The Mont-Blanc Project: Second Phase successfully finished

Until recently, the design and implementation of the majority of the HPC systems have not primarily focused on energy efficiency. However, it is unanimously accepted within the scientific and industrial community that, as system computing capabilities approach the exascale, future HPC centres will severely being constrained by their massive power consumption. The aim of the EU-funded Mont-Blanc project has been to contribute in addressing these significant challenges by designing a new type of computer architecture capable of setting future, global HPC standards in terms of energy efficiency.

During the first phase of the project (October 2011 to June 2015), the focus was on the development and evaluation of an HPC prototype using energy-efficient embedded ARM technology available at the time, as well as porting a set of representative applications to this system. The now successfully completed second phase (October 2013 to January 2017) concentrated on an initial design of the Mont-Blanc Exascale architecture by exploring different alternatives for the compute node as well as advancing the system software stack. In particular, it enabled further development of the OmpSs parallel programming model to automatically exploit multiple cluster nodes, transparent application check-pointing and software-based fault tolerance, and added support for ARMv8 64-bit processors. In addition, the project allowed the development of parallel programming tools for OmpSs and for the adaptation of system software such as job schedulers to the ARM ecosystem. Due to the shifted focus, several new partners joined the Mont-Blanc consortium for the second phase of the project, in particular Allinea, Inria, University of Bristol, and HLRS from the University of Stuttgart. Like the first phase, the second phase of the project was also coordinated by the Barcelona Supercomputing Center (BSC).

An important aspect of Mont-Blanc is hardware prototyping, in particular for the evaluation of the various software components developed in the project. In this regard, the consortium has designed and built a prototype based on 1080 ARMv7 nodes that has been installed at BSC. The interested reader can find more information on the architectural details of the Mont-Blanc system in the InSiDE magazine issue of Autumn 2015 or at the website http://www.montblanc-project.eu. Additionally, in the second phase of the project, researchers from the consortium focused their efforts on the evaluation of small ARMv8-based cluster systems, in order to assess their performance/energy trade-off and to keep track of the evolution of ARM platforms in the HPC domain (with special focus on ARM 64-bit solutions). Examples of these systems include but are not limited to a 16-node Nvidia Jetson TX1 cluster, a 3-node Applied Micro X-Gene 2 cluster and a 4-node Cavium ThunderX cluster. More details on the hardware configuration and system software of these small clusters is available at the project main website.

All three Gauss centres, i.e., HLRS, JSC, and LRZ, have been involved in the second phase of

the Mont-Blanc project. Their overall contributions, detailed in the following sections, mainly focused on:

- Energy-aware scheduling algorithms (LRZ),
- Performance and debugging tools (JSC and HLRS),
- Porting of a scientific application for evaluating the project's developments (HLRS).

Energy-aware scheduling

In line with its vision, LRZ had the primary role of advocating an energy-efficient operation of the system by conducting research activities in Work Package 4 on runtime systems. During the first phase of the project, LRZ developers implemented a fine-grained monitoring tool that, among other system parameters, is capable of retrieving the power consumption of the platform at the granularity of a computing node. In the second phase of the project, LRZ researchers successfully collaborated with Bull/Atos for the deployment and test of three experimental scheduling algorithms on the Mont-Blanc prototype that introduce energy-aware features in the resource and job scheduling management system. Specifically, the Power Adaptive Scheduling (PAS) algorithm, the Energetic Fairshare Scheduling (EFS) algorithm and the Energy Cap Scheduling algorithm have been developed. While the first two algorithms are included in SLURM since version 15.08, the integration of the last one is still work in progress.

The PAS algorithm [1] allows for dynamic adjustment of the instantaneous power consumption of a cluster in order to stay below a pre-defined tolerable power consumption, that is, a "power cap". This is possible by reducing the number of usable resources of the system and/or operating them at lower power. In this way, the scheduler is capable of running jobs only if their consumed power does not contribute in exceeding the defined cap. The original version of the algorithm overestimates the power consumed by compute nodes by using the (theoretical) maximum power consumption for each CPU frequency. The close collaboration between LRZ and Bull/Atos resulted in an improvement of the algorithm by developing a "Power Plugin" that enables the scheduler to acquire the real power consumption of nodes. This yields a more precise estimation of the total cluster power consumption with better assessment of eventual power cap violations, consequently improving the resource utilization of the system. Figure 1 illustrates an example of the advantages offered by this feature with or without the activation of the "Power Plugin".

