



CONTRACT NUMBER RI-222919

DEISA 2
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
SUPERCOMPUTING APPLICATIONS**

European Community Seventh Framework Programme
RESEARCH INFRASTRUCTURES
Integrated Infrastructure Initiative

Progress on Enhancing Scalability of Applications
in the First Year

Deliverable ID: DEISA2-D9.2
Due date: April 30th, 2009
Author: Mariano Vazquez, BSC

Project start date: May 1st, 2008
Duration: 3 years

| Project co-funded by the European Commission within the Seventh Framework Programme (2007-2011) | | |
|--|---|----------|
| Dissemination Level | | |
| PU | Public | X |
| PP | Restricted to other programme participants (including the Commission Services) | |
| RE | Restricted to a group specified by the consortium (including the Commission Services) | |
| CO | Confidential, only for members of the consortium (including the Commission Services) | |

Table of Contents

| | |
|--|----|
| Table of Contents | 1 |
| List of Figures | 2 |
| List of Tables..... | 2 |
| 1 Introduction | 3 |
| 1.1 Executive Summary..... | 3 |
| 1.2 References and Applicable Documents..... | 4 |
| 1.3 Document Amendment Procedure..... | 4 |
| 1.4 List of Acronyms and Abbreviations..... | 4 |
| 2 Enhancing Scalability: EPCC..... | 5 |
| 2.1 Motivation | 5 |
| 2.2 Code selection..... | 5 |
| 2.3 Test platforms | 6 |
| 2.3.1 HPCx | 6 |
| 2.3.2 HECToR | 6 |
| 2.4 Results and discussion | 7 |
| 2.5 Conclusions | 12 |
| 2.6 Future work..... | 12 |
| 3 Enhancing Scalability: BSC | 13 |
| 3.1 Motivation | 13 |
| 3.2 Code enhancement..... | 13 |
| 3.3 Conclusions and Future work | 17 |
| 4 Enhancing Scalability: FZJ..... | 18 |
| 4.1 Introduction and motivation | 18 |
| 4.2 Aim of the enabling work | 18 |
| 4.3 Performance analysis of the code | 18 |
| 5 Work plan for the Next 12M period | 21 |

List of Figures

| | |
|---|----|
| Figure 1 – MPI performance of CPMD for HECToR and HPCx..... | 7 |
| Figure 2 – Performance of CPMD on 16 processors. | 8 |
| Figure 3 – Performance of CPMD on 32 processors. | 9 |
| Figure 4 – Performance of CPMD on 64 processors. | 9 |
| Figure 5 – Performance of CPMD on 128 processors. | 10 |
| Figure 6 – Performance of CPMD on 256 processors. | 10 |
| Figure 7 – Performance of CPMD on 512 processors. | 11 |
| Figure 8 – Performance of CPMD on 1024 processors. | 11 |
| Figure 9 – NAMD runs comparison for Marenstrum for different versions..... | 14 |
| Figure 10 – Application behaviour changing the PME implementation: with Slab PME. | 14 |
| Figure 11 – Application behaviour changing the PME implementation: with Pencil PME. ... | 15 |
| Figure 12 – Communication delay as showed by the Projections performance analysis tool. 15 | |
| Figure 13 – NAMD's time profile graph showing four algorithm iterations. | 16 |
| Figure 14 – NAMD vs Gromacs comparison for a large system (700K atoms)..... | 16 |
| Figure 15 – NAMD vs Gromacs comparison for a medium-size system (145K atoms). | 17 |
| Figure 16 – Analysis of a 4096 core (1 rack) run on Jugene using a grid of 2048x2048x1024 grid points..... | 19 |

List of Tables

| | |
|---|---|
| Table 1 – Summary of potential mixed mode codes from DECI proposals. | 5 |
|---|---|

1 Introduction

1.1 Executive Summary

The role of Work Package 9 in DEISA2 (Distributed European Infrastructure for Supercomputing Applications) is to enhance the scalability of a set of selected applications, which must be adapted to efficiently use this infrastructure. As written in the DoW [1], WP9's objective is "to support WP5 by enhancing the scalability of the scientific applications". While WP5 mainly adapts a group of scientific applications to run in supercomputers, WP9 provides additional effort to reach the highest standards to a carefully selected subset of applications.

As described in the DoW, the keywords of the process of enhancing scalability are analysis, design, scaling and verification. The present deliverable deals with all the points addressed in the DoW:

- Analysis of applications to enhance scalability in different computational fields;
- Design of the optimization and deployment of scalable applications, both in terms of execution speed and memory usage, in cooperation with the scientific partners in charge;
- Scaling of applications: code modification, optimization and I/O tuning, in cooperation with the scientific partners in charge;
- Analysis and verification of parallel code efficiency and scalability, addressing new approaches and methodologies.

In this document we present separately what has been done by three partners that take part in this WP: EPCC, BSC and FZJ, respectively in three sections. It is worth to remark that the WP9 work in Enhancement is directly mapped in the new public versions of the target codes, because these tasks are always being done in strong collaboration with the code developers. This document is the follow-up of the first deliverable of WP9, which describes the use of performance analysis tools and the metrics they obtain.

The three codes targeted in this deliverable are respectively CPMD, NAMD and CLOUD09. In the first case, EPCC worked out the hybridization of CPMD code, using OpenMP threads. CPMD is a parallelized plane wave/pseudopotential implementation of Density Functional Theory, particularly designed for ab-initio molecular dynamics. The hybridization strategy is appealing for reasons such as to decrease the network usage or to reduce the potentially large amount of I/O. To "hybridize" a code, by combining MPI and OpenMPI-based parallelization techniques, means to take profit of the real architectures of most of the supercomputing facilities, which are vast clusters of nodes with distributed memories. But on the other hand, the nodes integrate more than one CPU, which at node level, share memory. Therefore, it seems reasonable to use both parallelization paradigms at the same time. However, regarding the efficiency of hybrid codes, opposing conclusions can be found in the literature because this strategy is useful depending many factors. The present case study is a positive example.

NAMD is a widely used parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems, from the University of Illinois at Urbana-Champaign. After some successful DECI projects based in NAMD, BSC and the code developers focused in some aspects of the code worth to be improved. One of these points is

communication, because this kind of codes is characterized by a heavy communication use. Another improved issue was the fine-tuning of compilation flags.

CLOUD09 is the name of a project where cloud formation under moist convection is simulated. The code is based in a pseudo-spectral method, suitable for solving turbulence in domains with symmetric boundary conditions. Once the code is enabled, a performance analysis tool, SCALASCA, was used to identify the bottlenecks of the FFT that lies at the core of the program.

1.2 References and Applicable Documents

- [1] Description of Work (Annex I of the Grant Agreement)
- [2] DEISA web pages: <http://www.deisa.eu>
- [3] Deliverable DEISA2-D1.1: Initial Report on Management
- [4] Deliverable DEISA2-D9.1: Initial Report on Enhancing Scalability
- [5] HPCx WWW Site, <http://www.hpcx.ac.uk>
- [6] Hutter, J., and Curioni, A., 2005. Dual level parallelism for ab initio molecular dynamics: Reaching teraflop performance with the CPMD code. *Parallel Computing*, (31) 1-17.
- [7] HECToR WWW Site <http://www.hector.ac.uk>
- [8] Top 500 for November 2008, <http://www.top500.org/list/2008/11/100>
- [9] CPMD WWW site, <http://www.cpmc.org/>
- [10] NAMD2.6 users guide: <http://www.ks.uiuc.edu/Research/namd/2.6/ug>
- [11] S. Kumar, Chao Huang, G. Almasi and L.V. Kale, "Achieving strong scaling with NAMD on Blue Gene/L", *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006*. <http://ieeexplore.ieee.org/xpl/RecentCon.jsp?punumber=10917>
- [12] D. Pekurovsky, "P3DFFT – Highly scalable parallel 3D Fast Fourier Transforms library", San Diego Supercomputer Center/UC San Diego, 2008 <http://www.sdsc.edu/us/resources/p3dfft/index.php>
- [13] A. Chan, P. Balaji, W. Gropp, R. Thakur, "Communication Analysis of Parallel 3D FFT for Flat Cartesian Meshes on Large Blue Gene Systems", *High Performance Computing – HiPC 2008, Proceedings* **5374**, p. 350-365 (2008).

