



CONTRACT NUMBER RI-222919

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

European Community Seventh Framework Programme
RESEARCH INFRASTRUCTURES
Integrated Infrastructure Initiative

Maintenance of the DEISA Benchmark Suite
in the First Year

Deliverable ID: DEISA2-D7-2.2
Due date: April 30th, 2009
Author: Michele Weiland, UEdin-EPCC

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 Content

Table of Content.....	1
List of Figures	2
List of Tables.....	2
1 Introduction	3
1.1 Executive Summary.....	3
1.2 References and Applicable Documents.....	3
1.3 Document Amendment Procedure.....	3
1.4 List of Acronyms and Abbreviations.....	3
2 Progress	5
2.1 IFS	5
2.2 Usage	5
3 Porting to New Platforms	6
3.1 IBM Power6 – vip	6
3.2 IBM BlueGene/P - Babel.....	6
3.2.1 Architecture	6
3.2.2 Porting and Performance	6
3.2.2.1 NAMD.....	7
3.2.2.2 IQCS	9
3.2.2.3 CPMD.....	11
3.2.2.4 QuantumESPRESSO.....	11
3.2.2.5 GENE.....	12
3.2.2.6 PEPC.....	14
3.2.2.7 BQCD	15
3.2.2.8 SU3_AHIGGS	17
3.2.2.9 NEMO	18
3.2.2.10 RAMSES	20
3.2.2.11 GADGET.....	21
3.2.2.12 Fenfloss.....	23
3.2.3 Summary of Results.....	23
4 DECI CPU conversion factors.....	24
4.1 Conversion factors based on Benchmark Suite	24
4.2 Example calculation	25
5 Website	26
5.1 Login procedure.....	26
6 Future Work.....	27
7 Conclusion.....	28

List of Figures

Figure 1 – NAMD performance using the 1GND dataset (starting at 64 cores).....	8
Figure 2 – Parallel speedup for NAMD using the 1GND dataset (relative to 64 cores).....	8
Figure 3 – Performance of IQCS with the 32 Qubits dataset (starting at 256 cores). The graph compares IQCS in VN, DUAL and SMP modes on Babel.....	9
Figure 4 – Parallel speedup (from 256 to 4096 cores) for IQCS – Babel results are from a run in VN mode.	10
Figure 5 – Performance of QuantumESPRESSO (starting at 128 cores).	11
Figure 6 – Parallel speedup for QuantumESPRESSO (relative to 128 cores).	12
Figure 7 – Performance of GENE on HECToR and Babel (from 512 cores).....	13
Figure 8 – Parallel speedup for GENE (relative to 512 cores).....	13
Figure 9 – Performance of PEPC with the 5 million particle dataset (starting at 256 cores). .	14
Figure 10 – Parallel Speedup for PEPC with the 5 million particle dataset (relative to 256 cores).	15
Figure 11 – Performance of BQCD using a 24^3*48 lattice (from 64 cores).	16
Figure 12 – Parallel speedup for BQCD on HECToR, HLRB II and Babel (relative to 64 cores).	16
Figure 13 – Performance of SU3_AHIGGS using a 256^3 lattice (from 64 cores)	17
Figure 14 – Parallel speedup for SU3_AHIGGS (relative to 64 cores).	18
Figure 15 – Performance of NEMO on HECToR and Babel (from 128 cores).....	19
Figure 16 – Parallel speedup for NEMO (relative to 128 cores).	19
Figure 17 – Performance of RAMSES on HECToR, HLRB II and Babel (from 64 cores). ...	20
Figure 18 – Parallel speedup for RAMSES (relative to 64 cores).	21
Figure 19 – Performance of GADGET on HECToR, HLRB II and Babel (from 256 cores)..	22
Figure 20 – Parallel speedup of GADGET (relative to 256 cores).	22
Figure 21 – New login procedure to access the download area for the DEISA Benchmark Suite.....	26

List of Tables

Table 1 – Details of Babel's system specification.	6
Table 2 – List of reference datasets and minimum core counts to be used across all platforms.	24
Table 3 – List of runtimes and core counts per application on both the reference and the target platforms.	25

1 Introduction

1.1 Executive Summary

- The work undertaken by Task 7.2 “DEISA Benchmark Suite Maintenance” of WP7 between PM6 to PM11 was focussed on porting the Benchmark Suite applications to a new architecture, proposing a procedure for the contribution to the DECI CPU conversion factors and improving the website.
- The DEISA Benchmark Suite Maintenance task is complementary to the PRACE benchmarking activity and thus communication and cooperation between the two tasks was encouraged in order to avoid duplication of effort.
- Section 2 of this report summarises general progress of task T7.2 and highlights sources of external usage for the Benchmark Suite.
- In Section 3, the new architecture is described in detail, together with porting and performance reports for all the applications inside the Benchmark Suite.
- Section 4 outlines the proposal for using the DEISA Benchmark Suite performance results to contribute to the calculation of the CPU conversion factors used within DECI.
- In Section 5, the improvements to the Benchmark Suite website are described, namely the introduction of a registration step prior to downloading the applications.
- Sections 6 and 7 round off the document by outlining future work and summarising the work accomplished in PMs 6 to 11.

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] DEISA Benchmark Suite web pages: <http://www.deisa.eu/science/benchmarking>
- [5] BG/P Babel service: <http://www.idris.fr/comp/scal/bluegene/babel>
- [6] “Initial Report on the DEISA Benchmark Suite”, DEISA deliverable DEISA2-D7-2.1
- [7] European Network of Excellence on High Performance and Embedded Architecture and Compilation: <http://www.hipeac.net>

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

BG/P	IBM’s BlueGene/P architecture
CPMD	Car-Parinello Molecular Dynamics application
CSC	CSC – IT Centre for Science Ltd
DECI	DEISA Extreme Computing Initiative
DEISA	Distributed European Infrastructure for Supercomputing Applications
DUAL	Dual Core mode on BG/P
ECMWF	European Centre for Medium Range Weather Forecast

EPCC	The Supercomputing Centre of the University of Edinburgh
FZJ	Forschungszentrum Jülich GmbH
HECToR	High-End Computing Terascale Resource, the UK's national HPC service
HLRB II	Höchleistungsrechner in Bayern, LRZ
HPC	High Performance Computing
IBM	HPC hardware vendor
IDRIS	Institut du Développement et des Ressources en Informatique Scientifique
IFS	Integrated Forecast System
LRZ	Leibniz Rechenzentrum
NAMD	Nanoscale Molecular Dynamics package
RZG	Rechenzentrum Garching
SGI	Silicon Graphics, HPC hardware vendor
SMP	Shared Memory Processing mode on BG/P
VN	Virtual Node mode on BG/P
WP	Work package

2 Progress

In the past six months (PM 6 to PM12 of DEISA2), progress was made on three different aspects: firstly, the Benchmark Suite was ported to a new architecture; secondly, a process to use the Benchmark Suite for the calculation of the DECI CPU conversion factors was developed; and finally, the website was improved to allow for user registration.

The Benchmark Suite has already been ported to a set of different architectures, such as IBM Power, Cray XT4 and SGI Altix. The remaining key architecture that makes up a significant part of the DEISA infrastructure (with three systems - at IDRIS, FZJ and RZG) is the BlueGene/P. As systems from both RZG (IBM Power6) and FZJ (IBM Power4) were already used in the past, it was decided to port the Benchmark Suite to IDRIS's Babel system. Details of the porting effort and the system's performance can be found in Section 3.

One subtask listed in the Description of Work is the development of procedure that allows the use of the DEISA Benchmark Suite to help calculating the DECI CPU conversion factors. Section 4 outlines the proposed procedure together with an example calculation.

The DEISA Benchmark Suite website [4] has been improved to allow for user registration prior to download of the Suite. The main motivation behind implementing this registration step is to gather information on the usage of the Suite and to thus generate access and download statistics.

2.1 IFS

The Integrated Forecasting System (IFS) is being integrated to the DEISA Benchmark Suite as part of the eDEISA project. The application will be available for download from the website by the end of April 2009 - sample results on different architectures will be made available.

2.2 Usage

The CPMD framework that is part of the DEISA Benchmark Suite was used as a starting point for mixed-mode performance analyses on this application undertaken in DEISA2 WP9 ("Enhancing Scalability"). The reasons stated for using the DEISA Benchmark Suite in this case were ease execution and result verification, as well as performance comparison between various architectures.

The European HiPEAC Network of Excellence [7] have asked for permissions to use the NAMD input datasets from the Benchmark Suite for their own application suite, giving full credit to DEISA.

Finally, CSC intends to be using part of the DEISA Benchmark Suite for internal benchmarking of their systems in the future.

3 Porting to New Platforms

3.1 IBM Power6 – vip

Deliverable D.7.2.1 (“Initial Report on the DEISA Benchmark Suite” [6]) summarised work that had been undertaken to port the Benchmark Suite to the IBM Power 6 (*vip*) at RZG. Some of the performance results that were achieved could not be explained and further investigations were said to be necessary. It was ultimately decided to postpone the rerun of the Benchmark Suite by DEISA on this system for a few months, as IBM was applying upgrades to the system between PM 6 and 12.

In order to avoid duplication of work and the need to rerun the Benchmark Suite more than once, this work will now be undertaken in year 2 of the project.

3.2 IBM BlueGene/P - Babel

The following paragraphs give an introduction to the BlueGene/P architecture and report the performance of the Benchmark Suite codes on Babel.

3.2.1 Architecture

The BlueGene architecture is based on a philosophy of achieving high performance with low power consumption. This is achieved by packing together a large number of low clock speed CPUs and using low latency, high bandwidth networks as well as lightweight kernels to complete a massively parallel system that caters well for highly scalable applications. Table 1 summarises that hardware specification for Babel, the BG/P at IDRIS.

Babel – IBM BlueGene/P				IDRIS
10240 compute nodes with 4 core per node				
<i>Processors</i>	<i>Cache</i>	<i>Torus Network</i>	<i>Collective Network</i>	<i>OS</i>
PowerPC 450 850 MHz 2GB RAM per node	L1: 32 KB per core L2: 14 stream prefetching L3: 8MB	Bandwidth: 5.1GB/s Latency: 100-800 ns	Bandwidth: 5.1GB/s Latency: 5µs	Linux (lightweight kernel on compute nodes)

Table 1 – Details of Babel's system specification.

The low clock speed CPUs and the “Double Hummer” floating point units are characteristic for the BlueGene architecture. For the BG/P, the IBM compiler option `-qarch=450d` generates instructions for two floating point pipes (Double Hummer) – however not all applications benefit from this. The option `-qarch=450` generates instructions for a single floating point pipe only and might show better performance.

The BG/P also gives the option to set a LoadLeveler keyword for the network (`bg_connection`) and an environment variable `BG_MAPPING` to map tasks to physical cores. Under 512 nodes (2048 cores), MESH is the only option for the network – above this, it is possible to use the TORUS. `BG_MAPPING` is set to XYZT by default, where X,Y and Z are the torus coordinates of the nodes and T is the core number within a node. Permutations of this setting (e.g. to TXYZ) might show increased performance.

3.2.2 Porting and Performance

This section reports on the porting issues and the performance of each application. The performance of the applications on the BG/P architecture is compared (mainly) to the XT4.

The performance graphs show the runtimes on Babel normalised against the runtimes on HECToR. In addition to the code performance, the parallel speedup is plotted for each application.

The ratio of the clock speeds between Babel (850MHz) and HECToR (2.8GHz) is 3.3 – codes are therefore expected to run approximately three times slower on Babel than on HECToR for the same number of cores. This does not take performance influencing factors such as memory or network into account. HECToR can perform two floating point operations per cycle (maximum 5.6 Gflops per core), whereas Babel can theoretically perform four (maximum 3.4 Gflops per core). This results in a peak performance ratio of 1.7. However it needs to be noted that this is only true when using the special double floating-point unit (Double Hummer mode), otherwise the theoretical peak performance for Babel is halved.

The BG/P architecture can pose problems for memory hungry applications – in VN mode, each core only has access to 500MB of memory. For such applications, it is necessary to underpopulate the nodes and run in DUAL (2 cores per node, 1GB memory per core) or SMP mode (1 core per node, 2GB memory per core).

It is important to keep in mind that the performance results reported here are indicative of an application's performance on a given system using the provided datasets. They are not necessarily indicative of an application's general performance – the use of different datasets might result in considerable different overall performance.¹

3.2.2.1 NAMD

NAMD was relatively straightforward to port to Babel. A number of changes to the build scripts were necessary for the source to compile correctly², but no changes to the source were required.