The EFS algorithm [2] is a modified version of the more common fairshare algorithm where CPU hours are considered as the main resource for accounting users. In EFS, instead, a counter further accumulates the energy consumed by jobs associated with a user and aligns it with the shares of each user account. Energy-efficient users will then be favoured with shorter waiting times in the scheduling queue compared to less energy-efficient ones. In this way, EFS attempts to provide incentives to users for optimizing their codes in order to save more energy.

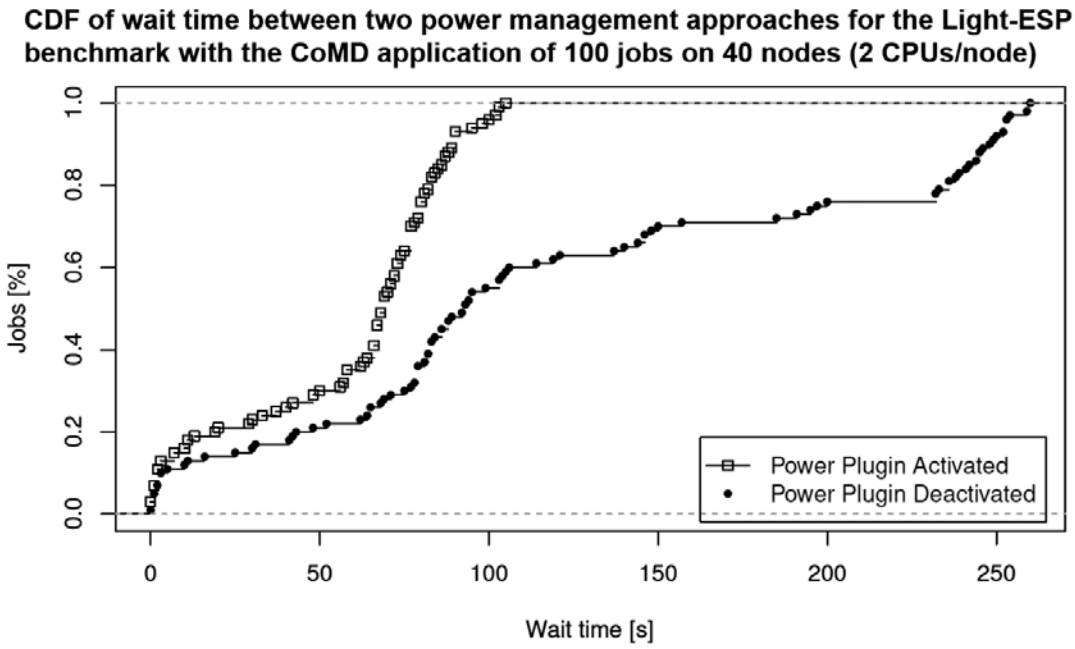

Fig. 1: Cumulative Distribution Function of job waiting time with and without the "Power Plugin". Note that when the plugin is activated, jobs wait less time in the queue, improving the system utilization. This is due to a better estimation of the power consumed by jobs, based on real values retrieved from the Mont-Blanc power monitoring tool, allowing them to start if the power cap is satisfied. When the plugin is deactivated, power consumed by jobs is always estimated as the maximum possible for the selected CPU frequency, increasing the probability of potential power cap violations and increasing the average job waiting time.

Testing of the EFS scheduler on the Mont-Blanc prototype has been possible through the realization of an additional custom "Energy Plugin" in SLURM, which provides the energy consumed by a user's jobs over their execution time. Results successfully demonstrated the effectiveness of the algorithm.

Finally, the Energy Cap Scheduling algorithm [3] implements a mechanism to operate a cluster under energy budget constraints by extending the PAS algorithm concept with energy consumption. In contrast to the PAS algorithm, the Energy Cap algorithm covers more realistic use-case scenarios, where system operators need to establish and maintain an energy budget mainly over a certain period, due to rising electrical energy costs and often due to contractual agreements with providers.