1.3 Document Amendment Procedure

This document is prepared according to the guidelines defined by the management of DEISA2. These rules can be found in section 2.7 of the deliverable DEISA2-D1.1 [3].

1.4 List of Acronyms and Abbreviations

| | |
|----------------|---|
| CPMD | Car-Parinello Molecular Dynamics application |
| DECI | DEISA Extreme Computing Initiative |
| DEISA | Distributed European Infrastructure for Supercomputing Applications |
| DoW | Description of Work (Annex I of the Grant Agreement) |
| HPC | High Performance Computing |
| MPI | Message Passing Interface |
| Open MP | An API for multi-platform shared-memory parallel programming in C/C++ and Fortran |

2 Enhancing Scalability: EPCC

This document gives a summary of the work carried out under DEISA2 WP9 at EPCC to investigate the performance of the mixed-mode (i.e. MPI and OpenMP) code, CPMD, on the HPCx and HECToR machines. We report on the motivation for this study, the methods used to select a suitable code, the test platforms and preliminary results. We also present our preliminary conclusions and outline the future direction that the study will take.

2.1 Motivation

Multi-core processors are becoming increasingly prevalent in high performance computing (HPC) systems nowadays. Systems containing dual, quad and octo-core nodes are becoming the rule rather than the exception. Eight out of ten of the current Top 500 HPC systems are quad-core systems [8]. Typically, this increase in the number of cores per chip does not come with a matching increase in the network or memory bandwidth available to each core. For example, a system may be upgraded from single to dual core, or dual to quad core with no appreciable change in the bandwidth available to the communications socket, i.e. the dual core now has half the bandwidth of the single core, quad core now has half the bandwidth of the dual core. As the number of cores continues to increase this problem is only going to get worse and it is likely that the performance of parallel codes will start to degrade as result.

Mixed mode or hybrid programming offers a potential improvement to this problem. Mixed mode programming involves using a combination of message passing between the nodes (typically via MPI) and shared-memory programming within the nodes (typically using OpenMP or POSIX threads). Using a mixed mode approach can potentially reduce the total number of messages going across the network. This is typically achieved by first aggregating the messages on the nodes (via OpenMP) before sending fewer, larger messages between the nodes (via MPI). It is also possible that keeping computations within a node may help to reduce the memory latency (data may be able to be kept on local caches) but this effect is likely to be small.

2.2 Code selection

| Name of code | Application area | Mixed mode [Y/N] | Freely available source code? | No. of projects involved |
|--------------|--|------------------|-------------------------------|--------------------------|
| CPMD | Ab-initio molecular dynamics using density functional theory (DFT) | Y | Y | 3 |
| CP2K | Ab-initio molecular dynamics using DFT | Y | Y | 5 |
| Molpro | Ab-initio code for electronic structure | Y | N – requires licence fee | |
| COSMOS | Earth system model | Y | ? | 1 |
| NANOPARS | Continuum/ quantum mechanics | Unknown | N | 1 |
| Siesta | Ab-initio molecular dynamics | N | Y subject to licence | 1 |

Table 1 – Summary of potential mixed mode codes from DECI proposals. The grey shading indicates codes which could potentially be used for this study – Molpro was later excluded due to licensing costs.

All current and future DECI proposals obtained up to January 2009 were examined and any proposals which claimed to utilise mixed mode codes were selected for further investigation. Each of the 10 proposals was examined to determine whether any of the codes intended to be used had mixed mode capabilities and if so whether they were suitable for our study. A summary of the codes considered is given in [5].

Of the proposals examined, three involved the CPMD code, with five involving CP2K. As CPMD and CP2K involve similar application areas we chose to use one of these codes. The CPMD code was chosen in preference to CP2K because it is already one of the DEISA benchmark suite codes. In addition, a detailed study of the mixed mode performance of the CPMD code was carried out in 2004/05 [6]. The Molpro code also claims to have mixed mode capabilities and appears to be a suitable candidate for further study. However, the licensing costs for this code make any further investigation impossible.

The COSMOS, NANOPARS and Siesta codes were rejected for the following reasons. The code used by COSMOS is no longer included in the DEISA benchmark suite and the proposal stated that installing/enabling this code would take longer than the timescale of this part of WP9. The NANOPARS source code is not available and thus cannot be used. The Siesta code is mentioned on the list of codes but not referred to in the actual proposal. However, assuming that it is the well-known ab-initio molecular dynamics code, then, Siesta does not have any mixed mode capability.

CPMD is an ab-initio molecular dynamics code which uses density functional theory. Further details on the code can be found at [9]. The code is written in Fortran 77/90 and is parallelised using MPI and OpenMP. Pure MPI, pure OpenMP and mixed mode versions can all be compiled. The code requires the BLAS and LAPACK libraries to be available. The code includes a Fast Fourier Transform (FFT) implementation but FFTW or other system-specific FFT implementations (e.g. IBM's ESSL library) can also be used. If the mixed-mode version is compiled then threaded versions of the BLAS, LAPACK and FFT libraries should ideally be used.

2.3 Test platforms

Two systems were selected for our initial investigations; HPCx and HECToR. These systems are part of the DEISA2 framework and have previously been tested with the CPMD code.

2.3.1 HPCx

HPCx is a cluster of 160 IBM POWER5 eServer nodes. Each node is a 16-way SMP containing 16 1.5 GHz IBM POWER5 processors, each of which have access to 2 GB of memory. This means the largest number of threads that a pure OpenMP job can have is 16. Further details on the system can be found at [5]. A previous study comparing the MPI, OpenMP and mixed mode performance of CPMD was carried out on one the earlier phases of HPCx [6].

2.3.2 HECToR

HECToR (Phase 1) is an integrated system comprising of a scalar and a vector component. This report uses only the scalar part which consists of a Cray XT4 system. The system comprises 5664 dual core 2.8 GHz AMD Opteron Processors each of which has access to 3 GB of memory. The largest pure OpenMP job that can be run is 2 threads. Further details on the system can be found at [7].

2.4 Results and discussion

Figure 1 shows the CPU time plotted against the number of physical processors for both HPCx and HECToR for problem sizes containing 1 to 256 molecules. Comparing the HPCx (solid symbols) and HECToR (open symbols) MPI results we see that overall the HECToR system is faster (in terms of CPU time) than HPCx but in addition it also appears to scale to larger processor counts. Comparing the HECToR MPI (open symbols, solid line) with the HECToR mixed-mode (open symbols, dashed line) we can see that the mixed-mode code scales better and in particular allows the code to continue scaling for processor counts at which the MPI scaling has broken down. Essentially, the mixed-mode version of the code allows the scaling envelope to be extended.