Figure 1 shows the performance of NAMD using the 1GND dataset. The runs shown here have all been executed in DUAL mode using a MESH network configuration. NAMD was compiled with the Double Hummer option, which proved to be beneficial to the code's performance.

The overall performance ratio between HECToR and Babel is approximately a factor of 3, and thus in the expected range. Figure 2 shows the parallel scaling of the same runs and it becomes obvious that the application does not scale beyond 256 cores. The scaling is slightly better on Babel than on HECToR, but it is generally poor.

In order to be able to run NAMD on large core counts and to achieve meaningful performance figures, changes to the input data sets need to be implemented. These changes could involve modifying the parallel decomposition, altering the communication tree or workload redistribution. In addition a newer version of NAMD (version 2.7), which based on initial tests shows increased scalability, could be used. Both options will be considered as part of a global revision of the Benchmark Suite content.

¹ Complete performance results for the applications are available at www.deisa.eu/science/benchmarking/results

² Many thanks to Michael Stephan from FZJ for providing the correct build configuration.

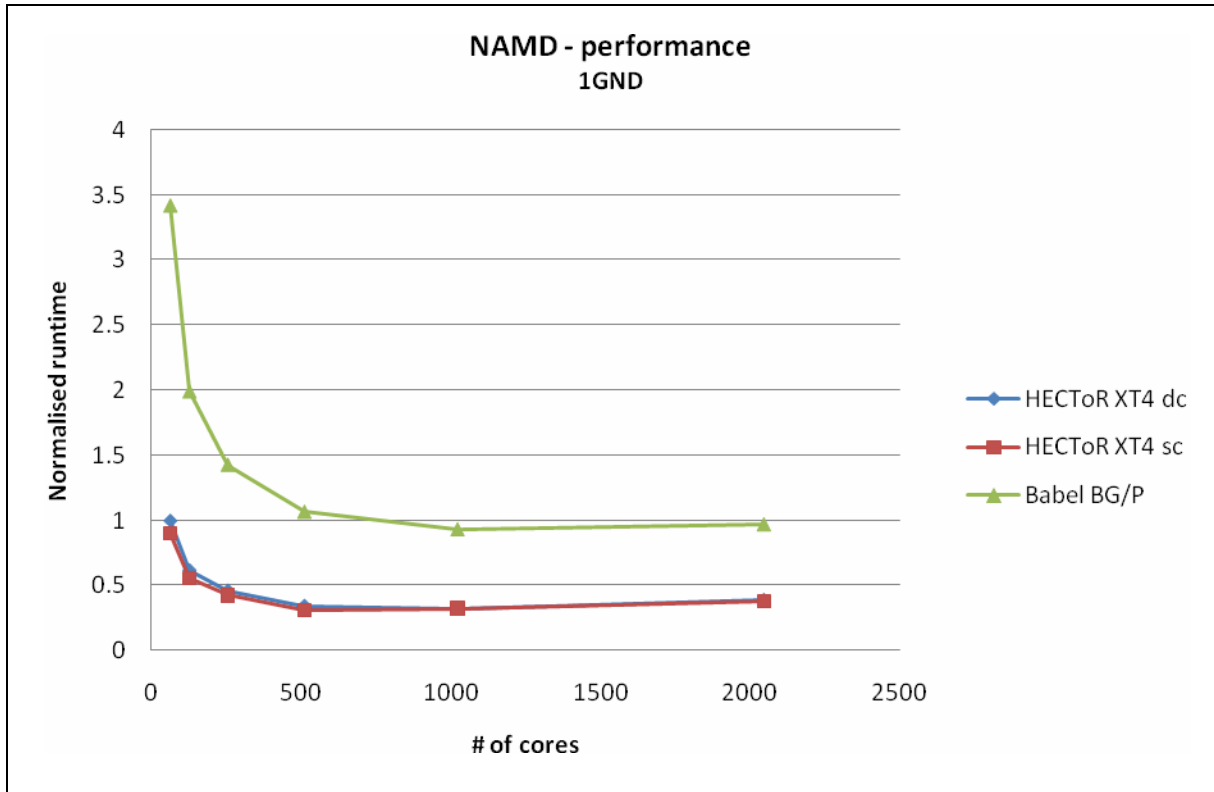


Figure 1 – NAMD performance using the 1GND dataset (starting at 64 cores)

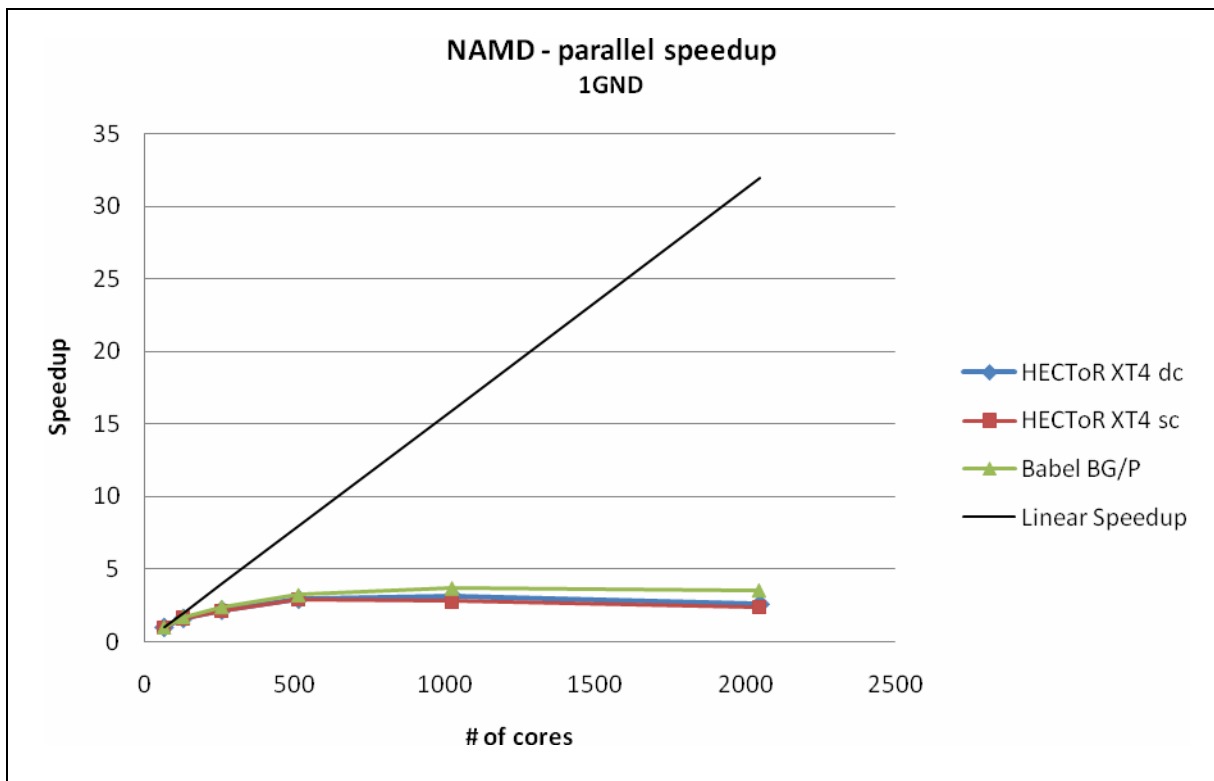


Figure 2 – Parallel speedup for NAMD using the 1GND dataset (relative to 64 cores)

3.2.2.2 IQCS

Porting IQCS to the BG/P architecture was straightforward and without problems. Initial tests showed that the code does not profit from the Double-Hummer and its overall performance was better using the `-qarch=450` option. Also, the TXYZ mapping (instead of the standard XYZT mapping) leads to a small increase in performance and the torus network clearly outperforms the tree network. The torus network was therefore used for runs with 512 and 1024 nodes.

Runs with 32 up to 34 Qubits have been performed. Figure 3 compares the normalised wallclock time of 32 Qubit runs on HECToR (dual core and single core) and Babel (VN, DUAL and SMP modes). The runs with 33 and 34 Qubits were performed using the SMP mode due to the memory demands of the code. Here, a minimum of 128 nodes (33 Qubits) and 256 nodes (34 Qubits) need to be used in order to have sufficient main memory.

The average ratio 2.5 of the timings between VN mode runs on Babel and HECToR DC is smaller than the ratio between the CPU clock speeds ($2.8\text{GHz}/850\text{MHz} = 3.3$). The reason for this is memory bandwidth. IQCS is very memory demanding - once the memory bandwidth is saturated the code does not profit from an increase in clock speed.

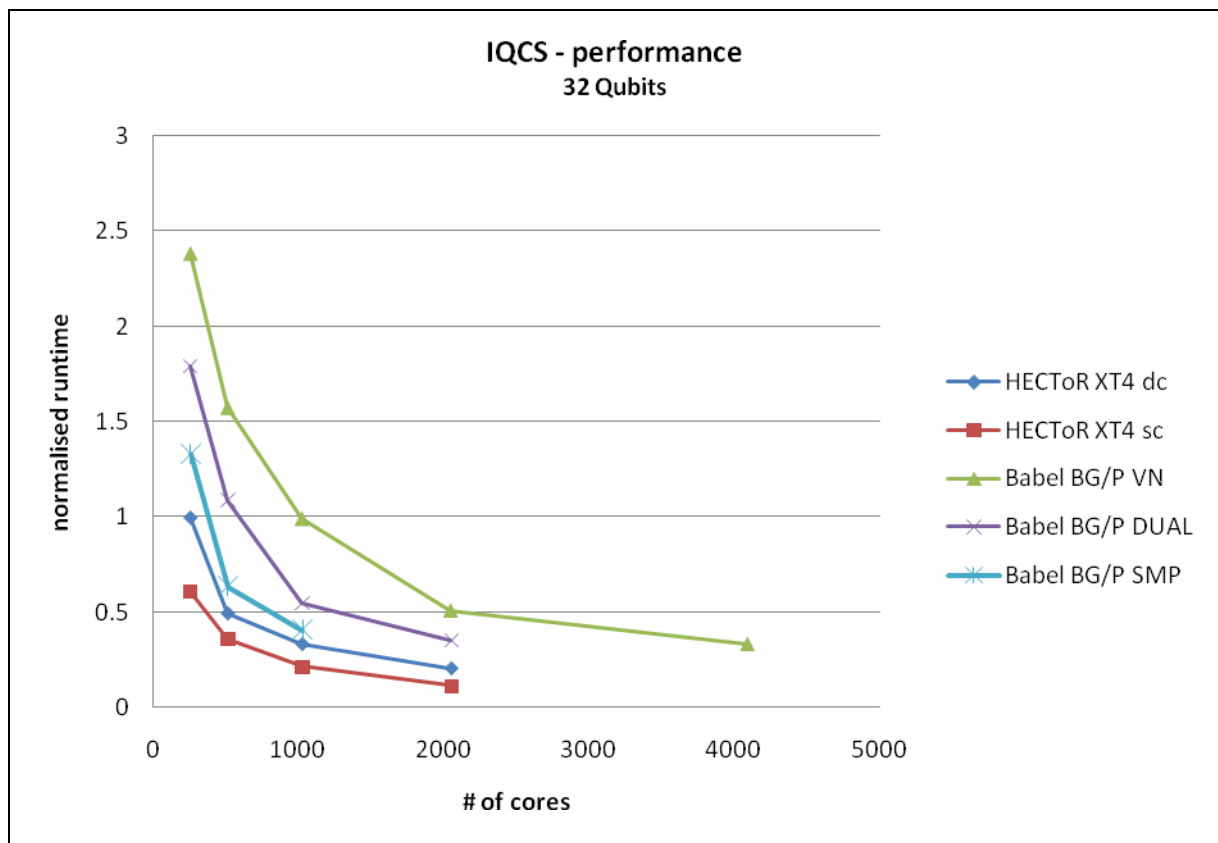


Figure 3 – Performance of IQCS with the 32 Qubits dataset (starting at 256 cores). The graph compares IQCS in VN, DUAL and SMP modes on Babel.

The code runs more efficient and faster (on average about 26% faster) in DUAL mode compared with VN mode. For example, in VN mode a 256 core run takes approximately 70 seconds while the same run in DUAL takes only about 52 seconds. The efficiency from 256 → 1024 cores is 60% and 81% in VN model and DUAL mode respectively. The efficiency in SMP mode is comparable to the one in DUAL mode. However the wallclock time (~39

seconds) on the same number of cores is again reduced with respect to the DUAL mode runs. The memory demand of IQCS and the memory bandwidth usable for each task are partly the reason for this behaviour. However, the communication plays a more important role in this case:

- The profile of the VN run reveals that the two most time-consuming routines are communication routines, which use up ~50% of the total wallclock time, while the Hadamard operation and the measuring of the spins take only about 24% of the total time. The communication in IQCS consists mainly of MPI_Sendrecv calls.
- In the DUAL mode run the two communication routines are still on top of the profile. However, their part decreases to ~45% compared with ~28% for the Hadamard operation and measuring of the spins.
- Finally, the actual Hadamard calculations are the most time-consuming step in the SMP run (23%), followed by the two communication routines. The Hadamard operation and the spin measurement use up 38% of the wallclock time together, which is equal to the top two communication routines.

