Performance Analysis of HPC Applications on Low-Power Embedded Platforms

Luka Stanisic, Brice Videau, Johan Cronsioe, Augustin Degomme, Vania Marangozova-Martin, Arnaud Legrand, Jean-François Mehaut

Performance Analysis of HPC Applications on Low-Power Embedded Platforms

(Invited Paper)

Luka Stanisic and Brice Videau and Johan Cronsioe and Augustin Degomme
and Vania Marangozova-Martin and Arnaud Legrand and Jean-François Méhaut
Université de Grenoble, UJF, CNRS, CEA
Laboratoire d'Informatique de Grenoble
France
Email: name.surname@imag.fr

***Abstract*—This paper presents performance evaluation and analysis of well-known HPC applications and benchmarks running on low-power embedded platforms. The performance to power consumption ratios are compared to classical x86 systems. Scalability studies have been conducted on the Mont-Blanc Tibidabo cluster. We have also investigated optimization opportunities and pitfalls induced by the use of these new platforms, and proposed optimization strategies based on auto-tuning.**

I. Introduction

Building supercomputer with peak performance in the exaflops range cannot be achieved with nowadays technology. Indeed, such a machine would consume much more than the 20 MW budget a supercomputer is supposed not to exceed. Nowadays the head of the Top500 [1] (the 500 most powerful computer systems in the world) is ranked third of the Green500 [2] (the 500 most powerful computer systems but ranked by efficiency). It reaches an efficiency of about 2 GFLOPS per Watt. Building an exaflopic computer under the 20MW barrier would require an efficiency of 50 GFLOPS per watt.

The trends of the performance development are presented in Figure 1. In order to break the exaflops barrier by the projected year of 2018 the efficiency of supercomputers need to be increased by a factor of 25 by this time.

One potential solution to achieve such an efficiency is using components produced by the embedded industry where power efficiency is paramount. The European Mont-Blanc project [3][4] was created to evaluate the use of such components in an HPC environment.

The remainder of the paper is organized as follow. Section II presents the goals of Mont-Blanc project, the HPC applications that were selected to evaluate the performance of the target platform and the design of one prototype used in the project. Section III presents preliminary efficiency results on a single node while Section IV focuses on scalability studies on the prototype. Optimization studies are presented in Section V and show that careful investigation of performance as well as auto-tuning will be required when conducting such investigation. Perspectives and conclusions are presented in Section VI and VII respectively.

978-3-9815370-0-0/DATE13/©2013 EDAA

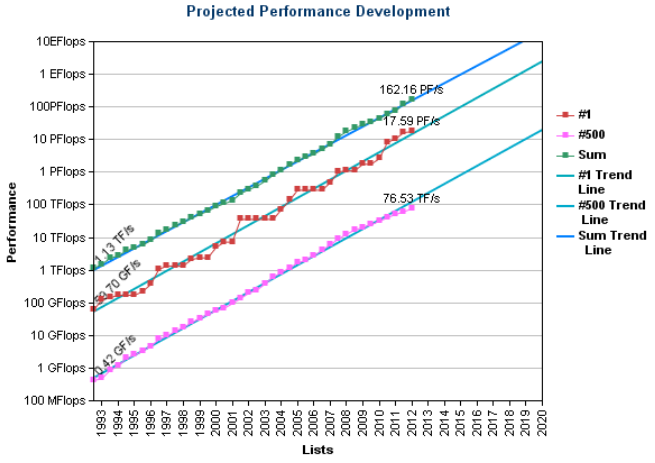


Figure 1: Exponential growth of supercomputing power as recorded by the TOP500.

II. The Mont-Blanc Project

The main objectives of the Mont-Blanc European project are:

- Develop prototypes of HPC clusters using low power commercially available embedded technology.
- Design the next generation in HPC systems based on embedded technologies and experiments on the prototypes.
- Develop a portfolio of existing applications to test these systems and optimize their efficiency, using BSC's OmpSs programming model [5].