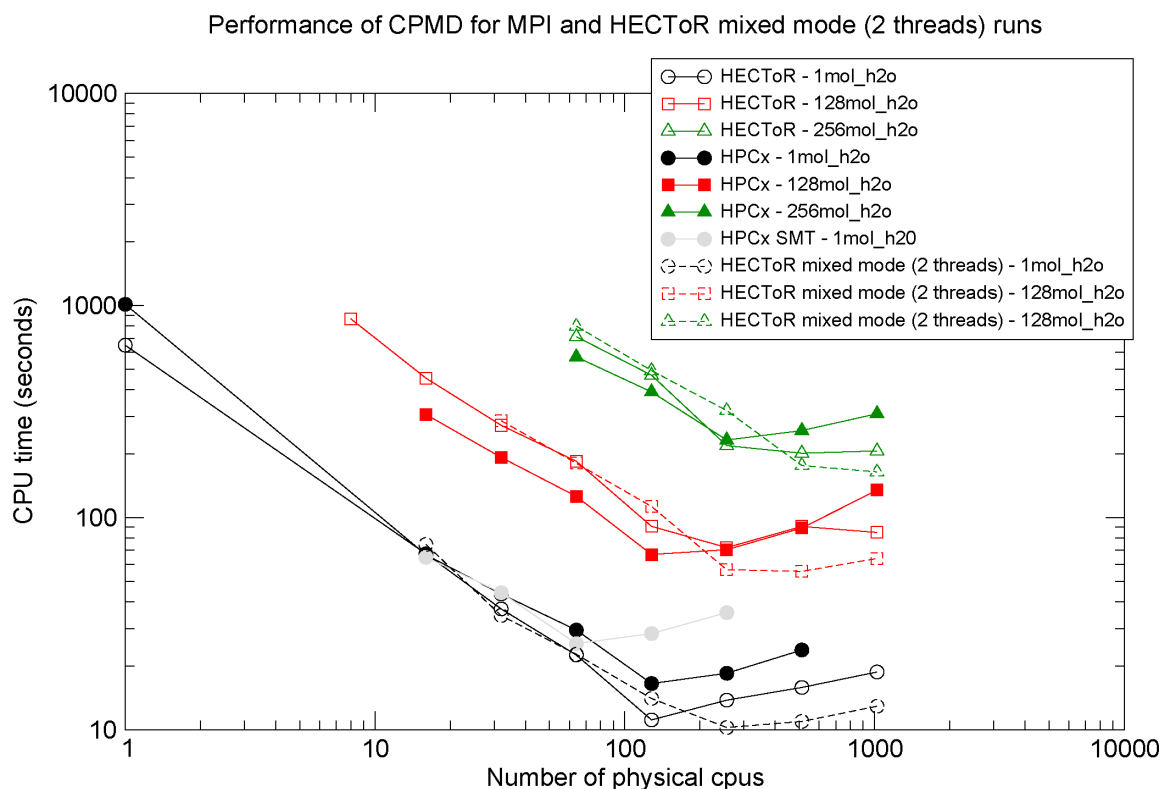


Figure 1 – MPI performance of CPMD for HECToR and HPCx. Mixed mode (2 thread) results also included (dashed lines) for HECToR

Figure 2 - Figure 8 show the performance plotted against number of threads for the physical processor count ranging from 16 to 1024. In each of these plots the performance is plotted using the MPI performance as a baseline with the dashed line indicating the MPI baseline. Points below this line indicate that the mixed mode version performed better than the pure MPI version and vice versa. For Figure 2 - Figure 5 the mixed mode version has little advantage over the pure MPI version (most data plots above the dashed line). However, for Figure 6 - Figure 8 the mixed mode version shows a clear advantage over the MPI version (most points plotting below the line).

On HPCx it is possible to examine how the performance relates to the number of OpenMP threads. The best performance is usually obtained for the 4 threads case (see Figure 4, Figure

7, Figure 8). For this case each node runs 4 MPI tasks each of which uses 4 OpenMP threads. For the 256 processor case (Figure 6) the 2 thread case gives the best performance. It is possible that the 4 thread case hits a sweet spot where the savings made by using OpenMP outweigh any overheads incurred by its use.

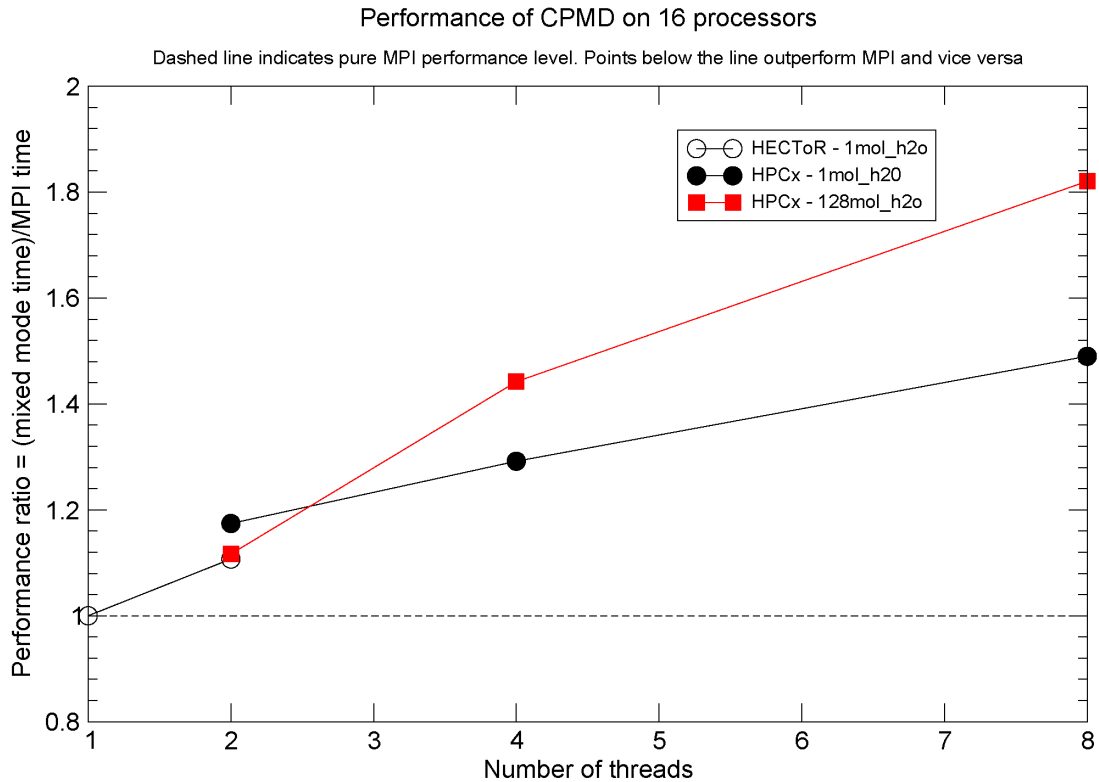


Figure 2 – Performance of CPMD on 16 processors.

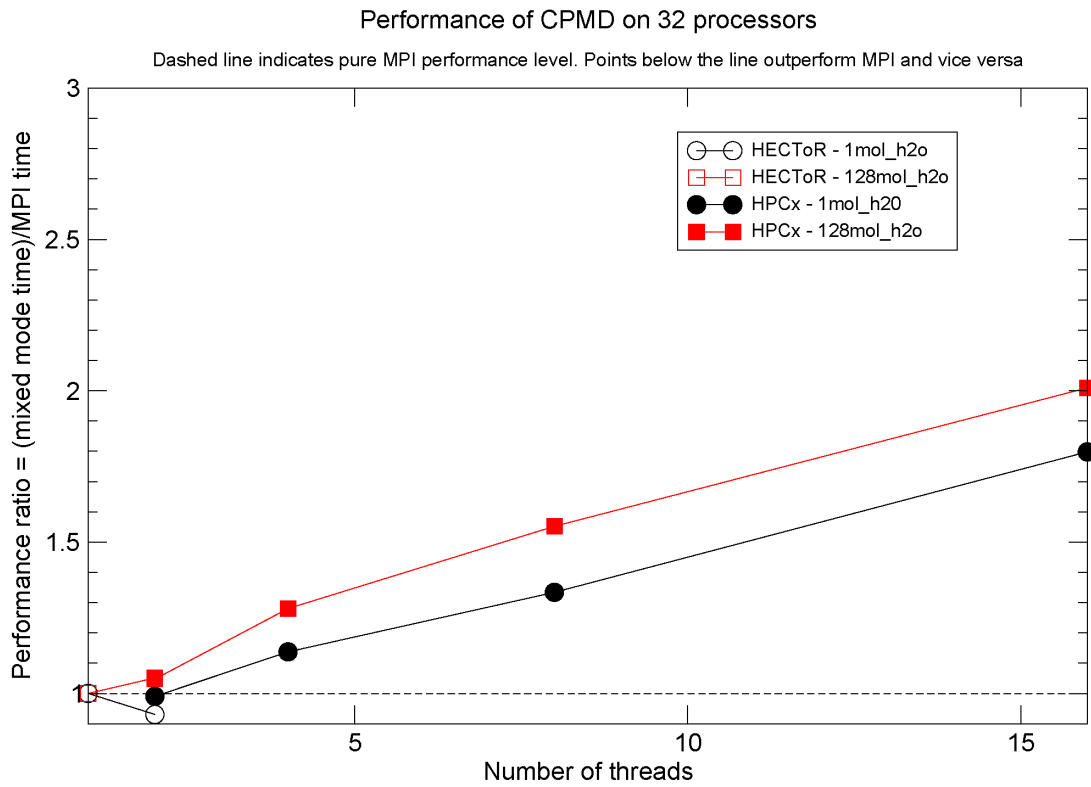


Figure 3 – Performance of CPMD on 32 processors.