The parallel speedup of IQCS on Babel in BN mode is comparable with the speedup achieved on HECToR (see Figure 4). Going from 1024 to 2048 cores, the TORUS instead of the MESH network was used on Babel, which explains the increase in the speedup.

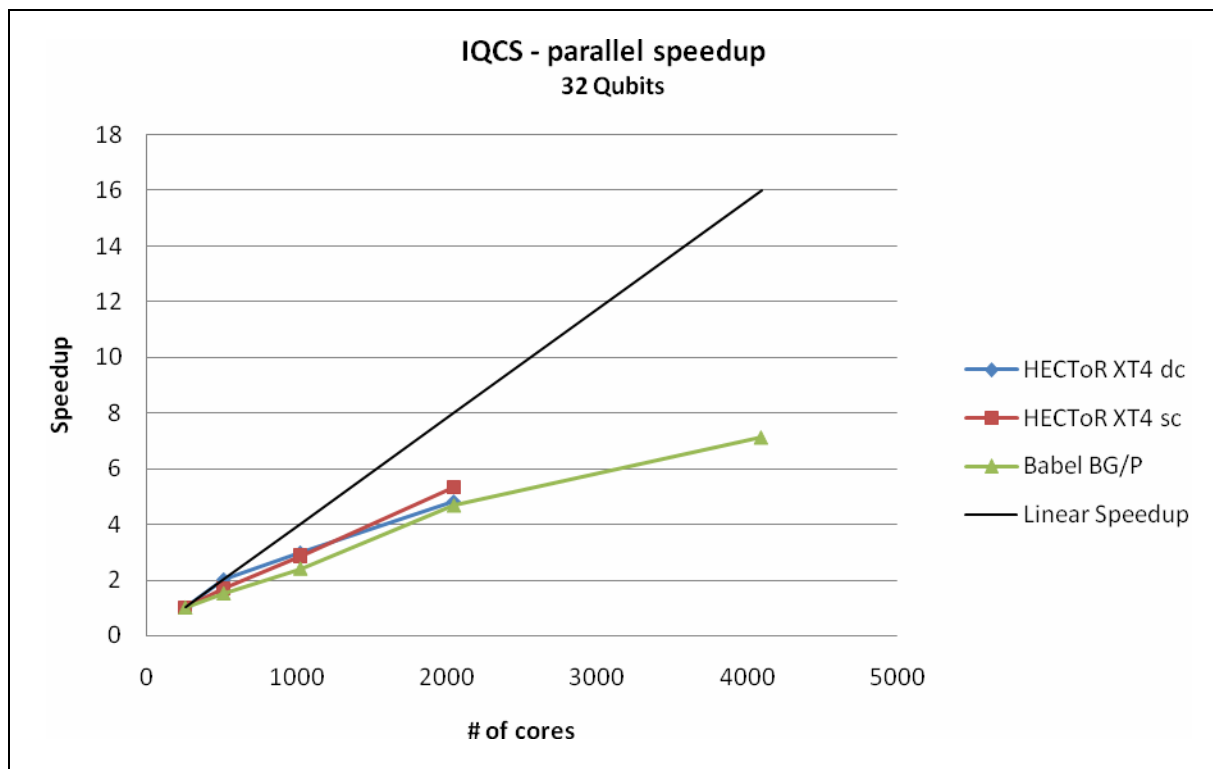


Figure 4 – Parallel speedup (from 256 to 4096 cores) for IQCS – Babel results are from a run in VN mode.

3.2.2.3 CPMD

The version of CPMD that the Benchmark Suite is currently set up for (version 3.11.1) does not run on Babel and executions fail due to a memory bug in this version of the code. The exact cause of the bug is unclear, but it is most likely an issue with dynamic memory allocation – each processor reports the following error:

```
*****
PROCESSOR 239 ALLOCATION OF      12 WORDS OF MEMORY FAILED
*****
```

IDRIS had come across this problem in the past and suggested running a more recent version of the application. Indeed the memory problem disappears with version 3.13.2_01.

We currently do not have performance results on other platforms for the new version of CPMD and we therefore cannot show performance comparison graphs. In a next step, we will review the differences between the two code versions and most likely update the Benchmark Suite to version 3.13.2_01.

3.2.2.4 QuantumESPRESSO

QuantumESPRESSO was ported to Babel without encountering any difficulties. The largest dataset available in the Benchmark Suite was only run on 256 cores in DUAL mode and 512 cores in VN mode because of memory and scaling limitations. The performance analysis is therefore done using the AUSURF112 data. All the runs were performed in VN mode with the MESH network configuration. The code benefits from using the Double Hummer option.

Performance on Babel is good – the performance ratio compared with HECToR is approximately 2.5 (see Figure 5), which is due to the use of two floating point pipes. The parallel speedup on Babel is comparable with the speedup on HECToR (see Figure 6).

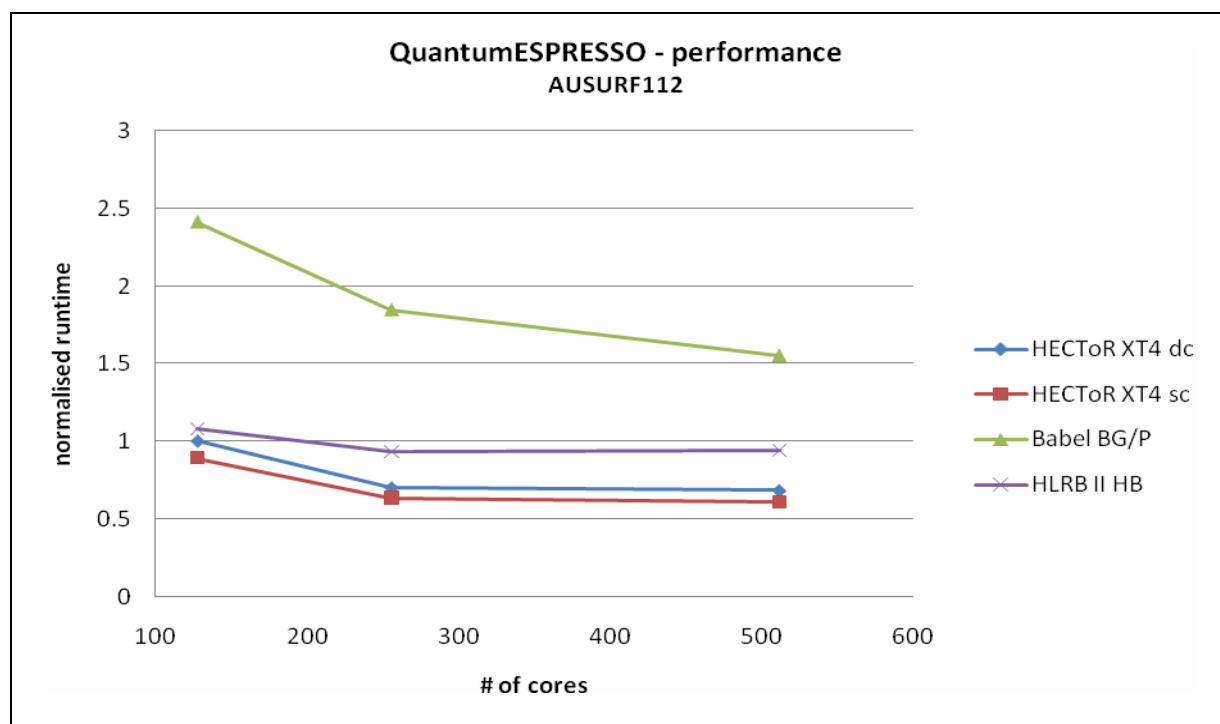


Figure 5 – Performance of QuantumESPRESSO (starting at 128 cores).

A new version of QuantumESPRESSO (with ScaLAPACK) will become available in the coming months. When reviewing the current content of the Benchmark Suite a move to this new version, which will most likely show increased performance and scalability, will be considered.

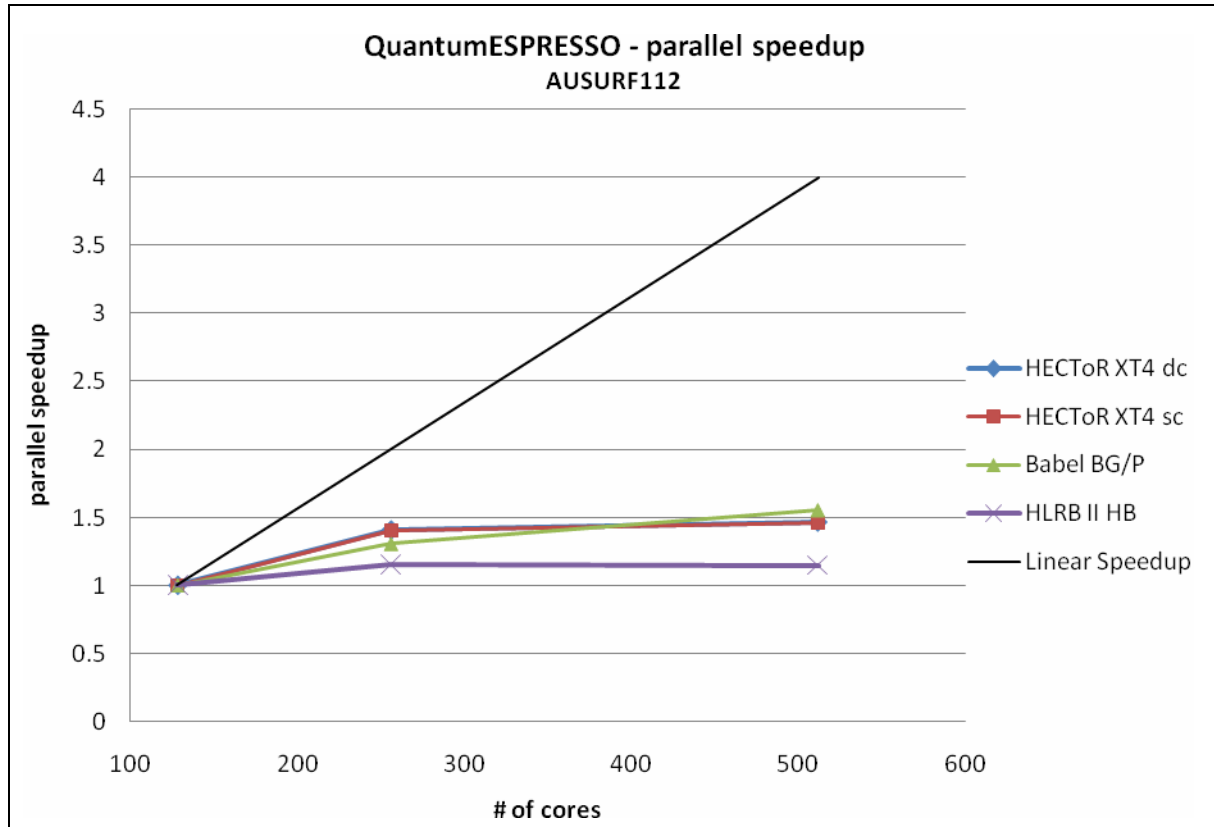


Figure 6 – Parallel speedup for QuantumESPRESSO (relative to 128 cores).

3.2.2.5 GENE

GENE had already been ported to the BG/P architecture (namely the BG/P at RZG) by the code authors. Thus the required porting effort was very small, the only stumbling block being the confusingly named `-DAIX` pre-processor option that was required also for the BG/P.

Another minor point concerned an incompatibility of the MPI module used with the IBM XL Fortran compiler. This was filed as a bug report with IBM, but there was a fairly simple workaround that allowed work to progress quickly.

The performance runs were done in VN mode using the MESH network. The Double Hummer option proved beneficial to code performance and thus the code was compiled using `-qarch=450d`. In VN mode, the 512 GB test case could not be run below 512 cores because of insufficient memory. Figure 7 shows GENE's performance on HECToR, HLRB II and Babel. GENE achieves very good performance on Babel (runs are only ~2.6 times slower than HECToR) and clearly benefits from the second floating point pipe.

GENE also shows good scalability, although the speedup on Babel drops very slightly from 1024 to 2048 cores (see Figure 8), compared to HECToR.