In addition to these activities, LRZ researchers further developed a proof-of-concept for automatic fingerprinting of scientific applications [4], allowing for an optimal selection of the CPU frequency for running user's jobs and consequently contributing to a more energy-efficient

operation of the system. Initial tests show promising results, motivating further research efforts towards this direction.

Performance analysis and debugging tools

The majority of the efforts of HLRS and JSC were in Work Package 5 on development tools, where both partners extended and improved their debugging and performance analysis tools, respectively.

Besides leading this work package, JSC focused on improving the instrumentation and measurement infrastructure Score-P, the Scalasca Trace Tools for automated analysis of event traces, and the performance report explorer Cube including the underlying libraries. As a first step, the aforementioned tool suites were ported to the 64-bit ARMv8 architecture. Moreover, the Cube software design was reworked to allow for a better extensibility through a newly developed plugin architecture and API [5].

While this plugin architecture is now used by all default Cube-internal views (e.g., system tree, box plot, and topology view), it also allows the development of additional views without touching the main Cube code base. For example, the Cube plugin interface has been used by BSC to develop two new visualizations for the results of their Folding tool [6], which combines coarse-grained sampling measurements with phase instrumentation from iterative applications to statistically improve the accuracy of the measurement results for each phase.

Furthermore, JSC actively participated in the OpenMP tools working group on the definition of the OpenMP tools interface (OMPT), which was voted into the "OpenMP Version 5.0 Preview 1" technical report [7] by the OpenMP Architecture Review Board. OMPT will enable tools to reliably work across different implementations of the OpenMP API. In particular, the Mont-Blanc partners BSC and JSC contributed a proposal to track tasks and their dependencies, BSC implemented draft versions of the OMPT API in their Nanos++ OmpSs runtime, and JSC developed a Score-P prototype based on the OMPT interface, which was subsequently validated with both the Nanos++ runtime as well as an extended version of the LLVM OpenMP runtime developed by the OpenMP tools working group.

With respect to the Score-P instrumentation and measurement infrastructure, JSC enhanced the prototypical support for the OmpSs programming model already developed during the first phase of the Mont-Blanc project. Most notably, the ability to support OmpSs@Cluster mode, where tasks are automatically distributed across a set of worker nodes by the OmpSs runtime, has been evaluated and an initial prototype has been implemented. In addition, Score-P has been enhanced to measure OmpSs tasks offloaded to GPU accelerators. The latter also motivated improvements regarding an integrated analysis and presentation of hybrid applications using multiple programming models—such as MPI, OpenMP or OmpSs, and CUDA or OpenCL—in combination.