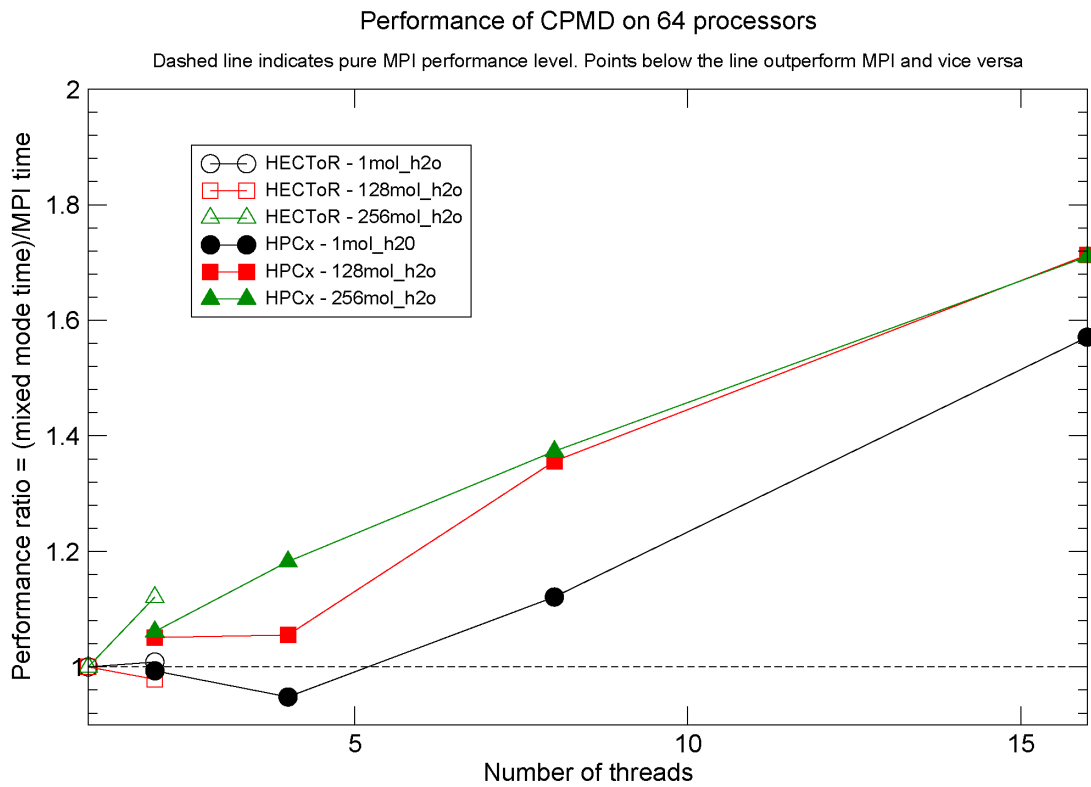


Figure 4 – Performance of CPMD on 64 processors.

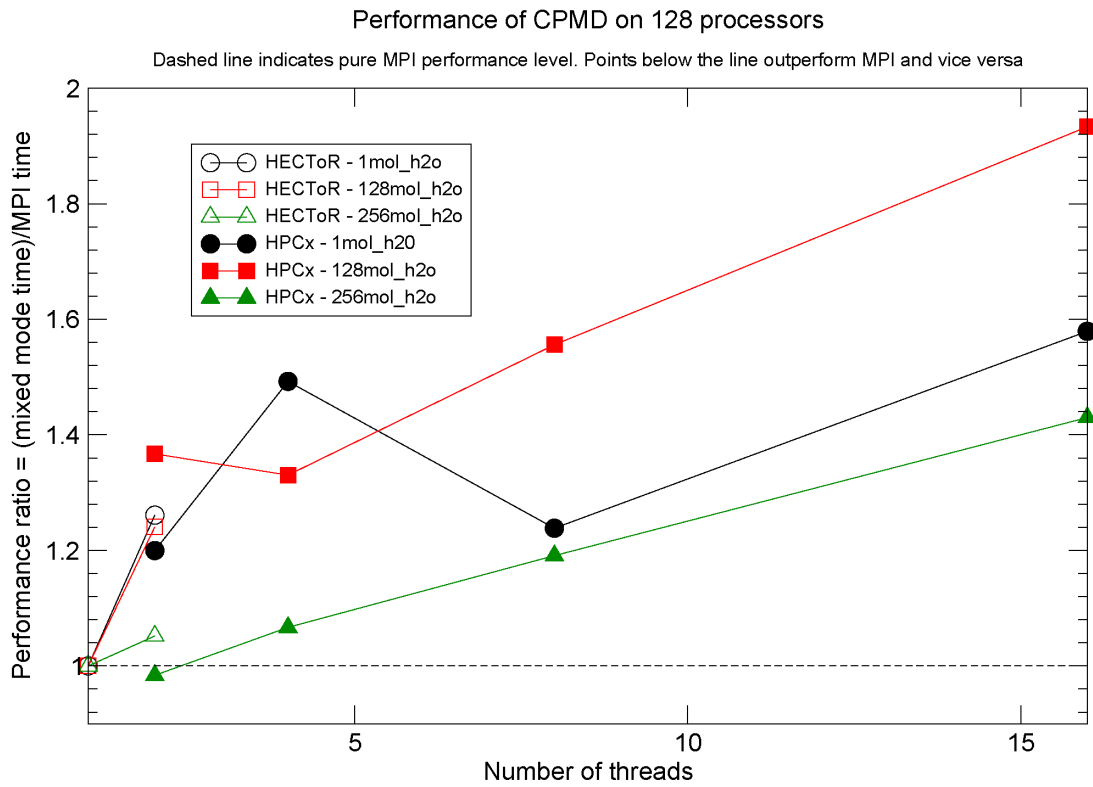


Figure 5 – Performance of CPMD on 128 processors.

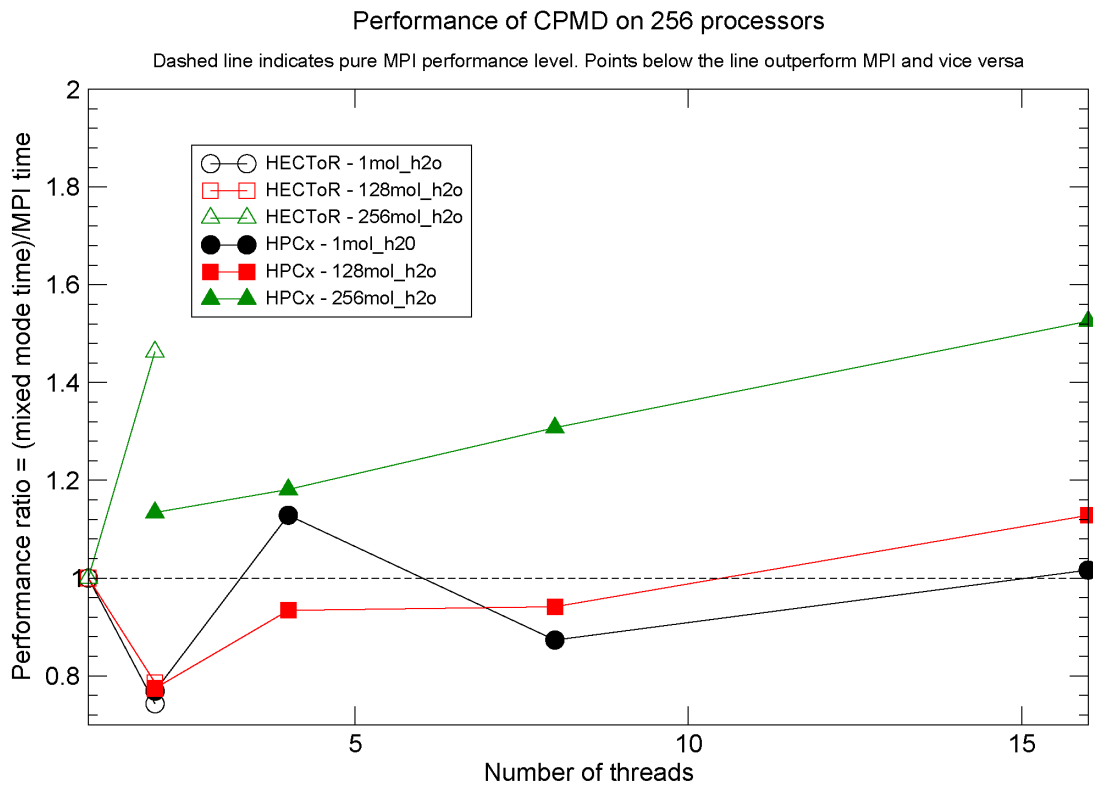


Figure 6 – Performance of CPMD on 256 processors.