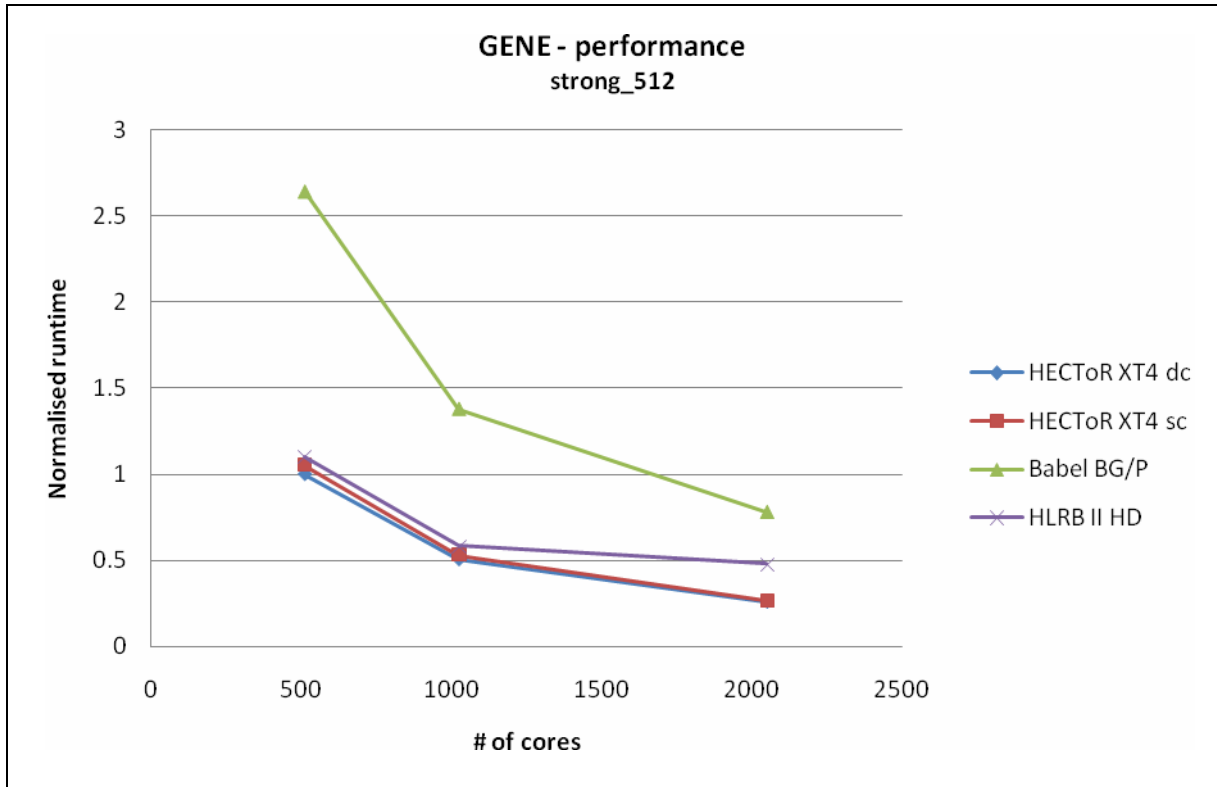


Figure 7 – Performance of GENE on HECToR and Babel (from 512 cores).

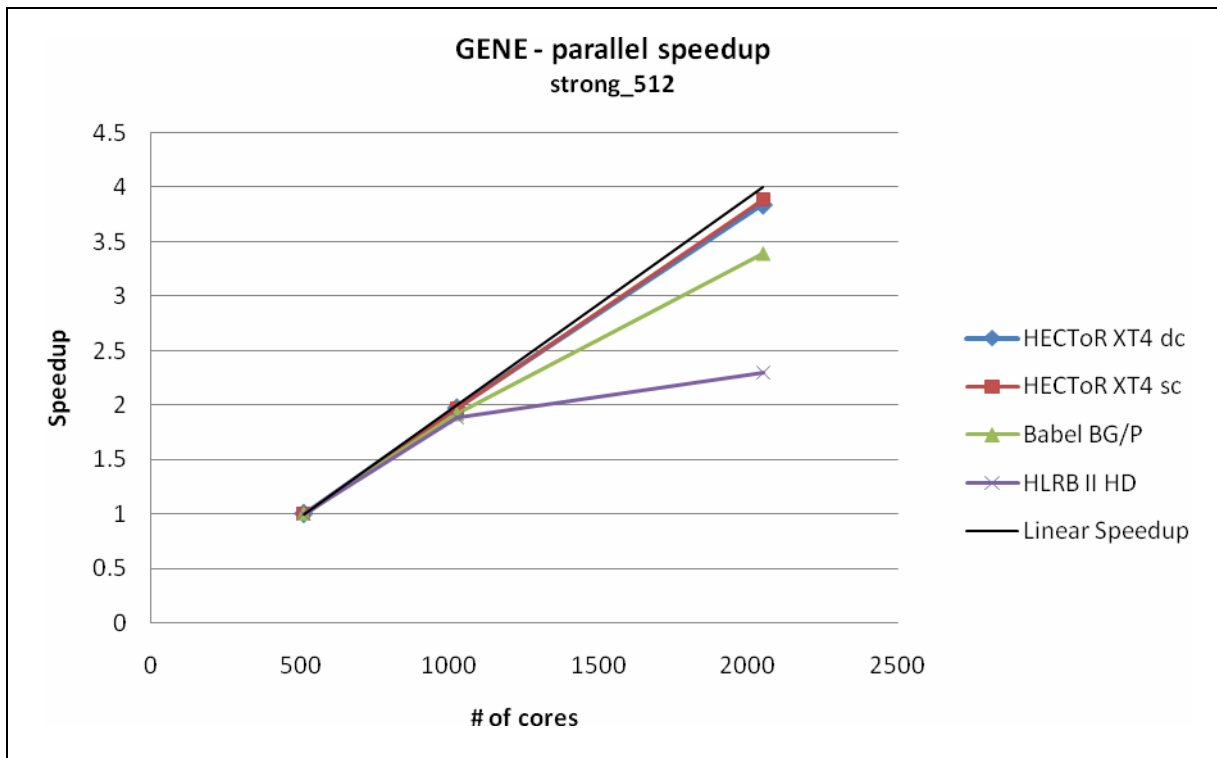


Figure 8 – Parallel speedup for GENE (relative to 512 cores).

3.2.2.6 PEPC

In order to facilitate the build of the PEPC benchmark on cross-compilation platforms, it was decided to change the compilation process when porting PEPC to Babel. The *configure* script was replaced by a standard *Makefile*. Following this, porting the application to the BG/P was straightforward.

The performance of the application benefits from enabling the “-qarch=450d” option (“Double-Hummer”). The current version of PEPC runs only up to 1024 cores (256 BG/P nodes in VN mode). The effect of the torus network on the performance could thus not be tested in VN mode, as only the tree network is available below 512 nodes.

Runs with 2, 3 and 5 million particles have been performed on 64, 128 and 256 nodes. Figure 9 shows the performance of the application with the largest dataset on Babel, comparing it to HECToR in both dual core and single core mode. The efficiency from 256 → 1024 cores on Babel (52.3%) is comparable to the corresponding runs on HECToR (55.8%). The efficiency from 512 → 1024 is however approximately 12% higher than the corresponding runs on HECToR. This behavior is probably due to cache effects. This difference in performance can also be seen in the parallel speedup graph (Figure 10), in which the speedup for Babel increases between 512 and 1024 cores.

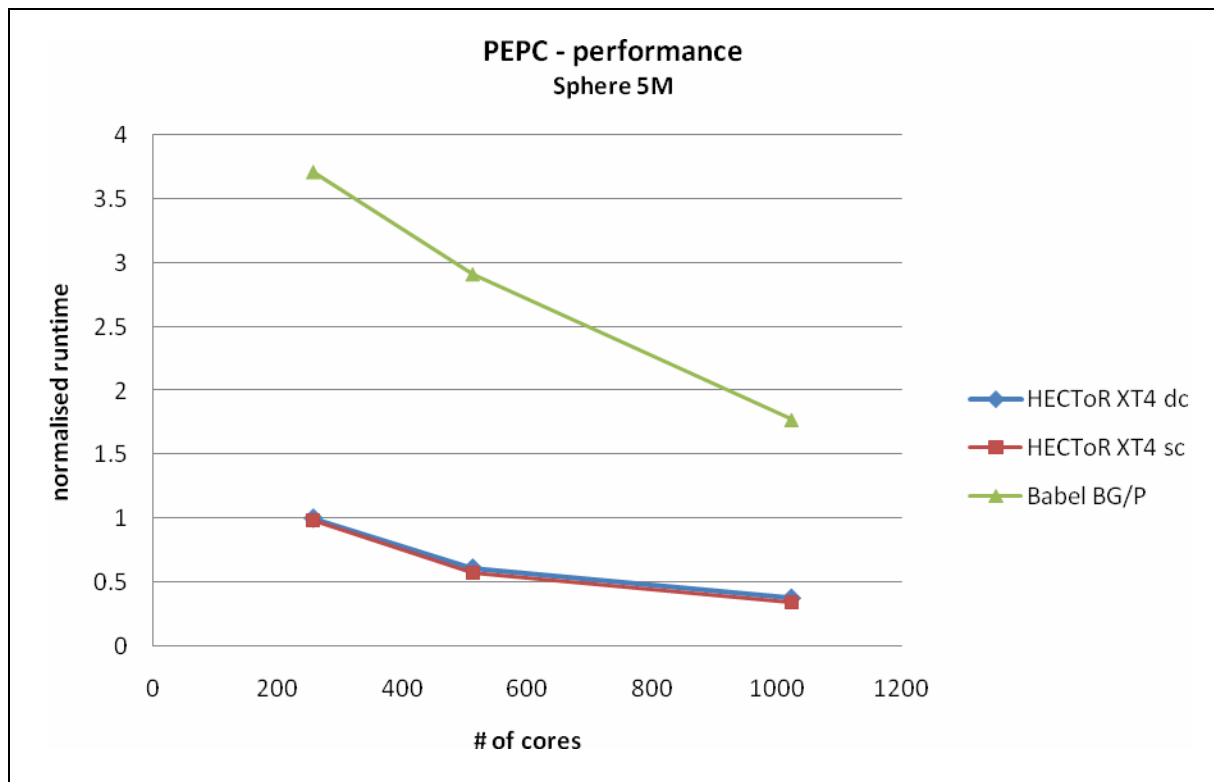


Figure 9 – Performance of PEPC with the 5 million particle dataset (starting at 256 cores).

While the ratios of the wallclock times for the runs on 256 cores between e.g. BG/P and Cray XT4 agree with the expected factor of about 4, the wallclock ratios between the runs on the different machines on higher number of cores become larger. The increase of the communication part is the reason for this behavior, because there are fewer particles per processor at higher core counts. Therefore, the clock speed becomes less important and the

network is more dominant. Since all runs are below one midplane, the mesh network must be used on the BG/P and the code cannot profit from the torus network.

The current version of PEPC does not run beyond 1024 cores. A new release of PEPC, which improved scalability, will shortly become available. In order to run PEPC on large core counts, it will be necessary to adopt the new version of the code as part of the Benchmark Suite.

