modes (mean elapsed time per time step)*

For the purpose of this analysis we have used WRF v.3.2 compiled in the 64-bit mode and MPI-only version (DMPAR). We have used *January 2000* test case of WRF as a benchmark. The following compiler options were used during the compilation phase on the POWER7 system: *-q64 –O3 –qarch=pwr7 –qtune=pwr7 =qaltivec –lmass –lmassv*. Results of the measurements are shown in Figure 2.

2.3 Molecular dynamics

We have selected two popular molecular dynamics codes for our benchmark: GROMACS [8] and CPMD [9].

GROMACS is a package for performing molecular dynamics simulations with hundreds to millions of particles. It is implemented in C and Fortran and uses MPPI library for parallelization. The test case used in this work was a vesicles in water system consisting of more than 1 million atoms. GROMACS v.4.0.7 was compiled and optimized on Power7 system with the use of following compiler options: *-q64 –qarch=pwr7 –qtune=pwr7 –O3 –qhot –qalitvec –qsimd=auto*. Results of the measurements are shown in Figure 3.

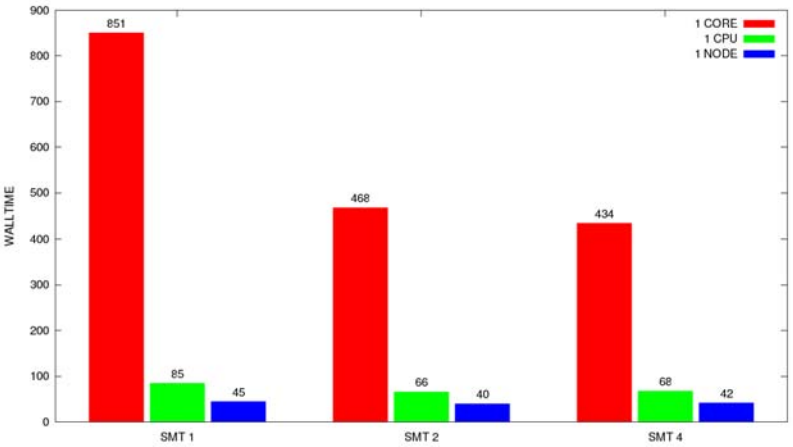

Figure 3 *Performance of GROMACS code measured against different SMT modes*

The CPMD code is an implementation of Density Functional Theory (DFT), particularly designed for ab-initio molecular dynamics. It runs on many different computer architectures including parallel systems. CPMD is parallelized with MPI and OpenMP. End users can choose between different versions of parallelization (distributed memory, shared memory and mixed modes) during the compilation phase. The test case used for performance measurements was a water system with 32 oxygen atoms and the mesh size of 128x128x128. CPMD v.3.13.2 was compiled and optimized on Power7 system in the MPI-only version with the use of following compiler options: *-q64 –qarch=pwr7 –qtune=pwr7 –O3 –qhot –qaltivec –qsimd=auto*. Results of the measurements are shown in Figure 4.