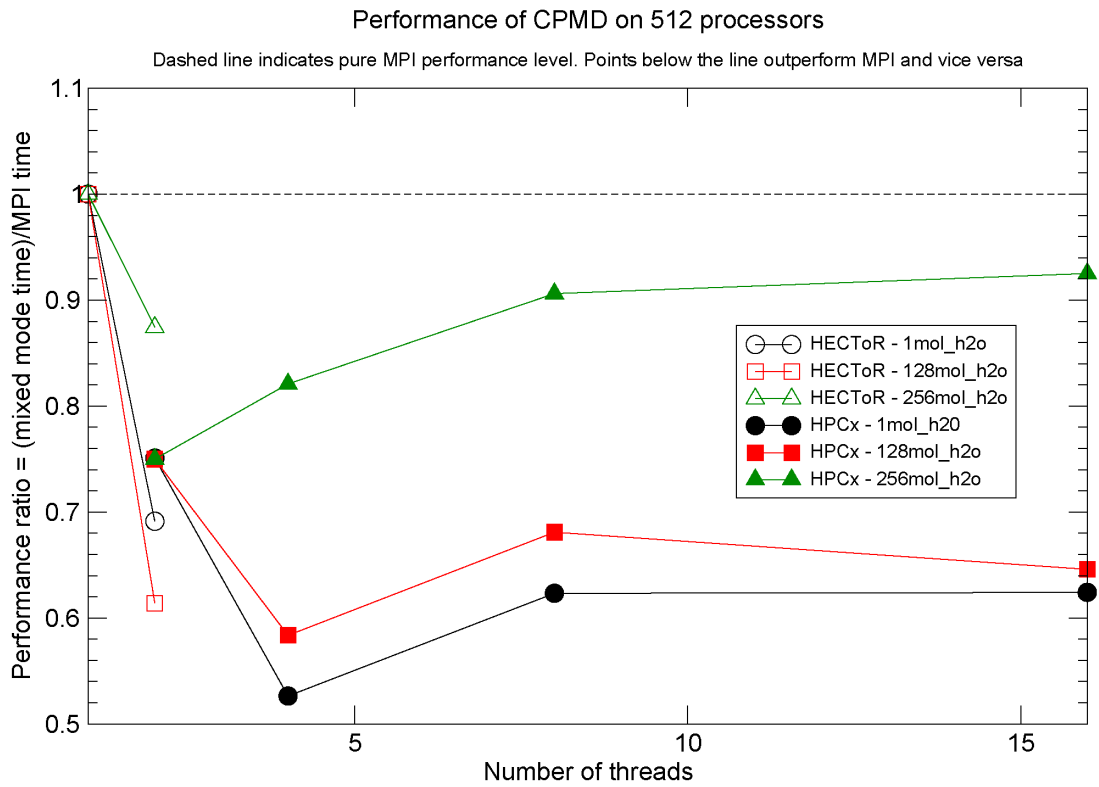


Figure 7 – Performance of CPMD on 512 processors.

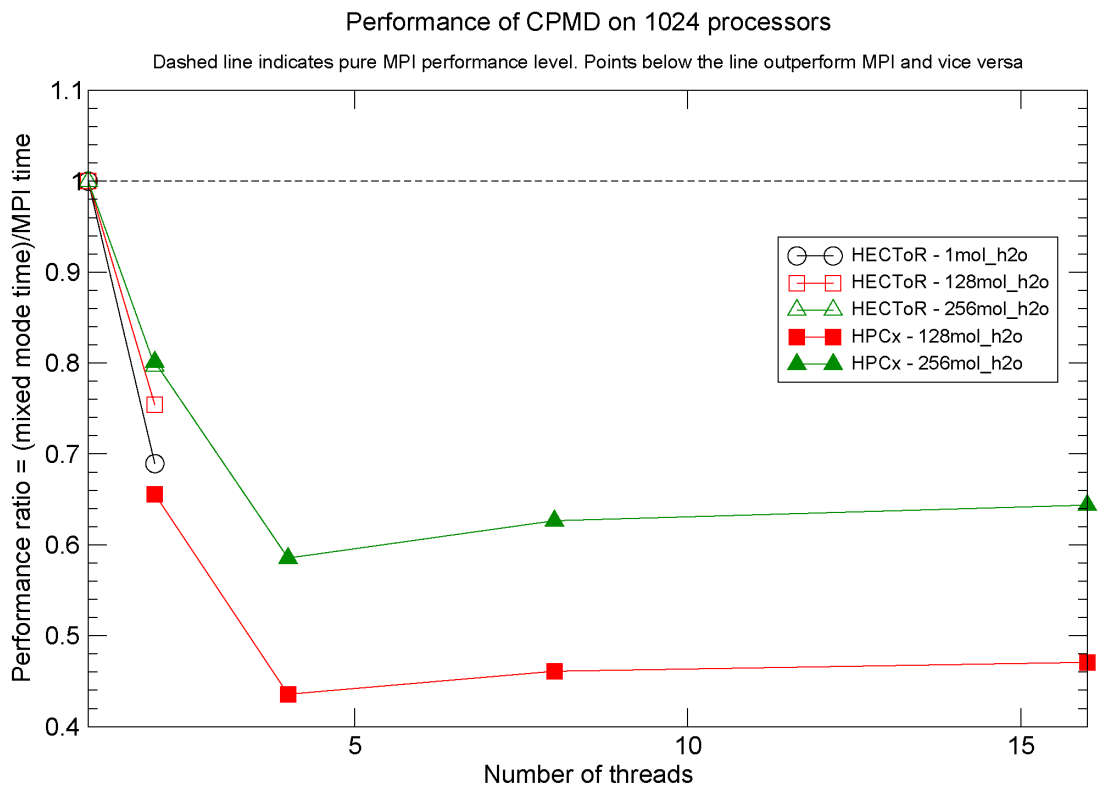


Figure 8 – Performance of CPMD on 1024 processors.

2.5 Conclusions

The results of this preliminary investigation show that for the CPMD code, the mixed mode version of the code allows the code to scale to a larger number of processors than is possible for the pure MPI version. However, the benefit is quite marginal overall as, at these numbers of processors, the code has ceased to scale particularly efficiently. In addition, for situations where the code is being run close to its scaling limit, the mixed-mode version has the potential to reduce the runtime for a given processor count.

2.6 Future work

The quad and octo core performance of the CSC Louhi system will be examined to allow comparison between dual, quad and octo core (2 quad cores) mixed mode performance. In addition and if time allows additional codes may also be investigated, e.g. CP2K. We will also compare the results obtained here to results obtained from the new mixed-mode micro-benchmark suite developed under the PRACE project. It will be interesting to discover how useful these low-level benchmarks are in predicting the mixed-mode performance of real parallel applications.

3 Enhancing Scalability: BSC

NAMD is a widely used parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems, developed and maintained by the Theoretical and Computational Bio-physics Group from the University of Illinois at Urbana-Champaign. BSC and the code developers focused in some aspects of the code worth to be improved, such as communication issues and compilation flags specific of MareNostrum.

3.1 Motivation

After a successful enabling and subsequent use in production, BSC support group and NAMD developers started a joint effort in improving the performance of the code. The main motivation of this effort is that the large users' community of NAMD continuously demands an improvement of the code scalability properties. As stated in its web page, "NAMD (NANOSCALE MOLECULAR DYNAMICS) can simulate the movement of proteins with millions of atoms, making it the world's fastest parallel molecular dynamics program". For this purpose one of the developers travelled to Barcelona to work side by side with the local support group.