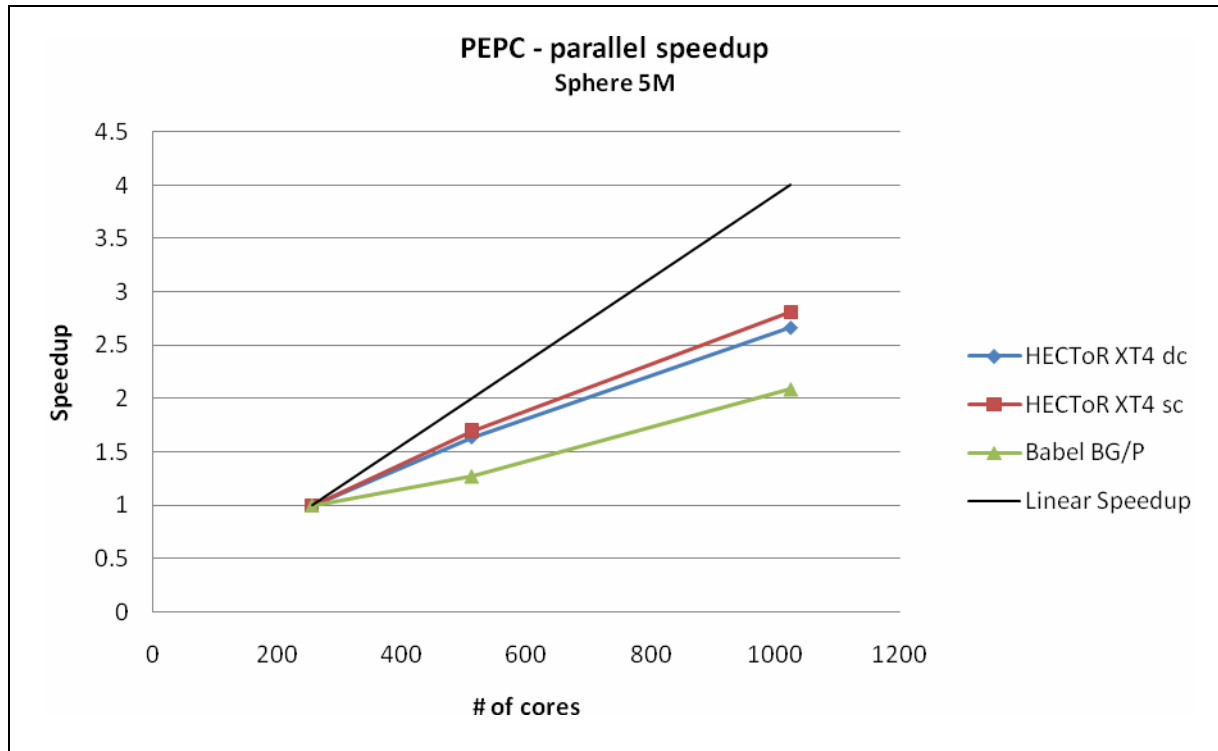


Figure 10 – Parallel Speedup for PEPC with the 5 million particle dataset (relative to 256 cores).

3.2.2.7 BQCD

BQCD was ported to Babel without any problems. Using the Double Hummer option as well as setting BG_MAPPING to TXYZ showed an increase in overall performance. The runs using the 24^3*48 lattice dataset were all performed in VN mode.

Figure 11 shows that the performance of BQCD on Babel (as compared to HECToR) is excellent, with the performance ratio being approximately the same as the peak performance ratio per core.

Figure 12 show the parallel speedup of the application – on Babel, BQCD continues to scale up to 4096 cores, whereas the scaling on HECToR, while still at a good level, drops of at 2048 cores.

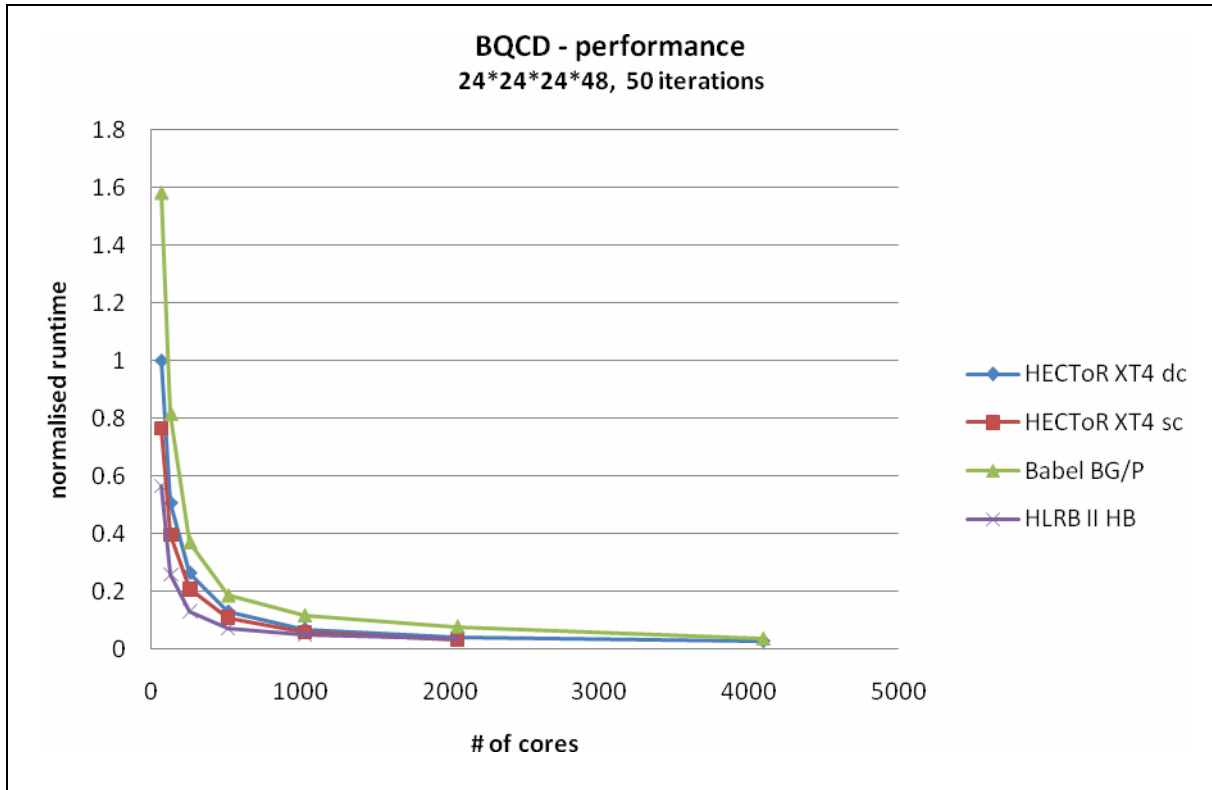


Figure 11 – Performance of BQCD using a 24³*48 lattice (from 64 cores).

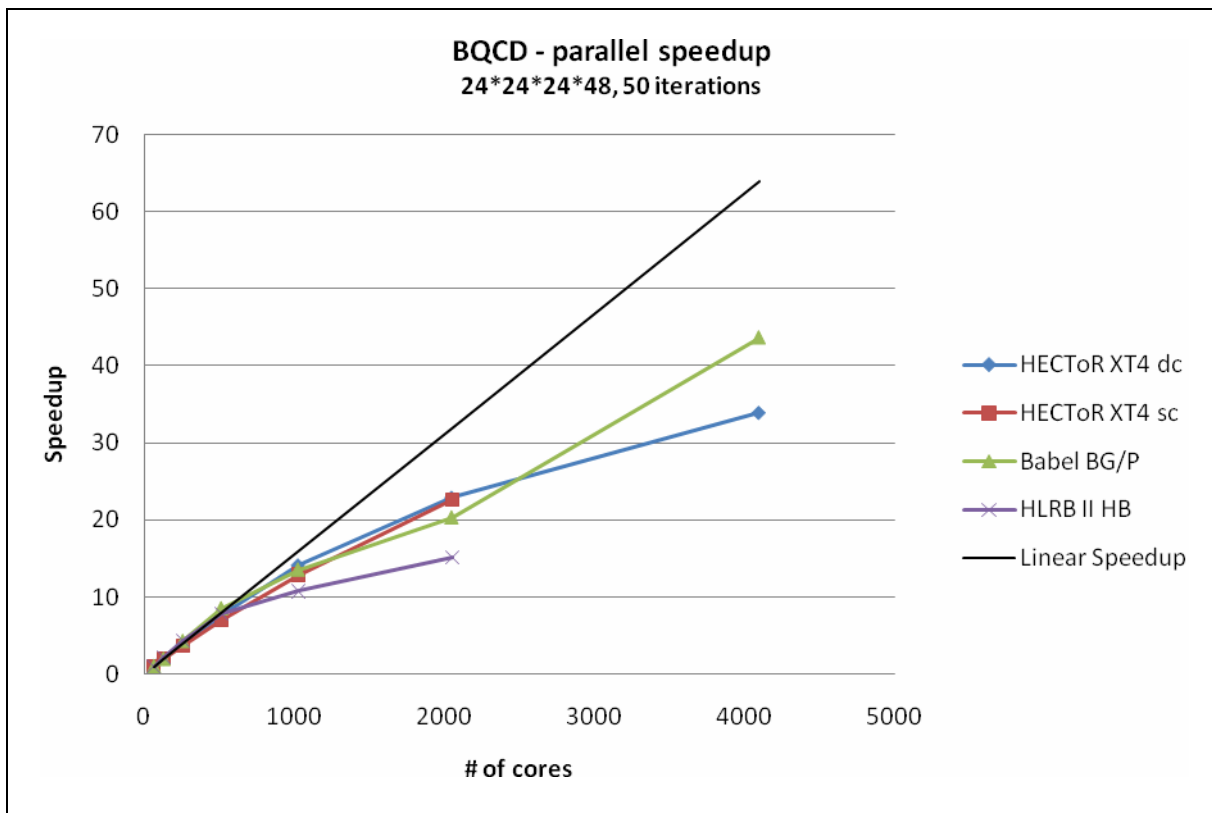


Figure 12 – Parallel speedup for BQCD on HECToR, HLRB II and Babel (relative to 64 cores).

3.2.2.8 SU3_AHIGGS

Porting SU3_AHIGGS to the BG/P architecture was unproblematic. The performance runs were done using the largest available dataset, a 256^3 lattice. All the runs were set up to use VN mode and the TORUS network (on more than 512 cores). The application showed increased performance when changing the mapping of tasks to physical cores from the default setting to BG_MAPPING=XYZ. The Double Hummer option was not beneficial to the performance of SU3_AHIGGS and thus the code was compiled with `-qarch=450`.

SU3_AHIGGS performs worse than expected on Babel – the code runs nearly 4.5 times slower on the BG/P than on HECToR (see Figure 13), whereas that performance ratio was expected to be closer to 3. The reasons for this drop in performance are not currently clear and will require further investigation.

Unlike the performance, the excellent scalability of the application is not a surprise. As can be seen in Figure 14, SU3_AHIGGS shows near linear parallel speedup on both platforms up to 2048 and 4096 cores.

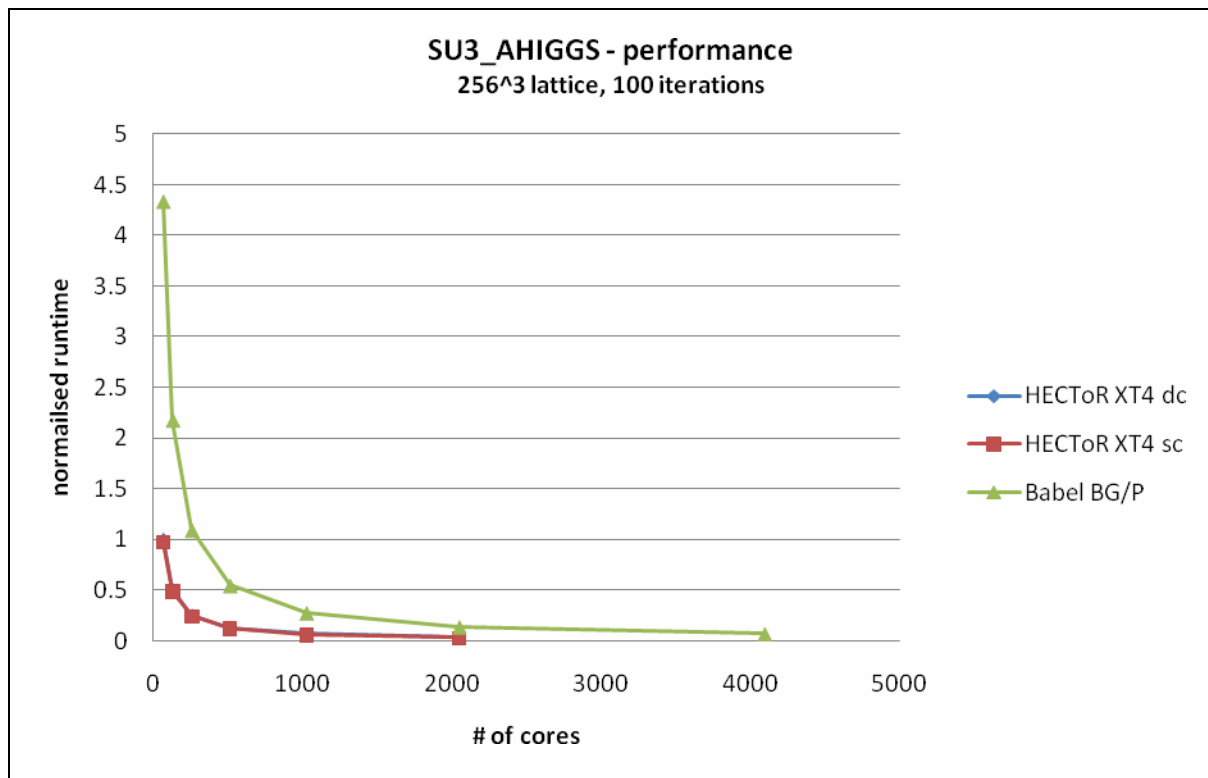


Figure 13 – Performance of SU3_AHIGGS using a 256^3 lattice (from 64 cores)