A. Beyond LINPACK and the Top500: Selected Applications for the Mont-Blanc Project

Top500 rankings are obtained using the LINPACK benchmark, which makes extensive use of dense linear algebra. But many other paradigms of programming are used in HPC nowadays. Thus the Mont-Blanc project actively sought real applications to evaluate the performance of the prototypes and the feasibility of the approach.

Eleven applications were selected as candidates for porting and optimization. They are presented in Table I. They are

Table I: Mont-Blanc Selected HPC Applications.

Code	Scientific Domain	Institution
YALES2	Combustion	CNRS/CORIA
EUTERPE	Fusion	BSC
SPECFEM3D	Wave Propagation	CNRS
MP2C	Multi-particle Collision	JSC
BigDFT	Electronic Structure	CEA
Quantum Expresso	Electronic Structure	CINECA
PEPC	Coulomb & Gravitational Forces	JSC
SMMP	Protein Folding	JSC
PorFASI	Protein Folding	JSC
COSMO	Weather Forecast	CINECA
BQCD	Particle Physics	LRZ

state of the art HPC codes currently running on national HPC facilities or on European supercomputer such as the ones deployed by the PRACE European Project.

In this paper we focus especially on two codes SPECFEM3D and BigDFT.

a) BigDFT: [6] The goal of BigDFT is to develop a novel approach for electronic structure simulation based on the Daubechies wavelets formalism [7][8]. The code is HPC oriented, i.e., it uses MPI, OpenMP and GPU technologies. So far, BigDFT is the sole electronic structure code based on systematic basis sets which can use hybrid supercomputers.

b) SPECFEM3D: [9] simulates seismic wave propagation on local to regional scales, and in its GLOBE version on global scale. SPECFEM3D uses continuous Galerkin spectral-element method. It is HPC oriented and uses MPI plus GPU technologies. The code has shown excellent scalability achieving 0.7 PFLOPS on 149 784 cores on the Jaguar cluster and using 1152 GPUs on $TSUBAME2.0$. This scalability is achieved by using careful load-balancing and point to point communications.

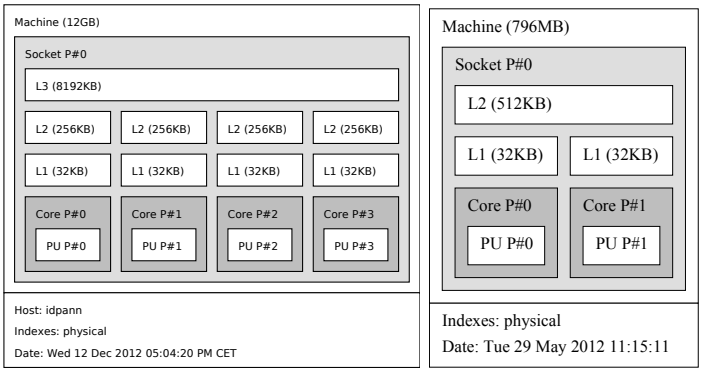
B. Beyond Classical Petascale Platforms: ARM and Ethernet Based HPC

The most well-established, low-power, embedded, off-the-shelf architecture available is the ARM one. It also has a well developed ecosystem. The first Mont-Blanc prototype is expected to be available during the year 2014. It will be using Samsung Exynos 5 Dual Cortex A15 processors with an embedded Mali T604 GPU and will be using Ethernet for communication.

In order to start evaluating the applications before 2014, a small cluster of ARM system on chip was built. It is named Tibidabo and is hosted at the Barcelona Supercomputing Center. Tibidabo [10] is an experimental HPC cluster built using NVIDIA Tegra2 chips, each a dual-core ARM Cortex-A9 processor. The PCI Express support of Tegra2 is used to connect a 1Gb Ethernet NIC, and the board are interconnected hierarchically using 48-port 1 GbE switches. Scalability studies in the paper are realized using this platform.