3.2 Code enhancement

The first addressed point were some MareNostrum specific optimizations such as fixes done to the NAMD CVS version which were causing some crashes due to a bug of `xlc`, the IBM C/C++ compiler. Once the crash problem was fixed, the use of the Projections performance analysis tool provided by Charm++ allowed to identify possible bottlenecks affecting the scalability. Charm++ and AMPI are the core tools of parallelization in NAMD. Charm++ is a parallel C++ library. AMPI is an adaptive MPI implementation. They are supported by the code developers group and based on its research on intelligent runtime systems, including load balancing and communication optimizations. So in this case, the performance visualization and analysis tools were provided by the NAMD group itself.

In addition, some modifications to NAMD configuration parameters were done to allow high parallelization. Some of the modifications studied were:

- An increasing of the number of objects involved in the simulation:
 - `twoAway{X, Y, Z}` options
- Choice of PME (Particle Mesh Ewald) implementation:
 - Slab (1D) vs. Pencil (2D) decomposition
- Offloading processors doing PME
- Use of spanning tree for communication

The lack of scalability beyond 1024 CPUs was related to communications delays only explained by the performance of the Myrinet and alien to the NAMD code. Most of the communication problems were solved after the last MareNostrum maintenance, although some further analysis of the network performance need to be done.

The accompanying images are obtained from the projections performance analysis tool provided by Charm++. As shown below, the Overview (Figure 10 and Figure 11) gives the user a general picture of application behaviour in terms of utilization across processors and

over time. The difference between Figure 10 and Figure 11 is the choice of PME (Particle Mesh Ewald) implementation from Slab (1D) to Pencil (2D) decomposition.

The Timeline (Figure 12) gives the most detailed look into exactly what performance events occurred on each selected processor, allowing the examination of causal effects and other runtime information. The figure shows a communication delay (up to 1.2 ms for a point-to-point communication) between processes. It could be related to a conflict in the communication layer.

Figure 13 shows a time profile graph, a breakdown of entry method activity over time, summed across all processors. In the ideal case of NAMD execution, the iterations should overlap and the valleys would disappear. The valleys indicates a low CPU load, probably because of communication delays; in this case roughly the 40% of one iteration the CPUs are underused and as a consequence, losing performance.

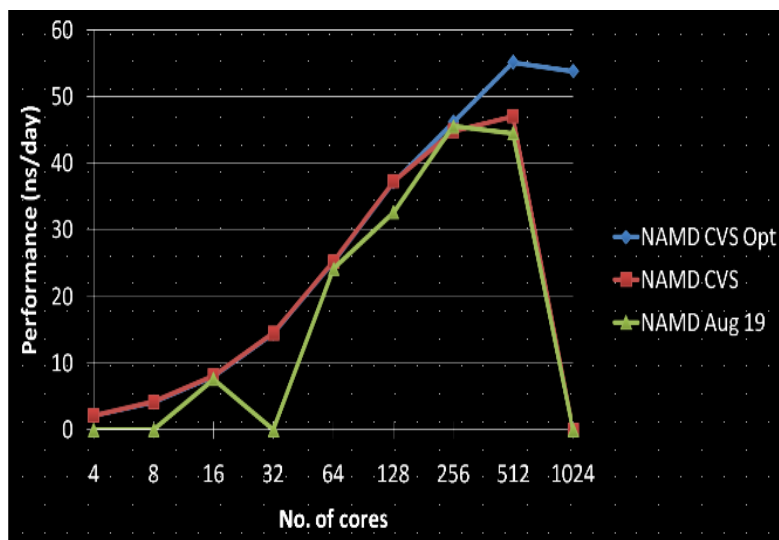


Figure 9 – NAMD runs comparison for Marenostrum production version (green) , the latest CVS version (red), and the latest CVS version optimized (blue).

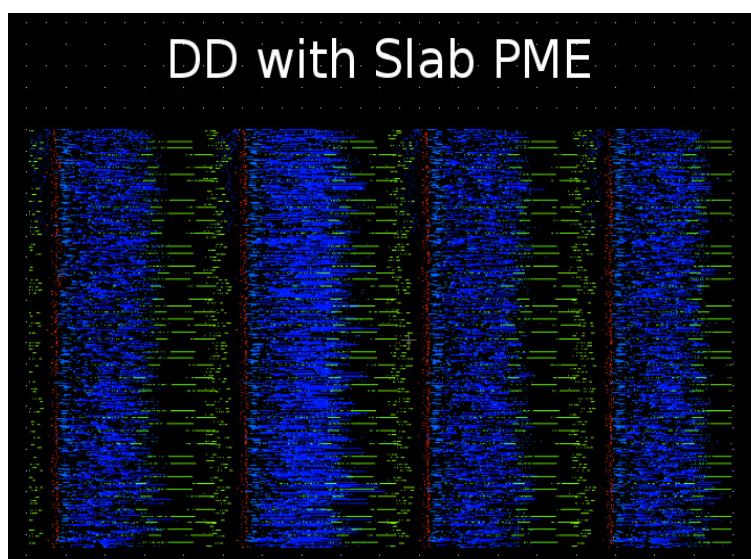


Figure 10 – Application behaviour changing the PME implementation: with Slab PME.

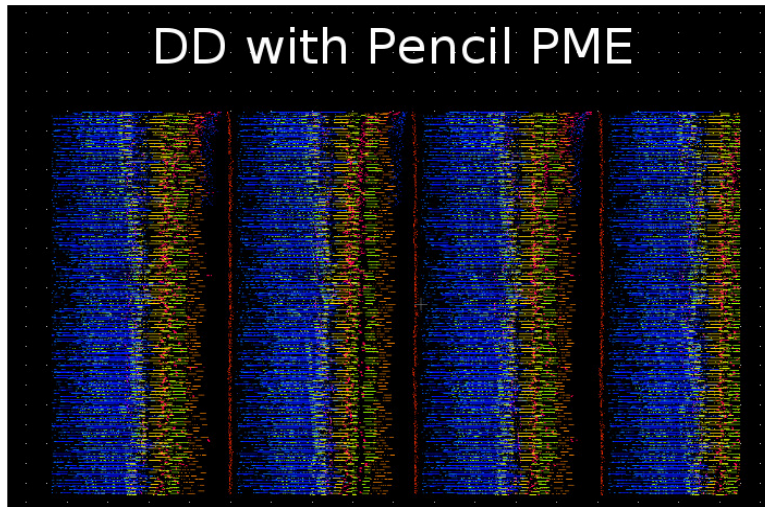


Figure 11 – Application behaviour changing the PME implementation: with Pencil PME.

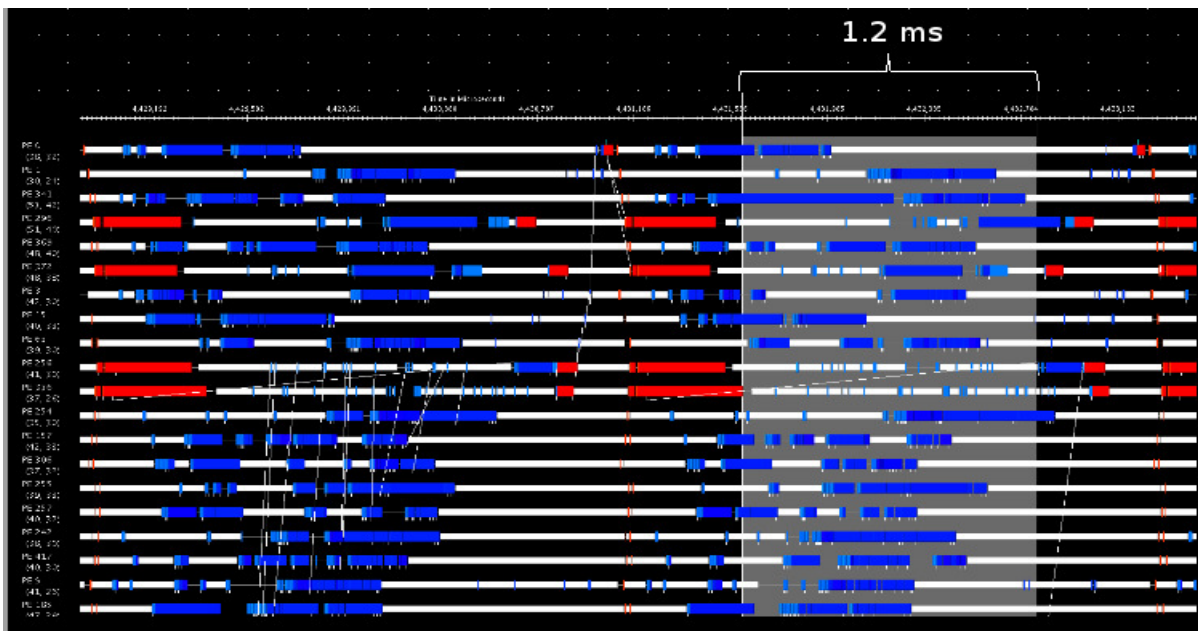


Figure 12 – Communication delay as showed by the Projections performance analysis tool.