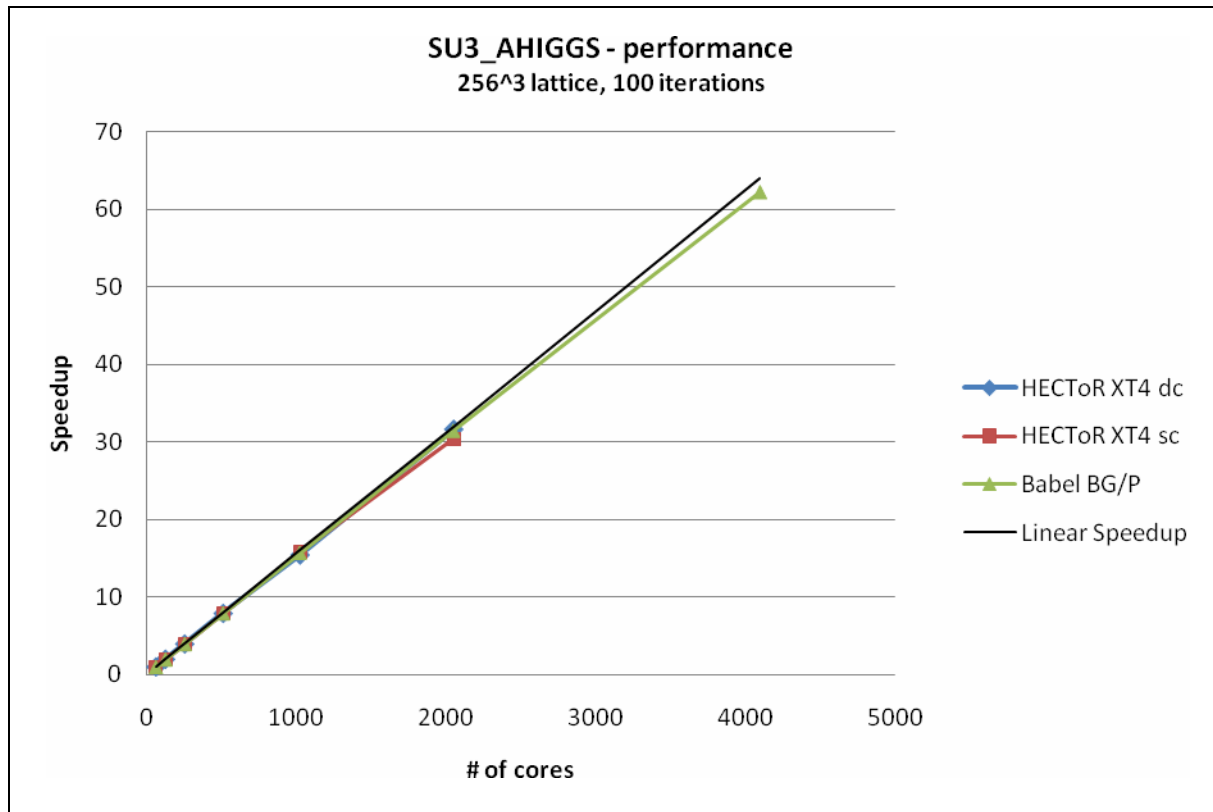


Figure 14 – Parallel speedup for SU3_AHIGGS (relative to 64 cores).

3.2.2.9 NEMO

NEMO had already been built on Babel, so no porting work was required. Figure 15 shows the performance of NEMO using the GYRE.50 test case (with no IO). The code was compiled without the Double Hummer option, as this did not benefit the performance. The runs up from 256 cores upwards were done in VN mode – on 128 cores it was necessary to use DUAL mode because of the memory limitations on the BG/P. The MESH network was used for all runs.

With GYRE.50, the performance ratio varies between 2 and 3 between Babel and Hector DC, which is what has been expected. On HECToR, the limit of the scalability of the code using this data set is between 1024 and 2048, whereas the parallel speedup on Babel continues up to 2048 cores.

NEMO is one of the only codes on the Benchmark Suite that can measure the impact of IO. On HECToR, the extra cost is varies between 2% on 64 cores and 30% on 1024 cores. On Babel however it is much higher, between 14% on 64 cores up to 50% on 1024 cores.

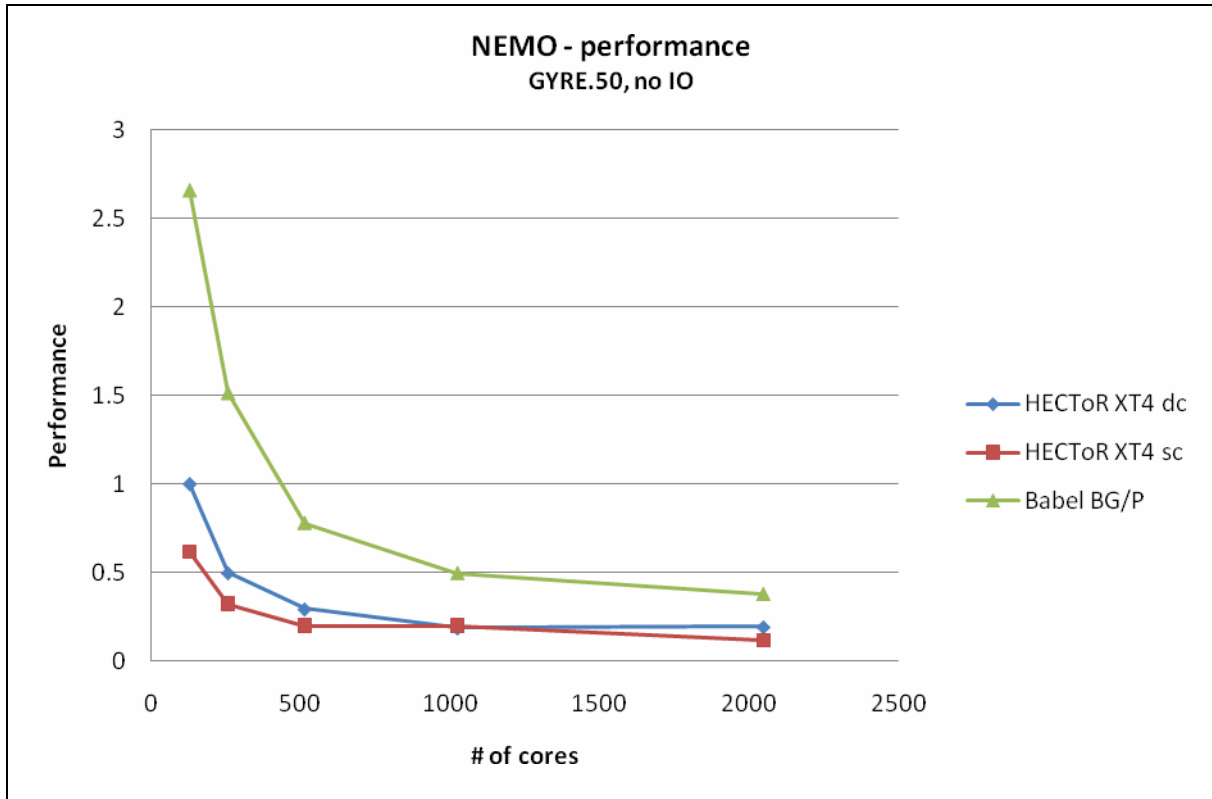


Figure 15 – Performance of NEMO on HECToR and Babel (from 128 cores).

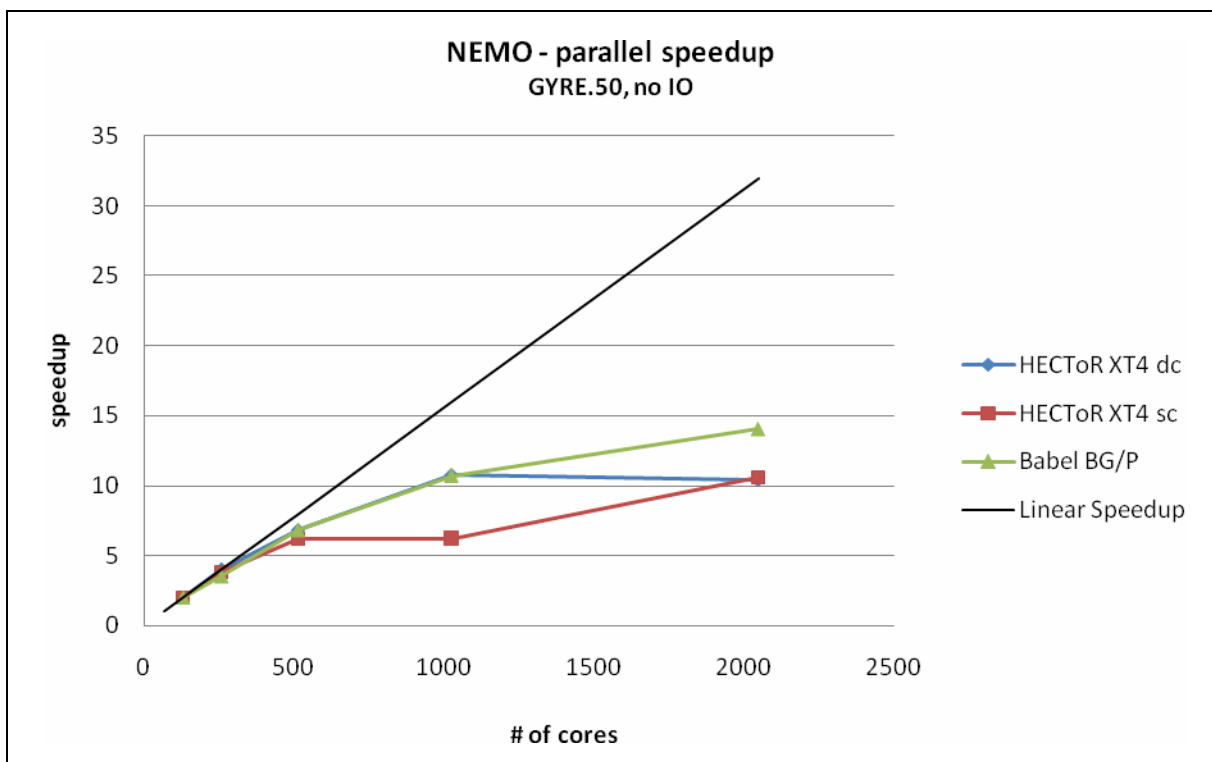


Figure 16 – Parallel speedup for NEMO (relative to 128 cores).

3.2.2.10 RAMSES

The performance of RAMSES on Babel is not as good as could have been expected. The performance of RAMSES with the Sedov3D dataset (which has perfect load balancing but is very CPU intensive) is shown in Figure 17. The application was run in VN mode with TXYZ task mapping. The MESH network was used up to 1024 cores, the TORUS network showed better performance for 2048 and 4096 cores. Setting the `-qarch=450d` compiler option resulted in improved runtimes.

The performance ratio between Babel and HECToR DC is 1.5, which is much better than expected with a ratio of the theoretical peak performance per core at 1.7. Also expected for the Sedov3D dataset, the parallel scaling is excellent up to 4096 cores (see Figure 18).

For the AMR data set, which has bad load balancing and is much more communication intensive than Sedov3D, the performance ratio varies from 2.5 (on 64 cores) to 1.7 (on 1024 cores) and even 0.4 (on 2048 cores). On 2048 cores Babel is two times faster than HECToR. The AMR test is known to have a very bad scaling: on HECToR, we get an improvement until 1024 cores, after this timing explodes. On Babel however the increasing in runtimes only occurs from 4096 cores: timings are stable between 1024 and 2048 cores, there is no improvement, but no degradation either.