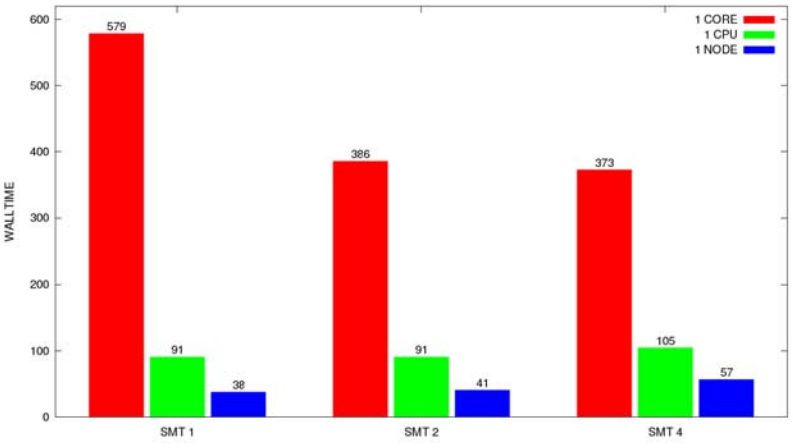

Figure 4 *Performance of CPMD code measured against different SMT modes.*

2.4 Materials sciences

We have executed SMT performance tests with GPAW simulation package [10]. GPAW is a Density-Functional Theory (DFT) code based on the projector-augmented wave method. It was written in Python and C and requires NuMPy and Atomic Simulation Environment (ASE) packages. The parallel version of the code was prepared with the use of MPI library. For the purpose of performance analysis on Power7 system we have run few iterations of a ground state calculations for 256 water molecules. GPAW v.0.7.6383 was installed on the Power7 system together with its all dependencies. The following options were used for C language parts of the package: *-q64 –O3 –qhot –qaltivec –qarch=pwr7 –qtune=pwr7*. Results of the measurements are shown in Figure 5.

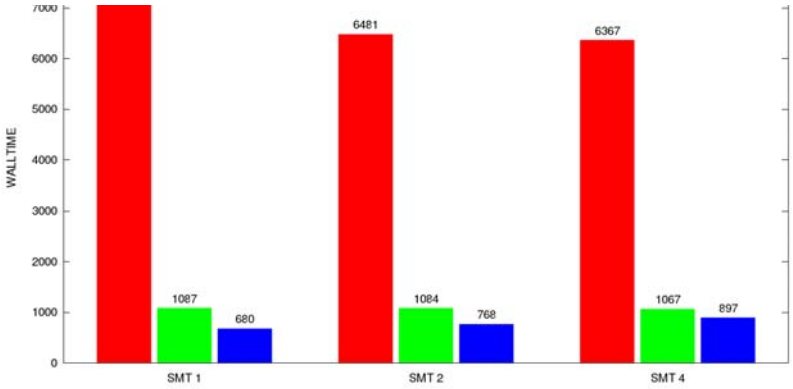

Figure 5 *Performance of GPAW code measured against different SMT modes.*

2.5 Conclusions

As it was expected the performance gain from SMT varies when measured with different applications. GADGET2 is an example of application which benefits from using SMT mechanisms. The best walltime results are always achieved for SMT4 mode regardless of number of cores in use. On the other hand in the case of WRF and GPAW codes the expected performance gain from using SMT is rather small (10% in the case of WRF). The usage of SMT mechanism in the case of WRF seems to be limited to SMT2. For GPAW the situation is even worst since using higher SMT modes decrease the overall performance. Performance results obtained for GROMACS and CPMD show that those codes achieve rather good results from using SMT2 mode, but not from SMT4.

All test described above are also very important recommendations for users of Boreasz Tier-1 system since they can check which type of SMT mode could be beneficial for their computations. As an example one of the latest large scale cosmological simulations of VIRGO consortium (so called CoCo simulation) computed on Boreasz system has been extensively using SMT2 mode. The simulation was executed on 68 nodes of the system and took around 1.5 months of computing. Performance gain was on the level of approximately 22%, which allowed us to save around 10 days of the machine time.

3 GADGET2 performance – in-depth analysis

In this section we take a closer look at the performance of GADGET2 code. This work was motivated by following observations:

- GADGET2 benefits from using higher SMT modes,
- GADGET2 implements and uses algorithms of different computational nature: tree code, FFT, particle computations.

We have decided to have a closer look at the performance of different GADGET2 algorithms measured in various SMT modes. Results are shown in Table 1.