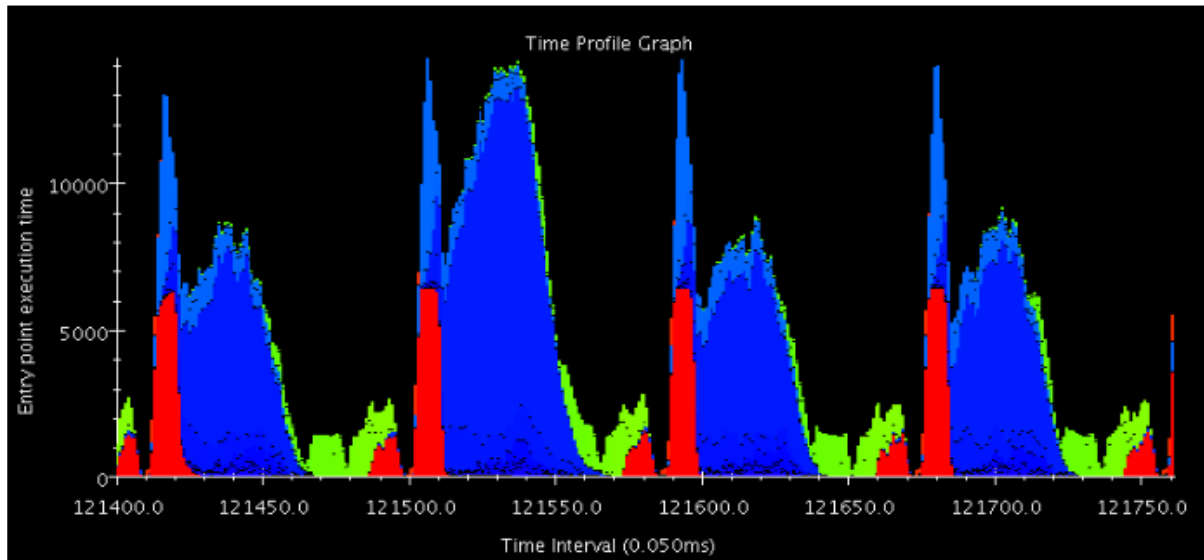


Figure 13 – NAMD's time profile graph showing four algorithm iterations.

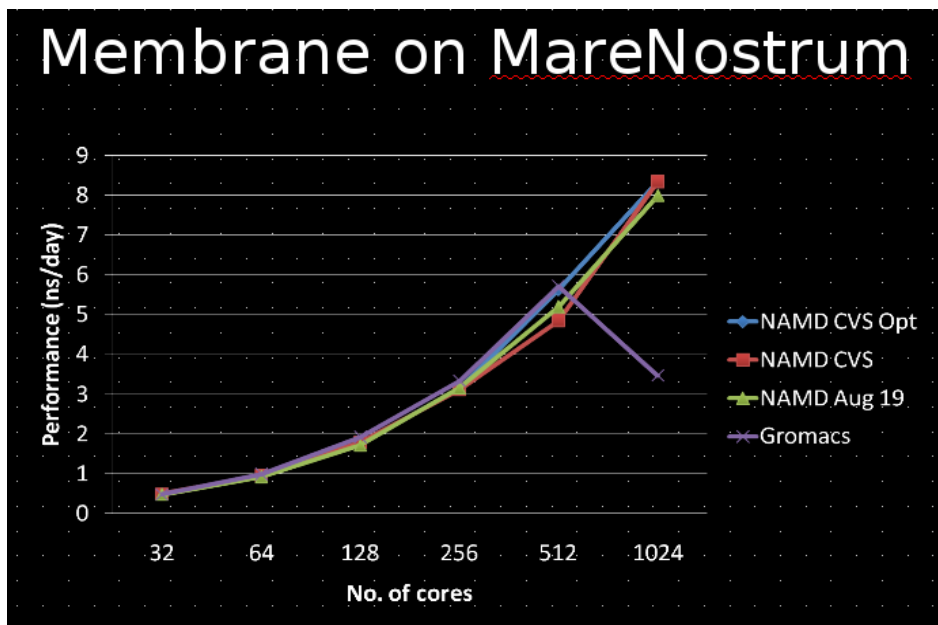


Figure 14 – NAMD vs Gromacs comparison for a large system (700K atoms). NAMD in red is the latest optimized version. The blue one is the previous production version.

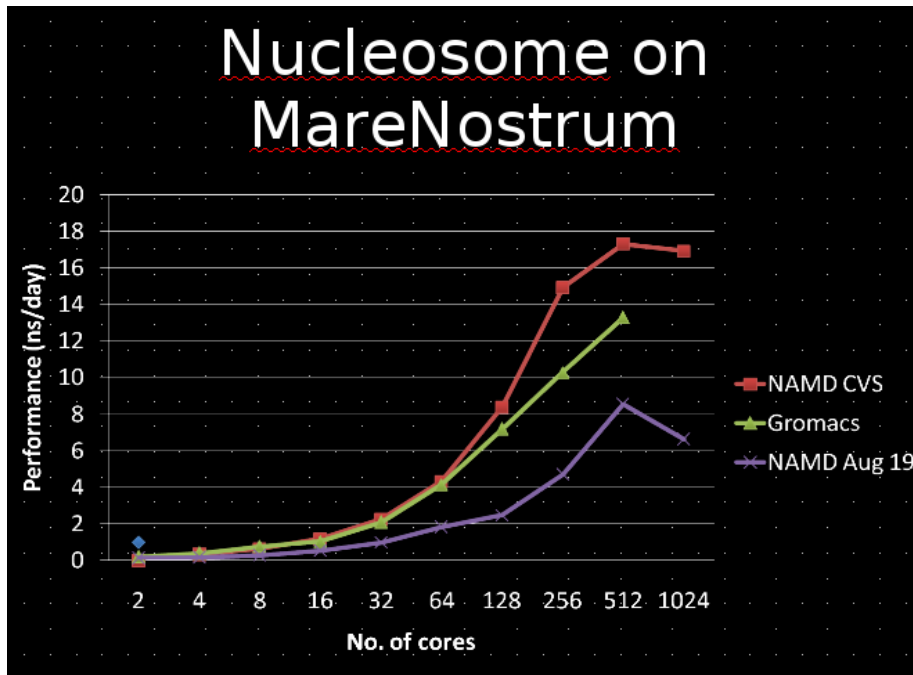


Figure 15 – NAMD vs Gromacs comparison for a medium-size system (145K atoms). NAMD in red is the latest optimized version. The blue one is the previous production version.

3.3 Conclusions and Future work

The testing of new features like the choice of PME (Particle Mesh Ewald) implementation has provided a way to increase the performance of large simulations at MareNostrum.

On the other hand, the lack of scalability beyond 1024 CPUs was related to communications delays only explained by the performance of the Myrinet and alien to the NAMD code.

The future work will be focused on the study of network contentions, so some further analysis of the network performance need to be done in order to reduce the scalability problems showed for large NAMD runs.

4 Enhancing Scalability: FZJ

This section describes the scalability enhancement effort done in FZJ. The project belongs to DECI 2008 and its acronym is CLOUD09, whose PI is Professor J. Schumacher. FZJ is the Home, Enabling and Execution site, running in Blue Gene/P. The project enabling started in January 2009 and the production stage is due to start in July 2009.

4.1 Introduction and motivation

The objective of the project is the investigation of cloud formation in moist convective turbulence. It combines aspects from three research areas, namely fluid turbulence, thermal convection and cloud physics. The combination of turbulence and thermodynamics makes moist convection a particularly difficult physical problem. The aim is to study moist convection in an idealized setting using piecewise linear thermodynamics by simplifying the thermodynamics but fully resolve the turbulence in the presence of phase changes.