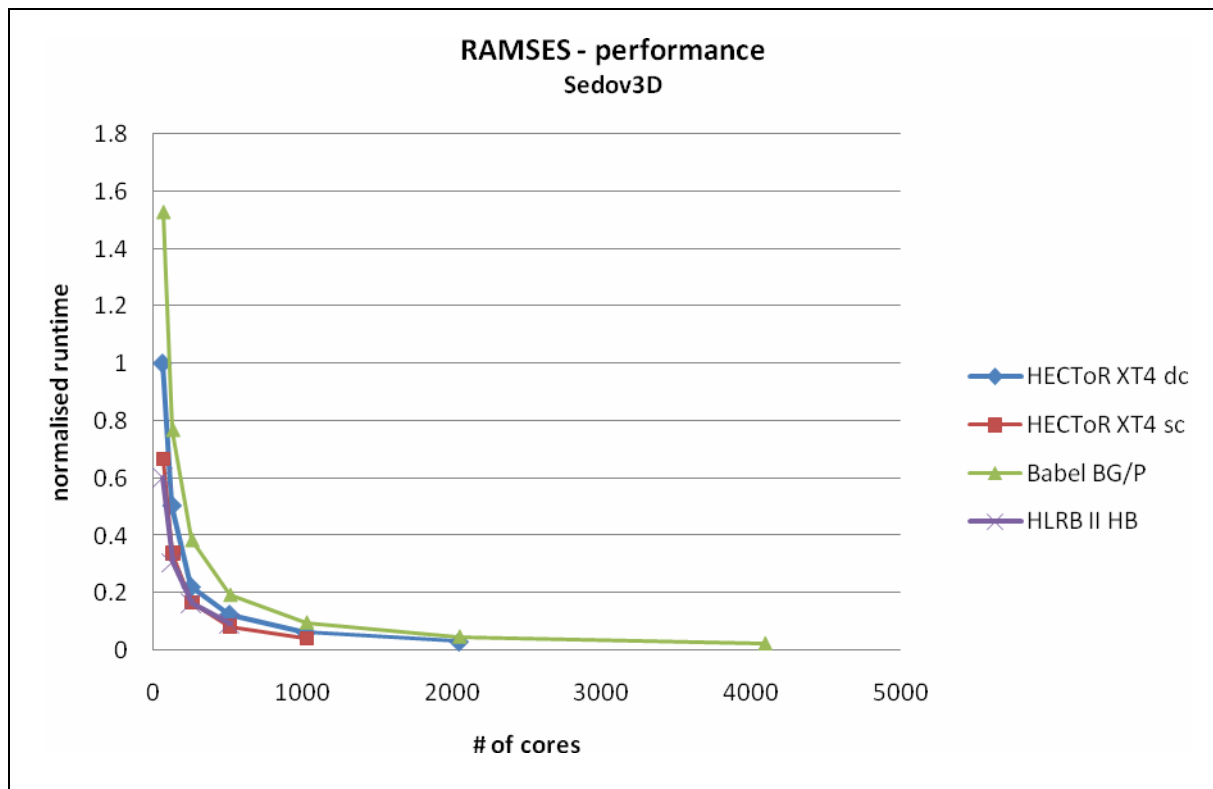


Figure 17 – Performance of RAMSES on HECToR, HLRB II and Babel (from 64 cores).

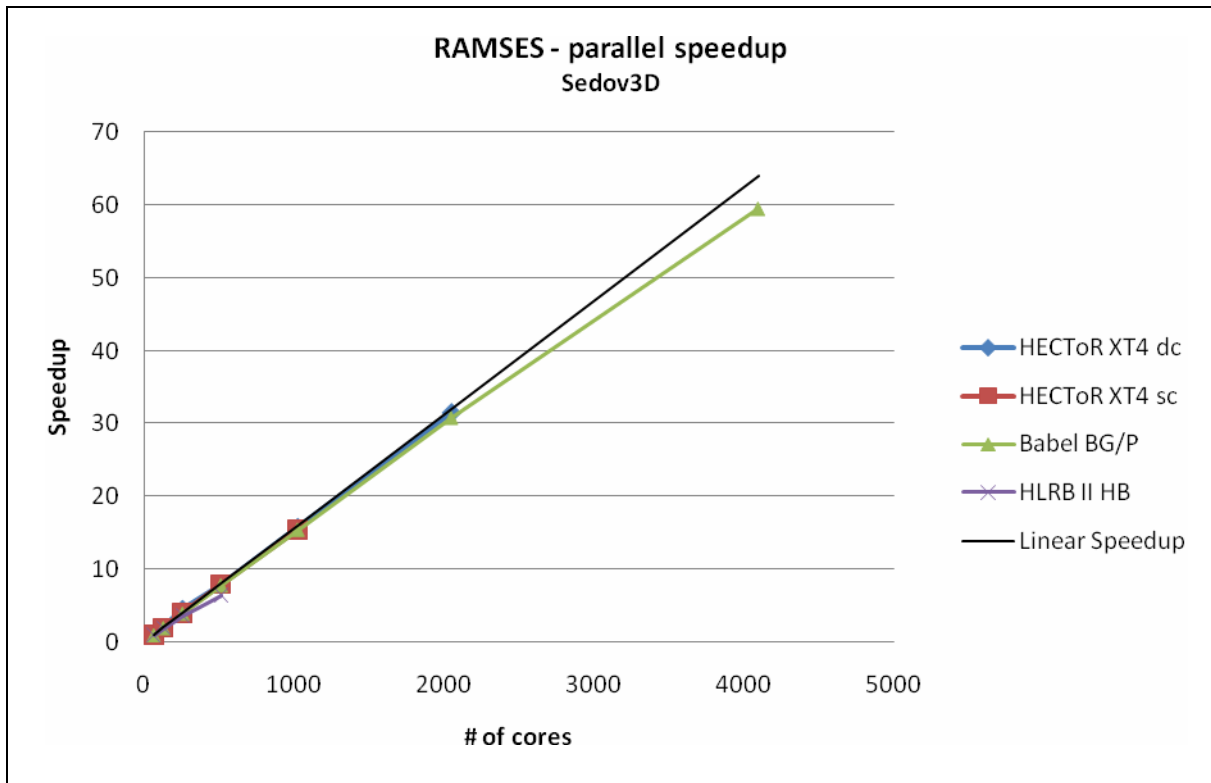


Figure 18 – Parallel speedup for RAMSES (relative to 64 cores).

3.2.2.11 GADGET

Similar to most applications in the Benchmark Suite, there were no problems porting GADGET to the BG/P architecture.

All performance runs used VN mode as well as the MESH network. Tests showed that GADGET's performance does not benefit for 2 floating point pipes and thus the code was build without the Double Hummer option.

Figure 19 shows the performance of GADGET, compared to HECToR and HLRB II. GADGET is more than 5 times slower on Babel than on HECToR, which shows much poorer performance than could be expected from this application. Initial tests show that computation, rather than communication, is the bottleneck for GADGET. For instance, the treewalk function takes twice the overall percentage of time to compute on Babel than on HECToR. The parts of the code that cause the drop in performance are integer operations intensive – further investigation comparing the integer operations rate on both HECToR and Babel is required in order to confirm whether this could be the cause of the disappointing performance.

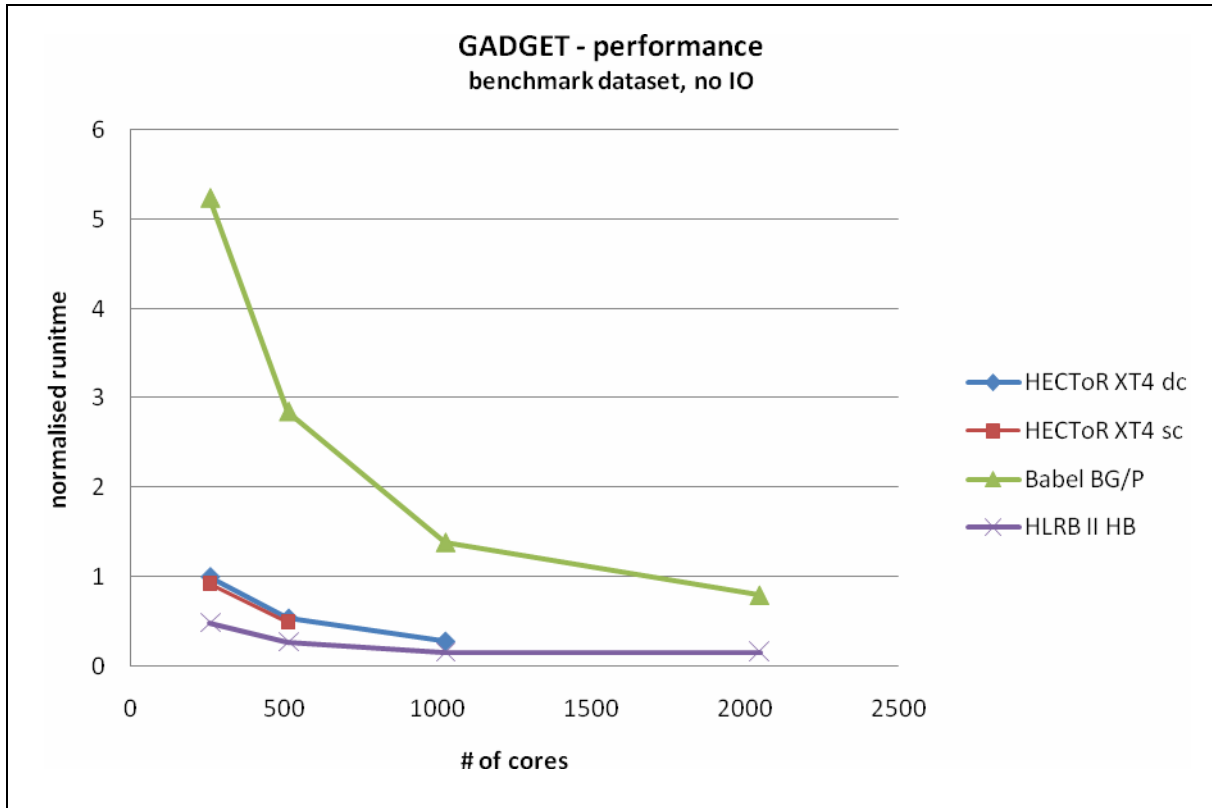


Figure 19 – Performance of GADGET on HECToR, HLRB II and Babel (from 256 cores).

The parallel speedup of GADGET is very good up to 2048 cores and the code maintains more than 80% efficiency (see Figure 20). Regardless of its relative performance, GADGET shows the best scalability on Babel compared to HECToR and HLRB II.

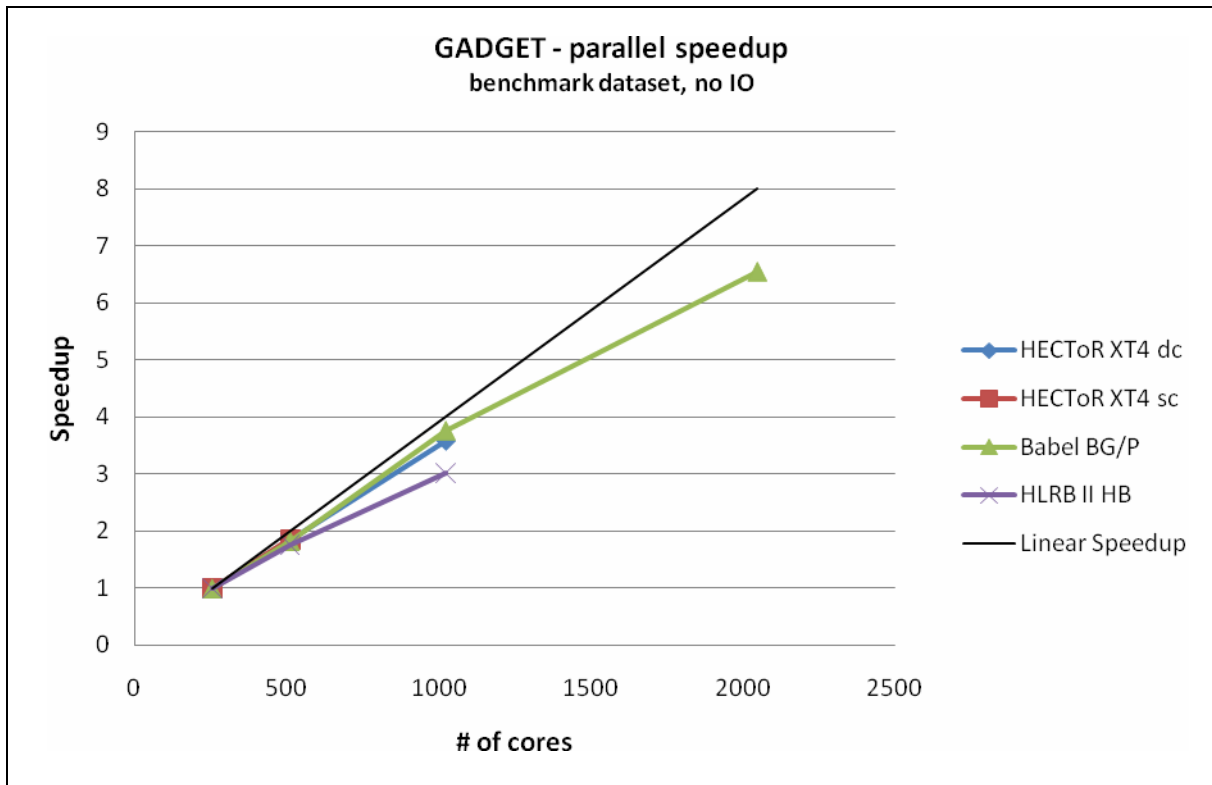


Figure 20 – Parallel speedup of GADGET (relative to 256 cores).