III. Efficiency results for a single node

For these experiments, we will use a Snowball embedded computer. These boards are developed by the French company Calao Systems [11] and use a dual-core ARM SoC with an integrated Mali GPU, designed by ST-Ericsson. We used these boards because their design is close to what we envision for the Mont-Blanc prototype: a low-power ARM SoC with integrated GPU.



(a) Xeon 5550 topology (b) A9500 topology

Figure 2: Memory characteristics of the platform used in our experiments.

A. Hardware and Software Setup

The Snowball board is a full embedded computer with state of the art features, and a powerful CPU developed by ST-Ericsson, the A9500. This SoC is a dual-core 1GHz ARM with an integrated Mali 400 GPU, and a Neon floating point unit (single precision only).

The main hardware features of the board we are using are :

- ARM Dual Cortex A9 @ 1GHz
- 8GByte e-MMC
- 1GByte LP-DDR2
- 1x Micro-SD
- 1x HDMI Full HD
- 1x Ethernet 100Mbits
- 1x USB OTG HS (480Mbits)

The boards can be powered by a battery, a sector adapter, or by USB only, so their power consumption is less than 2.5W (maximum value available from USB). The snowball boards can run multiple Linux-based systems, as Android or Meego, but our focus will be on the Ubuntu-based distribution Linaro[1]. All benchmarks and applications were built using *gcc-4.6.2*.

The Snowball board is benchmarked against an Intel Xeon X5550 running Debian 64 bits. This CPU is a quad core 2.6GHz Nehalem CPU, which has a TDP of 95Watts and is used in several top500 HPC systems around the world. The Xeon system runs Debian experimental and benchmarks were built using *gcc-4.6.2*. Topologies of the Xeon processor and the STE A9500 is exposed in Figures 2a and 2b.

B. Software Setup

In order to compare the two systems, we used SPECFEM3D and BigDFT on small instances. We also used 3 different

[1]http://www.linaro.org

Table II: Comparison between an Intel Xeon 5550 and ST-Ericsson A9500.

Benchmark	Snowball	Xeon	Ratio	Energy Ratio
LINPACK (MFLOPS)	620	24000	38.7	1.0
CoreMark (ops/s)	5877	41950	7.1	0.2
StockFish (ops/s)	224,113	4,521,733	20.2	0.5
SPECFEM3D (s)	186.8	23.5	7.9	0.2
BigDFT (s)	420.4	18.1	23.2	0.6

benchmarks:

- LINPACK: the standard HPC benchmark,
- CoreMark: a benchmark aimed at becoming the industry standard for embedded platforms,
- and StockFish: an open-source chess engine with benchmarking capabilities.

These programs were built using *gcc-4.6.2* with the following options: *-O3* for the Xeon platform and *-03 -march=armv7-a -mtune=cortex-a9 -mfpu=neon -mfloat-abi=softfp* for the ARM in order to use the neon FPU unit. It has to be noted that BigDFT, SPECFEM3D and LINPACK have been optimized for Intel architecture while the code remains unchanged when built on the ARM platform. BigDFT and SPECFEM3D were built using *gfortran-4.6.2*.

C. Experimental Results

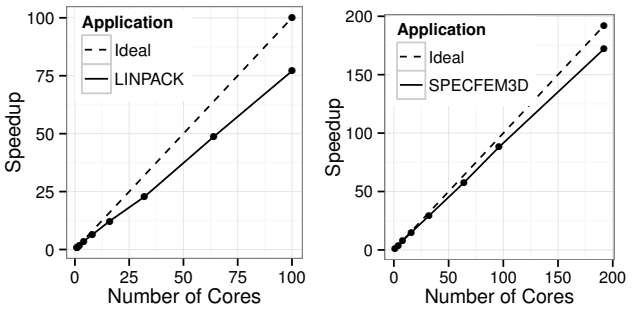
Experimental results are presented in Table II. They compare the Snowball using 2 cores to the Xeon platform using 4 cores but with hyperthreading disabled. The results assume a full 2.5W power consumption for the Snowball board, while only 95W of power (the TDP of the Xeon) are accounted for the Intel platform. This is a very conservative estimation, highly unfavorable for the ARM platform but we will see that even with this handicap the ARM boards are interesting.