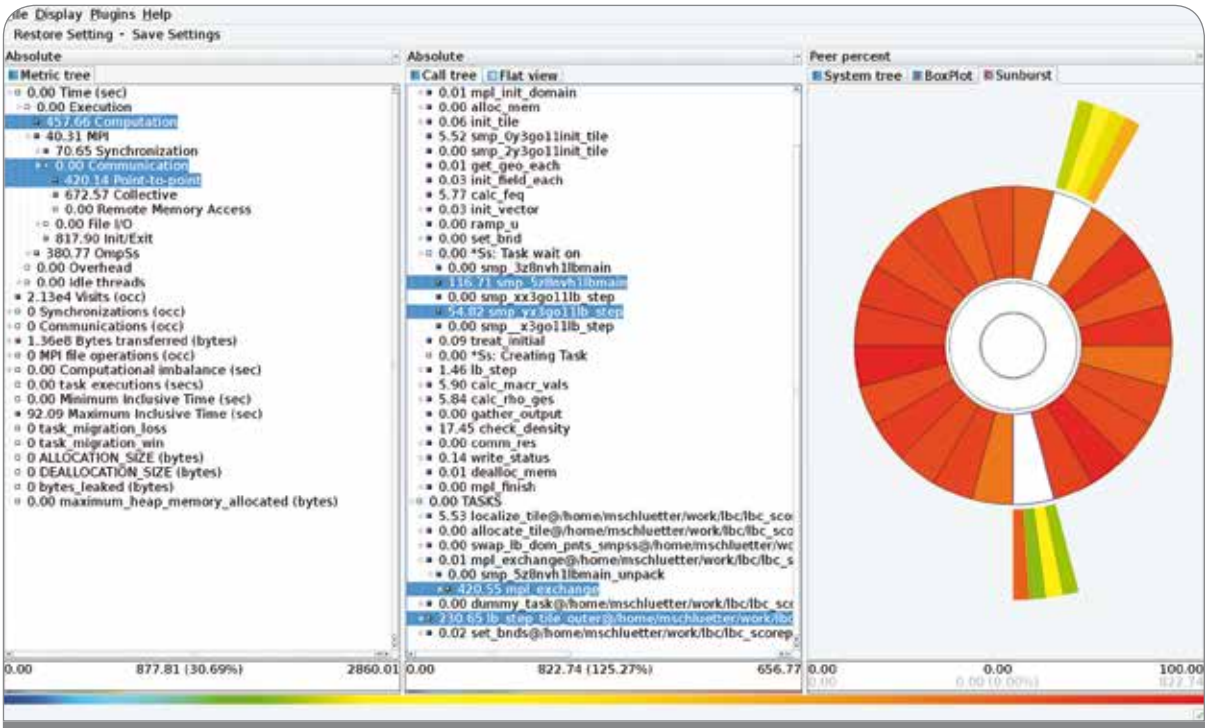

Fig. 2: Screenshot of the Cube performance report browser showing a profile of the OmpSs version of the LBC code measured on the Cavium ThunderX mini-cluster. The left pane shows the various performance metrics collected, the middle pane shows the call tree of the main application (top) as well as the asynchronously executed tasks (bottom), and the right pane shows the "Sunburst" diagram as an alternative visualisation for the system tree, implemented as a plugin using the newly defined Cube plugin API.

HLRS, on the other hand, concentrated on development of Temanejo [8] (Figure 3), a graphical debugger for task-based programming models. The foremost purpose of Temanejo is to display the task-dependency graph of applications, and to allow simple interaction with the runtime system in order to control various aspects of the parallel execution of an application.

Today's parallel programming models offer high-level concepts, such as task and data-dependencies, to the developers of parallel applications. Traditional debuggers, on the other side, aim to support a large variety of programming languages and models, and thus need to use the lowest common denominator, such as system calls and POSIX threads. Developers thus can only bridge this semantic gap by having specialised skills and know-how on both sides. Model-centric debugging closes the semantic gap between high-level programming model and the debugging by simply using the same high-level concepts and semantics. In particular, model-centric debuggers represent the state of an application in terms of the programming model and support interaction using its semantics.