	Single cpu			**Single node (2 cpus)**		
Algorithm	**SMT 1**	**SMT 2**	**SMT 4**	**SMT 1**	**SMT 2**	**SMT 4**
Tree walk	51,64s	33,57s	25,23s	26,73s	18,49s	13,07s
SPH	81,28s	56,26s	42,79s	42,2s	31,4s	22,42s
Particle-Mesh	40,39s	34,15s	33,53s	25,94s	31,84s	26,66s

Table 1 *Performance of GADGET2 algorithms measured against different SMT modes.*

Measurements obtained for different algorithms used in GADGET2 code demonstrate that SMT mechanism is not always the key for better performance. Especially the Particle-Mesh algorithm which extensively uses FFT computations does not benefit from SMT 2 and SMT 4 at all. However, both Tree walk and Smoothed Particle Hydrodynamics steps present very good performance when executed in higher SMT modes. These results motivated us for further work described in Section 5, where we investigate the performance of synthetic benchmarks. Our final goal is to understand why only chosen algorithms can benefit from multithreading. We believe that precise explanation can be only given through deep analysis of chosen performance metrics and performance counters.

4 Performance analysis of synthetic benchmarks

As it was described in Section 2, performance analysis of applications based only on walltime of the simulation run is sometimes not sufficient. To understand and identify performance bottlenecks an in-depth analysis of basic components of the applications (i.e. algorithms in use) is required. Therefore we have decided to analyze the performance of a representative set of algorithms which prove their usefulness in many scientific disciplines. Since the SMT mechanism is especially interesting in the case of multithreaded applications based on OpenMP/Pthreads model, we have decided to perform tests with BOTS benchmark [11]. BOTS was developed as a basic set of applications that allows researchers and vendors to evaluate OpenMP implementations, and that can be easily ported to other programming models. An additional goal was for the OpenMP community to have a set of examples using the tasking model available recently in OpenMP programming model.

Our testing procedure consisted of following steps:

1. Compile and execute BOTS benchmark on the Power775 system (Boreasz).

2. Identify the best scalable versions (tied/untied,for/single) through preliminary scalability testing of BOTS applications on the single node of Power775 system.

3. Test the performance of selected versions of BOTS applications against different SMT modes.

BOTS benchmark was compiled on Boreasz system with the use of following compiler options: *-q64 –qalloca –qsmp=omp –O3 –qarch=pwr7 –qtune=pwr7 –qaltivec -qthreaded.*

The following versions of chosen applications have been identified:

- Strassen (tied tasks, manual - 3) – computes a matrix multiply with Strassen's method, dense linear algebra algorithm
- N Queens (tied tasks, manual -3) – finds solutions of the N Queens problem, search algorithm
- SparseLU (tied tasks) – computes the LU factorization of a sparse matrix, sparse linear algebra algorithm
- Health (tied tasks, manual - 2) – simulates a country health system, simulation algorithm
- Floorplan (tied tasks, manual - 5) – computes the optimal placement of cells in a floorplan, optimization algorithm

Each of the application was executed with the largest possible input data provided within BOTS with the use of 32, 64 and 128 threads on the single node of Power775 system (Boreasz).

Application	**Single thread**	**Single node (4 cpus)**		
		SMT 1	**SMT 2**	**SMT 4**
Strassen	101,05s	5,11s	4,43s	5,88s
N Queens	27,8s	1,16s	0,83s	0,79s
SparseLU	129,19s	4,79s	4,18s	5,03s
Health	96,54s	3,97s	3,7s	4,8s
Floorplan	12,22s	0,55s	0,58s	0,79s

Table 2 *Performance of BOTS benchmark codes measured against different SMT modes.*

As can be seen in the above Table 2 performance of Strassen, SparseLU and Health algorithms can benefit only from SMT 2 mode. N Queens search algorithm improve its performance also in the SMT 4 mode. Only for the Floorplan algorithm we did not see any increase of performance when using higher SMT modes. As it was pointed out before, we believe that precise explanation of these results can be only given through analysis of chosen performance metrics and performance counters. This is one of the tasks that we plan to do in our future work.

5 Future work

Results described in Sections 2,3 and 4 motivated us to plan future work related to performance study of algorithms and applications on multithreaded and multicore architectures.