Indeed, running the LINPACK benchmarks costs the same energy (in our rough model) on the Xeon as on the Snowball. But for CoreMark and SPECFEM3D the energy required is 5 times lower. For StockFish and BigDFT only half the energy is consumed by the ARM platform.

IV. Scalability Results

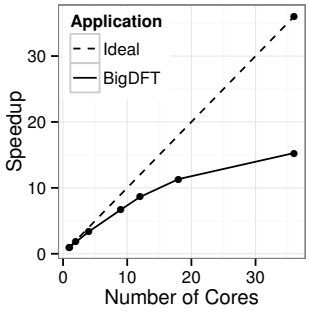
In order to be viable the approach needs applications to scale. Indeed, to achieve the same level of performance, in term of time to solution, the number of low-power nodes required can rapidly grow if the applications does not scale. We used Tibidabo to evaluate the scalability of our applications, as well as the scalability of the LINPACK benchmark.

The results are presented on Figure 3. Scalability of the LINPACK benchmark is acceptable, and close to 80% efficiency for 100 nodes. It has to be noted that the speedup curve is linear after 32 nodes, which indicates that scaling could continue for more nodes. The scaling of SPECFEM3D is excellent, showing strong scaling with an efficiency of 90% when comparing with a 4 core execution of the same instance. This detail is important because memory bus saturation often make HPC applications lose performance when using 2 cores on a node instead of 1. Unfortunately, since these are strong



(a) Speedup of the LINPACK benchmark on Tibidabo.

(b) Speedup of SPECFEM3D on Tibidabo.



(c) Speedup of BigDFT on Tibidabo.

Figure 3: Strong scaling of different application and benchmarks on Tibidabo. SPECFEM3D scaling is versus a 4 core run as the use-case cannot be run on less than 2 nodes.

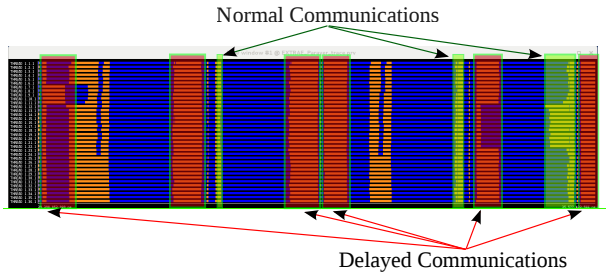


Figure 4: Profiling of BigDFT on tibidabo using 36 cores. Collective communications are sometimes delayed.

scaling experiments, one node does not have enough memory to load this instance, which hence requires at least two nodes. Nonetheless its scalability is excellent. BigDFT's case is more troubling as its efficiency drops rapidly.

In order to identify BigDFT's problem on Tibidabo, the execution of the program had to be carefully investigated. BigDFT was profiled using [12] an automatic code instrumentation library and Paraver [13], a visualization tool dedicated to parallel code analysis. A small portion of a 36 core execution is presented on Figure 4. BigDFT mostly uses all to all communication patterns. These communications operations are presented in orange on the communication diagram. The communications that are interesting in our case are the *all_to_all_v*, and these are circled in green in the figure. These communications should be small, as the one pointed as normal

communication. Unfortunately, when using 36 cores most of these collective communications are longer and delayed. In some cases all the nodes are delayed while in other, only part of them suffers from this problem. The Ethernet switches used in Tibidabo was identified as the origin of these bad performances. Since only collective communications really incur important congestion, SPECFEM3D doesn't suffer from the problem and LINPACK is only affected to a lesser extent. This problem is to be fixed by upgrading the Ethernet switches used on Tibidabo.

No power measurement was done so far at large scale, but experiments are ongoing. Nonetheless, with current hardware, the node power efficiency is likely to be counterbalanced by the network inefficiency. For the final Mont-Blanc prototype high speed Ethernet network with power saving capabilities has been selected and will hopefully correct most of these problems [**?**].