The code used to carry out the simulations is written in FORTRAN90 and uses purely MPI for communication. The so-called pseudospectral method in a rectangular box of aspect ratio 16 is applied. This leads to flat Cartesian meshes with a targeted spectral resolution for the current project of $N_x \times N_y \times N_z = 4096 \times 4096 \times 512$ grid points which require a 2d domain-decomposition scheme due to the vast lateral resolution and the relatively small vertical extension. Horizontal periodic and vertical free-slip boundary conditions are used and the Boussinesq-Navier-Stokes equations are advanced in time by a second-order predictor-corrector scheme. For the Fast Fourier Transformation the p3dffft [12] package by D. Pekurovsky is used. This package is based on either the 1d ESSL (Engineering Scientific Subroutine Library) or FFTW library.

4.2 Aim of the enabling work

Since the code was already successfully running on the BG/P architecture, the main focus of the enabling work within this DECI project was the performance analysis and if necessary and possible performance enhancement of the application. BG/P specific runtime settings like mapping of MPI tasks to processors and the shape of the torus network as well as different processor grids have been tested and optimized with the WP5 enabling effort and are reported in the corresponding deliverable. A more detailed performance analysis in order to check for bottlenecks within the code was the object of the WP9 enabling work and is reported here.

4.3 Performance analysis of the code

The analysis was focused on two parts of the code, which might pose bottlenecks. These two parts were on one hand the application of symmetry constraints for the Fourier modes (done in the subroutine `symm`) and on the other hand the second-order predictor-corrector scheme (subroutine `predic` and `correc`) with the underlying Fourier Transformation using the p3dffft library based on the IBM ESSL for BG/P.

The performance analysis of the code was done using the SCALASCA software, developed at FZJ. Two grid sizes of $512 \times 512 \times 256$ and $2048 \times 2048 \times 1024$ grid points were chosen and tested on 2048 and 4096 cores on Jugene, respectively. The results of the latter run are shown in Figure 16.

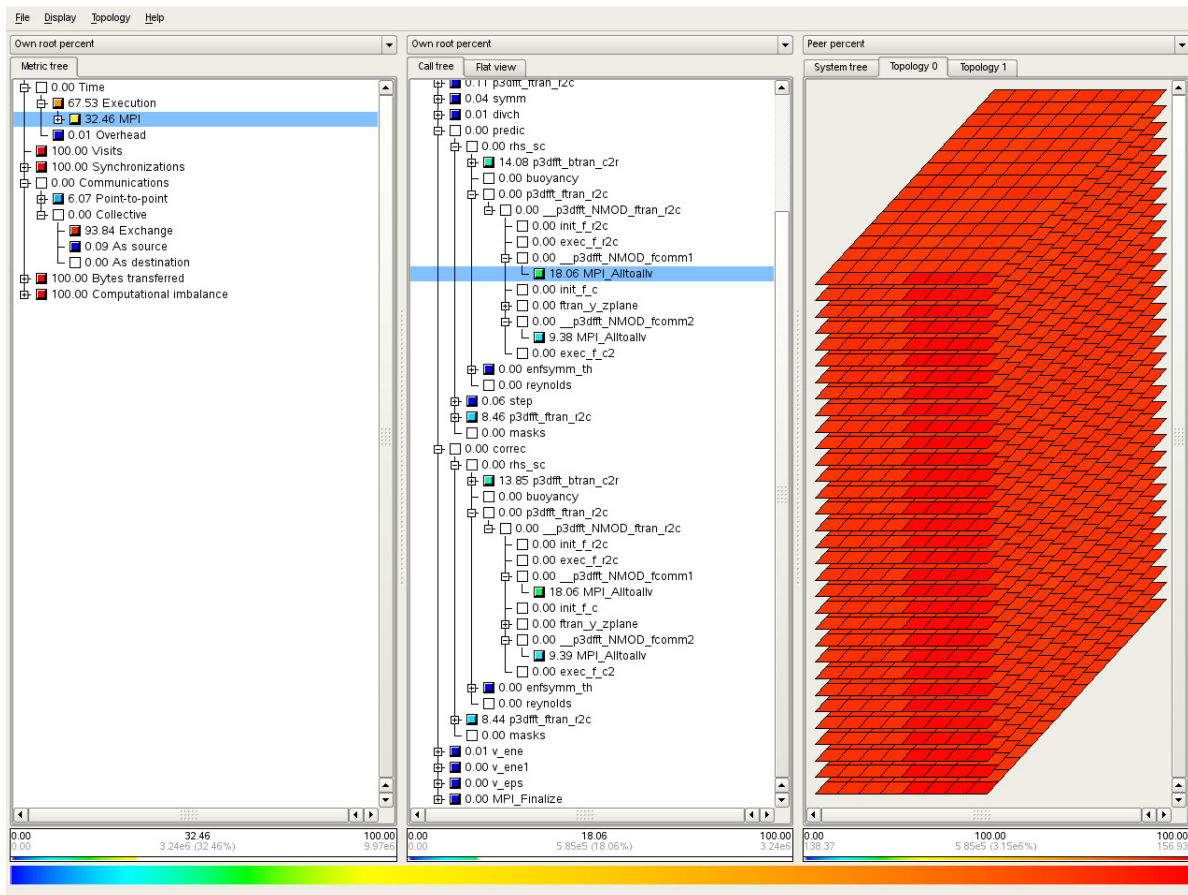


Figure 16 – Analysis of a 4096 core (1 rack) run on Jugene using a grid of 2048x2048x1024 grid points.

The left column shows the metric tree in units of own root percentage. The middle column depicts the call tree in the same units and the topology of the MPI_Alltoallv communication on the cores marked in column 2.

The analysis shows that the symmetry part of the code is negligible concerning time spent in communication (less than 1%, Figure 16) as well as spent in calculation (also less than 1%, not depicted). From Figure 1 it is evident that the most time-consuming part of the code is the communication of the Fast Fourier Transformation. The communication is based on MPI_Alltoallv and needs about 33% of the total runtime (marked in column 1). The analysis with the smaller grid size on 2048 cores shows basically the same results. This leads to the conclusion that the efficiency of the code depends mainly on the efficiency of the FFT library used, while the other parts are playing only a minor role, i.e. in order to improve the performance of the code the Fast Fourier Transformation has to be modified.

In the runs presented so far the FFT was based on the ESSL from IBM. Therefore, we tested the performance of the FFTW library in comparison with ESSL, since both can be used with the p3dff package. It turned out that the ESSL is still faster than the FFTW which therefore cannot be used as an alternative. Furthermore, version 2.2 and 2.3 of the p3dff package based on ESSL were compared with respect to the runtime, in response to a study by William Gropp and co-workers [13] at ANL (Argonne National Laboratory). They observed on a BG/L runtime differences of up to 10%. Our test was a forward-backward transformation of $f(x,y,z) = \cos(x) \cdot \cos(y) \cdot \cos(z)$. Two test cases were studied, namely $N_x = N_y = 4096, N_z = 1024$ and $N_x = 1024, N_y = N_z = 4096$. Both cases were tested with 2048 MPI tasks on 512 BG/P nodes. The measured runtime does not differ by more than about 2% among all cases.

Therefore, in order to further improve the performance the FFT part must be modified directly. One day of this year's first DEISA Training Courses in January at FZJ was devoted especially to porting and optimization of codes to the BG/P architecture. Professor

Schumacher, the PI of the project, attended the training and collaboration with a specialist of IBM was started with the aim to modify and optimize the FFT part. The work is currently ongoing and further analysis and tests will be performed.

5 Work plan for the Next 12M period

During the next period, two DECI calls are foreseen. The projects accepted will chiefly determine the work to be done in Enhancement. However, there is a very appealing line of work for Enhancement related to parallel hybridization.

The procedure will be again the following:

- A group of applications will be enabled to run in the different architectures available in DEISA2.
- A subset of these applications will be selected for a deeper study of performance passing basic tests of speed-up.
- A subset of this subset is selected to pass to Enhancement, depending on many variables, ranging from PI and developers commitment or scientific/societal interest up to real possibility of improvement or amount of effort foreseen.
- Enhancement process starts.
- *A posteriori* performance analysis.