Firstly, we are very keen to know why performance of chosen algorithms can benefit from using higher SMT modes while others do not. We believe that this problem might be addressed by detailed analysis of hardware performance counters. Currently we are analyzing hardware performance counters for the chosen applications from the BOTS benchmark. We are looking for a correlation between the ability to efficiently use the SMT mechanism and the performance profile of the given application. Such

a result would lead us to better understanding of computational nature of different algorithms, but it could also be used to propose an automatic heuristic algorithm (e.g. based on decision trees) to decide which multithreaded code fragment should be using SMT2 or SMT4 mode.

Secondly, many modern HPC applications use both MPI and thread parallel model (e.g. mixed MPI + OpenMP). Parallel processes executed on different computational nodes include many thread parallel regions which are executed on available computational cores. Very often the number of threads in thread based parallelization is controlled by a single switch (e.g. the OMP_NUM_THREADS environment variable). Since different algorithms and code fragments may present different scalability on given HPC platform, it would be appropriate to choose number of threads for execution to each parallel region individually. Moreover such a decision could be made automatically only with a minor information gathered from code user (mainly the information about the preferred execution model of the code).

The tool that we are currently developing within PRACE-2IP project will address both above mentioned challenges.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. FP7-283493. The work is achieved using the PRACE Research Infrastructure Tier-1 resource Boreasz at ICM, University of Warsaw in Poland. We would like to thank Maciej Marchwiany (ICM, University of Warsaw), Mateusz Wolszczak, Piotr Iwaniuk and Piotr Wojciechowski (MIM, University of Warsaw) for helping us to obtain sufficiently good performance with chosen benchmarks.

References

[1] J.Dongarra et al., “The international exascale software project roadmap”, International Journal of High Performance Computing Applications, 2011, 25(I), 3-60
[2] J.Abeles et al., “Performance Guide for HPC Applications on IBM Power 775 Systems”, Release 1.0, April 15, 2012, IBM Systems and Technology Group
[3] A.D.Simpson, M.Bull, J.Hill, “Identification and Categorisation of Applications and Initial Benchmarks Suite”, PRACE-PP Public Deliverables, http://www.prace-ri.eu/Public-Deliverables
[4] M.Bull, S.Janetzko, J.C.Sancho, J.Engelberts, “Benchmarking and Performance Modelling on Tier-0 systems”, PRACE-PP Public Deliverables, http://www.prace-ri.eu/Public-Deliverables
[5] P.Michielse, L.Arnold, O.-P.Lehto, W.Lioen, “Final Benchmark Suite”, PRACE-PP Public Deliverables, http://www.prace-ri.eu/Public-Deliverables
[6] V. Springel "The cosmological simulation code GADGET-2", 2005, Monthly Notices of the Royal Astronomical Society
[7] J. Michalakes et al., “Development of a Next Generation Regional Weather Research and Forecasting Model”, Development in Teracomputing: Proceedings of the 9th ECMWF Workshop on the Use of High Performance Computing in Meteorology, 20011 Eds. Walter Zwieflhofer and Norbert Kreitz, World Scientific, Singapore, pp. 269-276.
[8] Berendsen et al. (1995) Comp. Phys. Comm. 91:43-56.
[9] D.Marx, J.Hutter, “Ab-initio Molecular Dynamics: Theory and Implementation”, Modern Methods and Algorithms in Quantum Chemistry, Forschungzentrum Juelich, NIC Series, vol. 1, (2000)
[10] J.J.Mortensen, L.B.Hansen, K.W.Jacobsen, “Real-space grid implementation of the projected augmented wave method”, Physical Review B, Vol.71, 035109 (2005)
[11] A.Duran, X.Teruel, R.Ferrer, X.Martorell, E.Ayguade, “Barcelona OpenMP Tasks Suite: A Set of Benchmarks Targeting the Exploitation of Task Parallelism in OpenMP”, Proceeding ICPP '09 Proceedings of the 2009 International Conference on Parallel Processing, p.124-131, IEEE Computer Society Washington, DC, USA