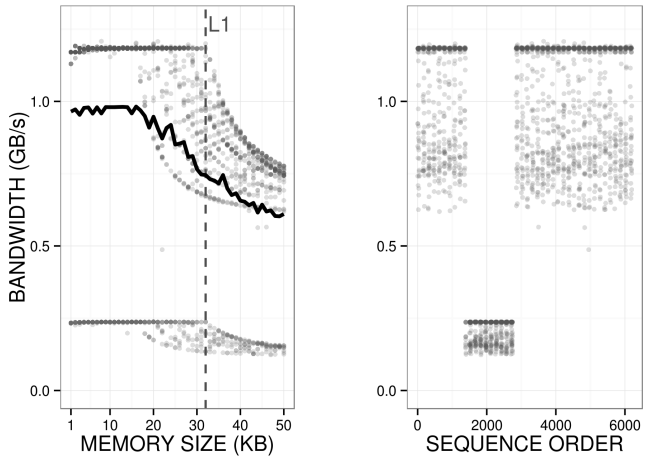
V. Toward Practical Optimization of HPC applications on ARM

A. Importance of Environment Parameters and Code Optimizations

In order to carefully investigate performance, we started with a very simple memory intensive kernel based on [14]. Essentially, this benchmark measures the time needed to access data by looping over an array of a fixed *size* using a fixed *stride*. Such parameters provide a crude estimation how temporal and spatial locality of the code impact performance on a given machine. Effective memory bandwidth is evaluated as the total number of accesses divided by the time it took to execute all of them. Since ARM processor have never been used for HPC before and have a very different memory hierarchy from those typically used, we suspected different behaviors comparing to the well know x86 architectures.

1) Influence of Physical Page Allocation: In our first experiments, we observed a very surprising behavior in term of experiment reproducibility. Despite very little performance variability inside a set of measurements on Snowball, from one run to another we were getting very different global behavior. Yet, the environment setup and input parameters were completely unchanged. The origin of this surprising phenomenon comes from the way the OS on ARM allocates physical memory pages. In some cases, nonconsecutive pages in physical memory for array size around 32KB (the size of L1 cache) are allocated, which causes much more cache misses, hence a dramatic drop of overall performance. Furthermore, during one experiment run, OS was likely to reuse the same pages, as we did malloc/free repeatedly for each array. Hence, array started from the same physical memory location for each set of measurements, which explains why there is almost no noise inside a run. This is extremely important as the performance of future application can severely vary depending on the pages chosen by OS. This also means such benchmarks and auto-tuning methods need to be thoroughly randomized to avoid experimental bias.

2) Unexpected Behavior With Real-Time Scheduling: Another important factor in overall performance is OS scheduler. Using real-time scheduler is often a good way to obtain the most performance out of an application on standard systems [15]. Surprisingly, this approach lead to unexpectedly poor and unstable performances on our ARM system. On Figure 5a one can observe 2 modes of execution. The first mode, which delivers the higher bandwidth values, is similar to the results we have obtained with other scheduling priorities, hence this scheduling mode does not bring any performance improvement. Furthermore, the second mode delivers degraded bandwidth values that are almost 5 times lower. One can also clearly see from Figure 5b that all degraded measures occurred consecutively, which is likely caused by plainly wrong OS scheduling decisions during that period of time.



(a) Bandwidth as a function of array size. Performance decrease when size exceeds the L1 cache. 2 modes of execution can be observed. Solid black line for average values.