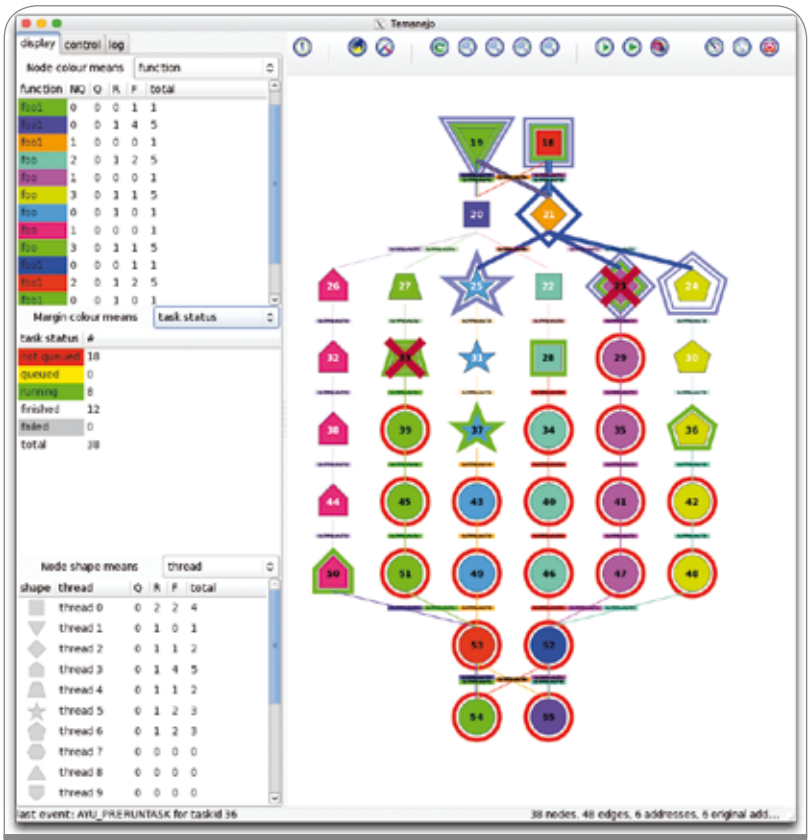

Fig. 3: Screenshot of Temanejo showing a basic dependency graph. The node color represents the function names, the margin color represents the task states and shape represents the different threads.

Temanejo was redesigned to support multi-process debugging with multiple attached programming models. Therefore, HLRS had to develop a completely new backend library, called Ayudame. This library is now capable of interfacing with different programming models like OmpSs, StarPU, OpenMP and also MPI. In fact, Ayudame may use the aforementioned monitoring interface OMPT to intercept events from OpenMP and OmpSs runtimes. The native event system of OmpSs continues to be supported. In order to control the programming model's behaviour, HLRS developed the Tasking Control Interface (TCA) [9] as an extension of the OMPT interface to allow interoperability.

Application porting for evaluation

As part of Work Package 3 on applications, HLRS took an existing version of the LBC code and parallelised it using OmpSs. During the parallelisation process, HLRS developed three different code versions. The first version is a very basic implementation and is based on the fork&join model. This version needs to synchronise all tasks before the communication with the neighbouring processes can be started. The second version hides the communication inside the computation. Therefore, only the tasks necessary for communication need to be synchronised before the communication with the neighbouring processes can be started. After the communication, the synchronisation with the remaining tasks is necessary. The third version is similar to version two, but in addition the OmpSs programming model is aware of the underlying MPI communication inside tasks. This feature allows the programming model to interrupt the communication task and execute computation tasks while waiting for the MPI communication to finish.

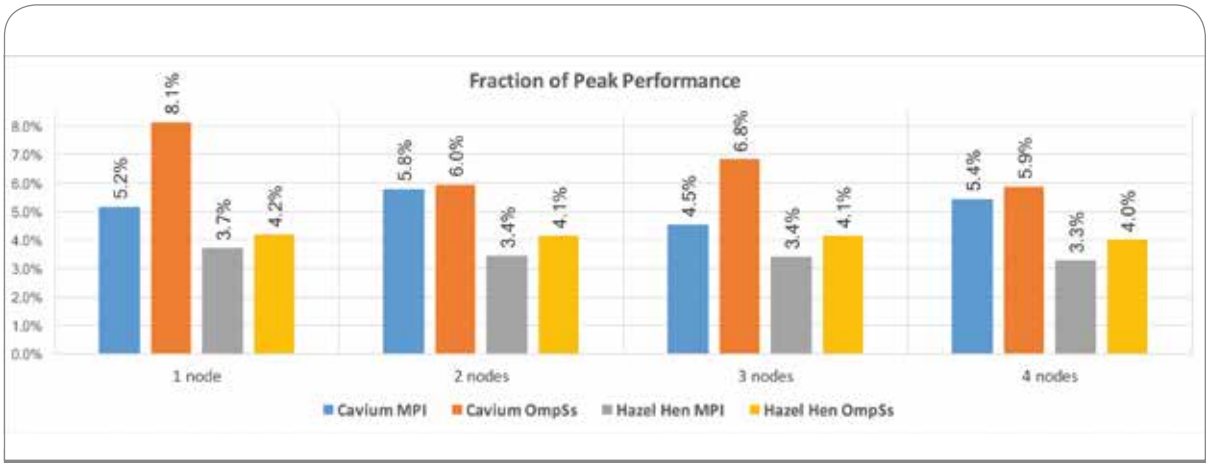

Fig. 4: Percentage of peak performance the LBC application reaches on the respective target platform. As theoretical peak performance for the Cavium ThunderX system 345 GFLOP/s and 960 GFLOP/s for the Cray XC40 is used. Normalising the result to the theoretical peak performance shows a big benefit for the Cavium ThunderX system and especially the hybrid OmpSs/MPI version gains a lot from this architecture.