3.2.2.12 Fenfloss

The final code, Fenfloss, was again ported to Babel with no difficulties. However the runtimes that we achieved were very poor. Runs on 512 cores were slower than runs on 256 cores:

```
256 Procs: 91 s
512 Procs: 115 s
1024 Procs: 202 s
```

At 256 cores, the runtime ratio compared to HECToR DC is 3.1 (i.e. 91s / 29s), which is an expected result. However the performance on Babel rapidly decreases. The exact reason for this is currently not clear, however there is a suspicion that this slowdown is caused by IO. The application has every process write a binary file to disk, which seems to result in a bottleneck on the IO system. Further tests need to be conducted in order to determine whether the IO is indeed the cause for the drop in performance. These tests include changing the output directory (from \$WORKDIR to \$TMPDIR) and running in SMP mode (twice the number of IO nodes would be allocated to each run).

3.2.3 Summary of Results

The results for the DEISA Benchmark Suite applications on the BG/P architecture reflect a convincing performance profile for this platform. Most of the codes perform better (or at least as good) as expected with comparison to HECToR, and nearly all the applications show increased scalability. The fact that the BG/P architecture shows excellent parallel speedup is perhaps not surprising, given the specialised networks optimised for communication and code execution.

A small number of codes show disappointing performance, and in most cases it is not directly clear what the causes are - further investigation is required. Memory however is an expected bottleneck on BG/P and memory hungry applications struggle with the BG/P architecture.

4 DECI CPU conversion factors

Projects that are accepted under the DEISA DECI programme are given allocations of standardised computing time. The time given to the projects is converted into budgets of CPU/hours using CPU conversion factors. Currently these factors are based on various application benchmark results (including external results), sometimes only on few and different codes. It is planned to use the DEISA benchmark Suite in future for the determination of these conversion factors, also for new architectures.

The proposal is to set up a committee of experts, one from each DEISA partner, to agree on a procedure for the determination of the platform specific DECI CPU conversion factors for the DEISA infrastructure. This committee should meet on a six monthly basis and review the conversion factors. The performance results from the DEISA Benchmark Suite should be taken into account when reviewing the factors. The set of applications, which cover a broad range of scientific areas and algorithms, are ideally suited to be used to gain a balanced and realistic view of the performance profile of the DEISA infrastructure. The runtimes that are achieved on the different platforms reflect, for each of the applications, the performance of a combination of factors such as CPU clock speed, interconnect and memory bandwidth and latency.

The following paragraphs outline the proposal for an easy-to-maintain method of using the Benchmark Suite to summarise the performance of the DEISA platforms.

4.1 Conversion factors based on Benchmark Suite

We propose to use Cray's XT4 HECToR as the new reference system³ for the determination of future conversion factors, according to the method outlined below.

For every application in the Benchmark Suite, a reference dataset and a minimum core count for each performance run is defined. The "minimum core count" represents the highest number of cores on which the application achieves >50% efficiency across the platforms. Table 2 shows our proposed set of reference datasets and minimum core counts.

Code	Dataset	Min # cores
NAMD	4f2hc	256
IQCS	32 Qubits	256
CPMD	H2O_256mol	256
QuantumESPRESSO	PSIWAT	256
GENE	strong_512	512
PEPC	Sphere 5M	512
BQCD	48*48*48*96, 50 iterations	512
SU3_AHiggs	256^3 lattice, 100 iterations	512
NEMO	GYRE.50, no IO	512
RAMSES	Sedov3D	512
Fenfloss	Cavity224	512
GADGET	benchmark dataset	512

Table 2 – List of reference datasets and minimum core counts to be used across all platforms.

³ Please note that HECToR will be upgraded during 2009. The Benchmark Suite will need to be rerun on the upgraded system for a new set of reference results.

The actual timings used in calculating the conversion factors are taken from those runs with the lowest “cost” (i.e. cost = # of cores * time). The number of cores used to calculate this cost needs to be equal to or larger than the defined minimum core count.

Using the runtimes and core counts listed in Table 2, the code conversion factors for each application can be computed thus:

$$\text{factor}(\text{code}, \text{new_plat}) = \frac{\text{ref_cores} * \text{runtime}(\text{ref_dataset}, \text{ref_cores}, \text{ref_plat})}{\text{cores} * \text{runtime}(\text{ref_dataset}, \text{cores}, \text{new_plat})}$$

The final conversion factor for a platform can then be calculated, for example, as the arithmetic mean of the code factors:

$$\text{Factor}(\text{plat}) = (1/\text{codes}) [\text{factor}(\text{NAMD}, \text{plat}) + \text{factor}(\text{IQCS}, \text{plat}) + \dots]$$

Alternatively, the platform conversion factor could be calculated using the geometric mean. It is also not necessary to use all the application factors in the calculation of the platform factor. The final decision as to how the proposed process will be used in practice (including the final procedure for the platform conversion factor) will be taken by the committee of experts.

4.2 Example calculation

Using the proposed method below is a sample conversion factor calculation for HLRB II, using HECToR as the reference platform. Table 3 lists the specific runtimes and core counts used to calculate the code performance factors.

<i>Code</i>	<i>SGI Altix HLRB II</i>		<i>Cray XT4 HECToR DC</i>	
	<i>Runtime</i>	<i># cores</i>	<i>Runtime</i>	<i># cores</i>
NAMD	3036.08	256	6710.41	256
IQCS	19.31	256	29.26	256
CPMD	10.99	256	13.616	256
QuantumESPRESSO	818.19	256	835.15	256
GENE	6.98	512	7.46	512
PEPC	102.02	512	44.5	512
BQCD	232.27	512	367.93	512
SU3_AHiggs	26.70	2048	24.4	2048
NEMO	414.00	512	609	512
RAMSES	41.00	512	14	2048
Fenfloss	39.82	512	14.74	512
GADGET	29.29	512	14.535	512

Table 3 – List of runtimes and core counts per application on both the reference and the target platforms.

For example, using the values in Table 3, the performance factor for the RAMSES application on HRLB II is calculated thus:

$$\begin{aligned} \text{factor}(\text{RAMSES}, \text{HLRB II}) &= \frac{2048 * \text{runtime}(\text{Sedov3D}, 2048, \text{HECToR DC})}{512 * \text{runtime}(\text{Sedov3D}, 512, \text{HLRB II})} \\ &= \frac{2048 * 14.00}{512 * 41.00} \\ &= 1.365 \end{aligned}$$

The performance factors are then calculated for every application, and the resulting platform conversion factor results as the mean of the application factors, in our case 1.14.

5 Website

5.1 Login procedure

The maintenance of the Benchmark Suite includes the maintenance of the website that is used to distribute the applications and provide information and sample results for users.

In order to be able to gather information on the user base, it was decided to implement a simple registration procedure, prior to reaching the download area of the website⁴. Figure 21 shows a screenshot of the registration page. Users are asked for their name and institute, and can add comments if they wish. All the information gathered in this way is used for statistical purposes only, allowing us to see who uses the Benchmark Suite and what applications they download.

Access statistics will be provided once the new registration mechanism has been in place sufficiently long to be able to provide meaningful data.

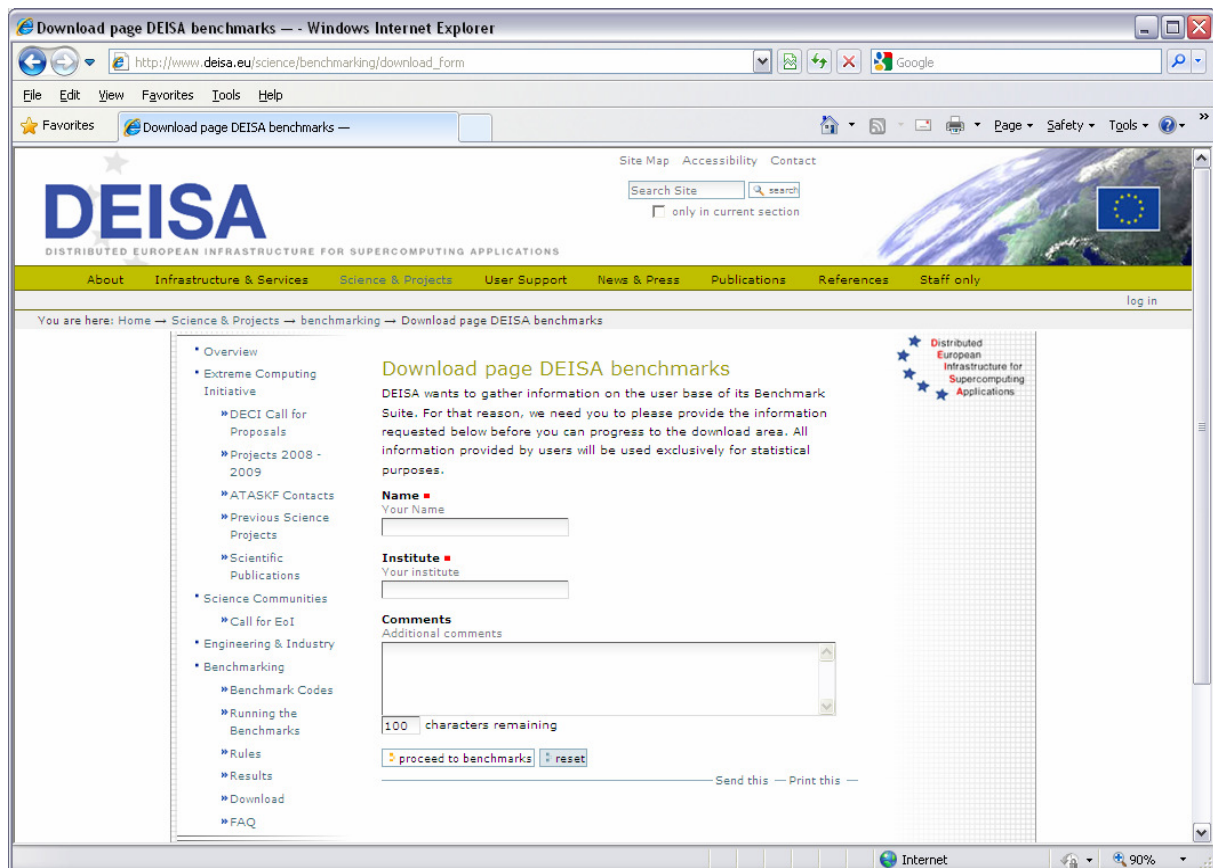


Figure 21 – New login procedure to access the download area for the DEISA Benchmark Suite.

⁴ Many thanks go to Christof Hanke from CSC for implementing the registration procedure.

6 Future Work

In the coming twelve months, various tasks will need to be completed:

- At the end of year 1, it is necessary to review the content of the Benchmark Suite. This involves both reviewing the applications in the Suite as well as the versions of those applications. During the porting exercise to the BG/P, a few applications were already highlighted as potentially benefiting from being updated to more recent versions, be it because of bugs or performance issues.
- Once this review has been completed, it will be necessary to return to the Power6 at RZG and complete the porting work and performance analyses.
- Following this, the Benchmark Suite will need to be rerun on platforms that were addressed previously, but have since undergone upgrades. A complete performance picture will be needed for the contribution to the DECI CPU conversion factors.
- This work will be complemented by the maintenance of the website.
- Identification of highly scalable datasets suitable for benchmarking upcoming supercomputer architectures in a challenging way

7 Conclusion

A large amount of work was completed over the past six months, including porting to a new architecture and performance analyses.

The fact that the DEISA Benchmark Suite has started to be picked up by external users shows that the work is valued by scientists. The website tries to provide users with all the help and support that might be required in order to become familiar with the Suite. The new registration step on the other hand allows us to get a picture as to who constitutes our user based.

The performance results achieved with the Benchmark Suite will also contribute to the calculation of the DECI CPU conversion factors following the procedure outlined in this deliverable and based on the expert opinions of a committee, set up especially for this purpose.

With the completion of the porting work on the BG/P system, the DEISA Benchmark Suite has now been ported to all major architectures in the DEISA infrastructure. The Suite has not yet been ported to all the DEISA machines, which all have slight differences in hardware and set-up, however a major step towards this goal has been achieved.