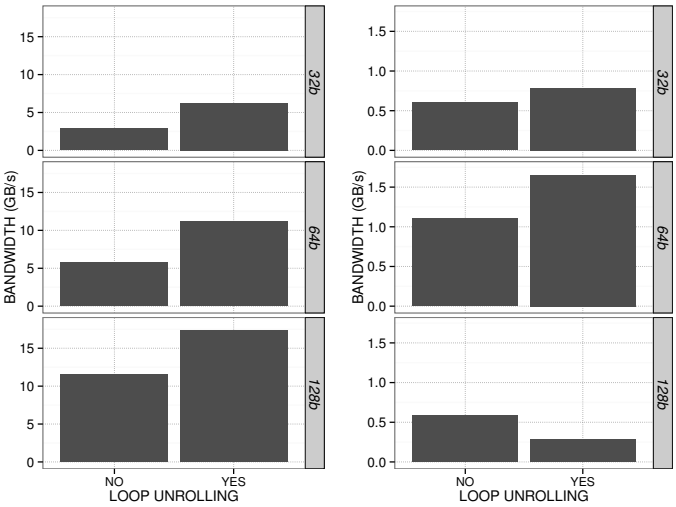
(b) Same data represented with a sequence order plot. Measurements in degraded mode are actually consecutive.

Figure 5: Impact of real-time priority on ARM Snowball's effective bandwidth (using a fixed stride=1 and varying array size). 42 randomized repetitions for each array size 1KB-50KB.

3) Influence of Code Optimizations: Among different optimization techniques, changing element sizes to vectorize and loop unrolling to improve pipelining opportunities are generally very effective. In Figure 6, the left column depicts the results measured with the initial kernel (without loop unrolling) while right column depicts the results when manually unrolling loops 8 times. Rows illustrate the influence of element sizes of the array.

As it can be observed, increasing element size from 32 bits to 64 bits practically doubles the bandwidths on both architectures. Loop unrolling also has a very positive effect and allows to go toward the true limits of the processor. As can bee seen on Figure 6b, the best performance for Nehalem are obtained when vectorizing with 128 bits elements and unrolling loops.

Surprisingly, on ARM (Figure 6b), vectorizing with 128 is similar to using 32 bit elements and loop unrolling may even dramatically degrade performance. The best configuration on ARM is obtained when using 64 bits and loop unrolling but this kind of choice is likely to depend on computation kernel.



(a) Xeon 5500/Nehalem: unrolling loops and vectorizing both constantly improve performance.

(b) Snowball/ARM A9500: both vectorizing and loop unrolling may be detrimental.

Figure 6: Influence of code optimizations on ARM Snowball (effective bandwidth for 50KB array with stride 1).

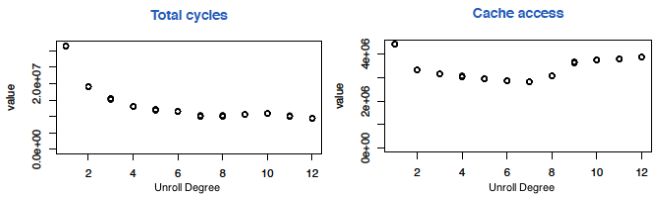
Although a simplistic kernel was used, results intensively varied depending on environment setup and optimizations in drastically different way from what is generally obtained on more classical HPC architecture. This means that a particular attention should be given to auto-tuning techniques, which may have to explore more systematically parameter space, rather than being guided by developers' intuition.

B. Auto-Tuning: Loop unrolling Use Case

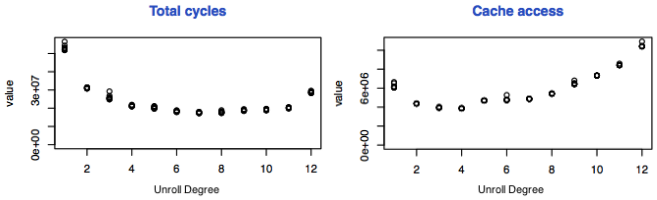
HPC code that do not rely on external libraries are responsible for their own optimization. Usually, when a procedure is written it is empirically optimized for a specific target platform. When using similar architectures these optimizations may still exhibit a decent level of performance, but they should be seriously revisited when changing for a radically different architecture.

Nonetheless, optimizing a whole HPC application is a very costly and error prone process as beneficial optimizations generally requires to deeply transform the source code. One way to automate this optimization process is to harness the developer's knowledge and to provide him tools to express optimization variations. These variations are then benchmarked and the most suitable for the platform selected.