All of the above-described versions were implemented with the help of Temenajo and the tools available in the Mont-Blanc consortium. A performance analysis done by JSC helped us to detect and solve a performance-critical issue. Figure 3 shows a comparison of LBC on two different platforms, the Hazel Hen (Cray XC40) located at HLRS and the ARM-based Cavium ThunderX mini-cluster. On both platforms, the hybrid OmpSs+MPI implementation shows better performance than a pure MPI version.

Impact

For the three Gauss centres, the second phase of the Mont-Blanc project has been an excellent environment to continue established research lines and develop methods and software packages to a state which is ready for production environments. For instance, the energy-aware scheduling algorithms have been pushed upstream to SLURM since version 15.08 and more solutions and improvements of the existing mechanisms and on application fingerprinting are expected to become available in future releases. Similarly, the extensions made to the performance analysis and debugging tools Cube, Scalasca, Score-P, and Temanejo are either already available or will be released soon. In general, the Gauss centres are committed to release all software developed within Mont-Blanc funding under an open-source license. The end of the second phase of Mont-Blanc establishes an important milestone, which marks the availability of necessary system middleware/software for research and development in the next phase of the project. Most importantly however, the standardisation of the OMPT interface has a significant impact on the HPC community as a whole. It greatly simplifies the development of new tools for OpenMP via a portable interface across different runtimes, including the Nanos++ OmpSs runtime.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under the Mont-Blanc project (http://www.montblanc-project.eu), grant agreement n° 288777 and n° 610402.

References

[1] **Y. Georgiou et al.:**
"Adaptive Resource and Job Management for Limited Power Consumption," IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW) 2015, 25-29 May 2015, Hyderabad, India.

[2] **Y. Georgiou et al.:**
"A Scheduler-Level Incentive Mechanism for Energy Efficiency in HPC," 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) 2015, 4-7 May 2015, Shenzhen, China.

[3] **P. F. Dutot et al.:**
"Towards Energy Budget Control in HPC," 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) 2017, 14-17 May 2017, Madrid, Spain (accepted for publication).

[4] **The Mont Blanc Consortium:**
"Deliverable D4.9: Final Evaluation of the Compiler and the Runtime System," the Mont-Blanc Project, January 2017.

[5] **P. Saviankou, M. Knobloch, A. Visser, B. Mohr:**
"Cube v4: From Performance Report Explorer to Performance Analysis Tool", Procedia Computer Science, 51:1343-1352, June 2015.

[6] **H. Servat, G. Llort, J. Gimenez, K. A. Huck, J. Labarta:**
"Unveiling Internal Evolution of Parallel Application Computation Phases", ICPP 2011: 155-164.

[7] **OpenMP Architecture Review Board:**
"OpenMP Technical Report 4: Version 5.0 Preview", Nov 2016.

[8] **S. Brinkmann, J. Gracia, C. Niethammer, R. Keller:**
"TEMANEJO - a debugger for task based parallel programming models," Oct 2012.

[9] **M. Nachtmann, J. Gracia:**
Enabling Model-Centric Debugging for Task-Based Programming Models – a Tasking Control Interface," in "Tools for High Performance Computing 2015, Proceedings of the 9th International Workshop on Parallel Tools for High Performance Computing", Sep 2016.

Written by Daniele Tafani[1], Marc Schlütter[2], Markus Geimer[2], Bernd Mohr[2], Mathias Nachtmann[3] and José Gracia[2]

[1]Leibniz Supercomputing Centre
[2]Jülich Supercomputing Centre
[3]Höchstleistungsrechenzentrum Stuttgart

Contact: José Gracia, gracia@hlrs.de