We followed this approach with the BigDFT core function – the magicfilter –, which performs the electronic potential computation via a three-dimensional convolution. This convolution can be decomposed as three successive applications



(a) Intel Nehalem



(b) Tegra 2

Figure 7: Number of cycles and cache accesses needed to apply the magicfilter depending on the unroll degree and the architecture

of a basic operation, which consists of nested loops. Such loops can be unrolled and, depending on the unrolling degree, performance may be greatly improved.

The code with and the optimization were described in an dedicated tool we designed. The tool generated the magic filter with unrolling varying from 1 (no unrolling) to 12. These convolutions were then automatically benchmarked using PAPI [16] counters.

Benchmarks results for two counters (number of cycles needed and number of access in the cache) and for two platforms are presented in Figure 7. The shapes of the curves are somehow similar but differ drastically in scale. They are all roughly convex in the unroll degree except maybe for some sort of small staircase in the number of cache accesses (unroll=9 for Nehalem compared to unroll with unroll=5 for Tegra2). Interestingly, on Tegra2, the total number of cycles significantly grows when unrolling too much (unroll=12). This performance degradation can actually be anticipated by looking at the number of cache accesses that start growing very quickly (starting at unroll=4) and becomes at some point detrimental. In these experiments, the main difference in term of behavior between Nehalem and Tegra2 is thus in term of scale. As a consequence, the sweet spot area where loop unrolling is beneficial and does not incur a too high number of cache accesses is smaller on Tegra2 (the [4:7] range) than on Nehalem (the [4:12] range). On other kernels, the sweet spot range is even smaller, which means that on such machines, tuning will require an even more systematic and careful attention than on classical HPC architectures. Actually, such tuning process will have to be fully automated to ensure the portability of the performance of the application.

VI. Perspectives

Although the experiments on the Snowball and Mont-Blanc Tibidabo cluster are only preliminary, they are very promising and open several perspectives.

A. Toward Hybrid Embedded Platforms

The use of General Purpose Graphical Processing Units (GPGPU) is a growing trend in the HPC community. Indeed these GPGPU offer impressive peak computation performance as well as memory bandwidth. Low-power versions of these accelerators exist and have a very attractive performance per Watt ratio. That is why Tibidabo is being extended with Tegra 3 with an adjoined GPU suitable for general purpose programming. This will allow codes that can use single precision to exploit a low-power hybrid computer. SPECFEM3D is such a code.

For codes that only support double precision, the final Mont-Blanc prototype will use Exynos 5 Dual from Samsung which incorporate a Mali-T604 GPU. They will have a peak performance of about a 100 GFLOPS for a power consumption of 5 Watts. Of course the network has to be accounted for, as well as the cooling and storage, but even an efficiency of 5 or 7 GFLOPS per Watt would be an accomplishment.

B. Auto-Tuning

As we have illustrated in Section V, the peculiarities of this kind of hardware under an HPC workload calls for careful investigation and for the generalization of auto-tuning techniques. Furthermore, Tibidabo and the Mont-Blanc prototype will have GPUs with different designs: an NVIDIA design for Tibidabo and an ARM design for the prototype. The porting and optimization efforts should not be lost when moving from one to the other, which calls for systematic tuning methodology. In such a context, two levels of auto-tuning can be considered:

- Platform specific tuning of the application. The auto-tuning process is run at the compilation of the program on the target platform. This could be called static auto-tuning.
- Instance specific tuning of the application. In many applications, some good optimization parameters depend on the problem size. For instance, optimal buffer size used in GPU kernel could be tuned to match the length of the input problem. Runtime compilation of OpenCL kernels allows for just-in-time generation and compilation of such kernels.

VII. Conclusion

The Mont-Blanc European project is so far a success. HPC application have been successfully ported to Tibidabo an experimental cluster using Tegra 2 SoC. These applications have been shown to require less energy to run using an embedded platform than a classical server processor.

Many of the eleven applications selected in the Mont-Blanc project can already make use of GPGPUs and thus the porting to the extension of Tibidabo should be relatively easy for these applications.

We have also shown that the optimization process can be counter-intuitive and error prone. The use of systematical and precise benchmarking is required in order to understand the behavior of these codes. Auto-tuning of HPC applications is also a must in order to quickly and painlessly adapt to the ever-evolving HPC environment. We cannot hope to achieve interesting energy efficiency if the codes are not well adapted to the platform targeted.

The use of hybrid embedded system on chips in the future Mont-Blanc prototype is a great opportunity for both the HPC community and the embedded computing community to join their strength and address new challenges arising from this context.

Acknowledgment

This project and the research leading to these results is supported by Mont-Blanc project (European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no 288777) and PRACE project (European Community funding under grants RI-261557 and RI-283493).

References

[1] Top500.Org, "Top500," http://www.top500.org.
[2] Green500.Org, "Green500," http://www.green500.org.
[3] "The mont-blanc project," http://www.montblanc-project.eu.
[4] N. Rajovic, N. Puzovic, L. Vilanova, C. Villavieja, and A. Ramirez, "The low-power architecture approach towards exascale computing," in *Proceedings of the second workshop on Scalable algorithms for large-scale systems*. ACM, 2011, pp. 1–2.
[5] A. Duran, E. Ayguadé, R. Badia, J. Labarta, L. Martinell, X. Martorell, and J. Planas, "Ompss: a proposal for programming heterogeneous multi-core architectures," *Parallel Processing Letters*, vol. 21, no. 02, pp. 173–193, 2011.
[6] "The bigdft scientific application," 2012. [Online]. Available: http://inac.cea.fr/LSim/BigDFT/
[7] L. Genovese, A. Neelov, S. Goedecker, T. Deutsch, S. Ghasemi, A. Willand, D. Caliste, O. Zilberberg, M. Rayson, A. Bergman *et al.*, "Daubechies wavelets as a basis set for density functional pseudopotential calculations," *The Journal of chemical physics*, vol. 129, p. 014109, 2008.
[8] H. Nussbaumer, "Fast fourier transform and convolution algorithms," *Berlin and New York, Springer-Verlag.*, vol. 2, 1982.
[9] D. Peter, D. Komatitsch, Y. Luo, R. Martin, N. Le Goff, E. Casarotti, P. Le Loher, F. Magnoni, Q. Liu, C. Blitz *et al.*, "Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes," *Geophysical Journal International*, vol. 186, no. 2, pp. 721–739, 2011.
[10] N. Rajovic, N. Puzovic, A. Ramirez, and B. Center, "Tibidabo: Making the case for an arm based hpc system," Barcelona Supercomputer Center, Tech. Rep., 2012.
[11] Calao Systems, "Snowball," http://www.calao-systems.com/articles.php?lng=fr&pg=6186.
[12] J. Gonzalez, J. Gimenez, and J. Labarta, "Automatic evaluation of the computation structure of parallel applications," in *Parallel and Distributed Computing, Applications and Technologies, 2009 International Conference on*. IEEE, 2009, pp. 138–145.
[13] V. Pillet, J. Labarta, T. Cortes, and S. Girona, "Paraver: A tool to visualize and analyze parallel code," *WoTUG-18*, pp. 17–31, 1995.
[14] M. M. Tikir, L. Carrington, E. Strohmaier, and A. Snavely, "A genetic algorithms approach to modeling the performance of memory-bound computations," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, ser. SC '07. New York, NY, USA: ACM, 2007, pp. 47:1–47:12. [Online]. Available: http://doi.acm.org/10.1145/1362622.1362686
[15] B. Videau, E. Saule, and J.-F. Méhaut, "PaSTeL: Parallel Runtime and Algorithms for Small Datasets," in *2009 International Workshop on Multi-Core Computing Systems (MuCoCoS'09)*. IEEE Computer Society Press, Mar. 2009.
[16] P. Mucci, S. Browne, C. Deane, and G. Ho, "Papi: A portable interface to hardware performance counters," in *Proc. Dept. of Defense HPCMP Users Group Conference*. Citeseer, 1999, pp. 7–10.