The Mont-Blanc prototype: An Alternative Approach for HPC Systems

Nikola Rajovic*†, Alejandro Rico‡★, Filippo Mantovani*, Daniel Ruiz*†, Josep Oriol Vilarrubi*, Constantino Gomez*†, Luna Backes*, Diego Nieto*, Harald Servat*, Xavier Martorell*†, Jesus Labarta*†, Eduard Ayguade*†, Chris Adeniyi-Jones‡, Said Derradji§, Herve Gloaguen§, Piero Lanucara¶, Nico Sanna¶, Jean-Francois Mehaut‖, Kevin Pouget‖, Brice Videau‖, Eric Boyer**, Momme Allalen††, Axel Auweter††, David Brayford††, Daniele Tafani††, Volker Weinberg††, Dirk Brömmel‡‡, René Halver‡‡, Jan H. Meinke‡‡, Ramon Beivide[xi], Mariano Benito[xi], Enrique Vallejo[xi], Mateo Valero*† and Alex Ramirez[x]★

* *Barcelona Supercomputing Center,* `first.last@bsc.es`

† *Computer Architecture Department, Universitat Politecnica de Catalunya - BarcelonaTech*

‡ *ARM* § *Bull/ATOS* ¶ *Cineca* ‖ *CNRS - Universite Grenoble Alpes - LIG* ** *GENCI*

‡‡ *Forschungszentrum Jülich GmbH* †† *LRZ* [x] *NVIDIA* [xi] *Universidad de Cantabria*

Abstract—**High-performance computing (HPC) is recognized as one of the pillars for further progress in science, industry, medicine, and education. Current HPC systems are being developed to overcome emerging architectural challenges in order to reach Exascale level of performance, projected for the year 2020. The much larger embedded and mobile market allows for rapid development of intellectual property (IP) blocks and provides more flexibility in designing an application-specific system-on-chip (SoC), in turn providing the possibility in balancing performance, energy-efficiency, and cost. In the Mont-Blanc project, we advocate for HPC systems being built from such commodity IP blocks, currently used in embedded and mobile SoCs.**

As a first demonstrator of such an approach, we present the Mont-Blanc prototype; the first HPC system built with commodity SoCs, memories, and network interface cards (NICs) from the embedded and mobile domain, and off-the-shelf HPC networking, storage, cooling, and integration solutions. We present the system's architecture and evaluate both performance and energy efficiency. Further, we compare the system's abilities against a production level supercomputer. At the end, we discuss parallel scalability and estimate the maximum scalability point of this approach across a set of applications.

1. Introduction

The evolution of High-Performance Computing (HPC) systems is driven by the need of reducing time-to-solution and increasing the resolution of models and problems being solved by a particular program. Important milestones from the HPC system performance perspective were achieved using commodity technology. Examples are the ASCI Red and the Roadrunner supercomputers, which broke the 1 TFLOPS and 1 PFLOPS barriers, respectively. These systems showed how commodity technology could be used to take the next step in HPC system architecture.

Driven by a much larger market, commodity components evolve faster than their special-purpose counterparts, eventually achieving the same performance and eventually surpassing or replacing them. For this reason, RISC processors displaced vector processors, and x86 displaced RISC.

Nowadays commodity is in the embedded / mobile processor segment. Mobile processors develop fast, and are still not at a point of diminishing performance improvements from new designs. Furthermore, they progressively incorporate the capabilities required for HPC.

The embedded market size and endless customer requirements allow for constant investments into innovative designs, and rapid testing and adoption of new technologies. For example, LPDDR memory technology was first introduced in the mobile domain and has recently been proposed as a memory solution for energy proportional servers [1].

The Mont-Blanc project aims at providing an alternative HPC system solution based on the current commodity technology: mobile chips. As a demonstrator of such an approach, the project designed, built, and set-up a 1080-node HPC cluster made of Samsung Exynos 5250 SoCs. The Mont-Blanc project established the following goals: to design and deploy a sufficiently large HPC prototype system based on the current mobile commodity technology; to port and optimize the software stack, and enable its use for HPC; to port and optimize a set of HPC applications to be run at this HPC system.

Comparing the Mont-Blanc prototype to a contemporary supercomputer, MareNostrum III, reveals that a single-socket Mont-Blanc node is 9× slower than a dual-socket MareNostrum III node, while saving up to 40% of energy. MPI parallel applications show a 3.5× slowdown when running with the same number of MPI ranks on both machines, while consuming 9% less energy on the Mont-Blanc prototype on average. When targeting the same execution time, the Mont-Blanc prototype offers 12.5% space savings.

★ *The majority of the work was done while the author was with Barcelona Supercomputing Center.*

The contributions of this paper are:

- A detailed description of the Mont-Blanc prototype architecture
- A thorough performance and power evaluation of the prototype, comparing it to a Tier-0 production system in Europe, the MareNostrum III supercomputer.
- A set of recommendations for the next-generation HPC system built around the Mont-Blanc approach.

The rest of this paper is structured as follows. In Section 2, we reveal the architecture of the Mont-Blanc prototype. In Section 3, we evaluate the performance and energy of a Mont-Blanc node and compare it to a MareNostrum III node. In Section 4, we present the evaluation and tuning of the prototype's node interconnect. In Section 5, we evaluate the parallel scalability of our system, compare its performance and energy efficiency against MareNostrum III, and discuss the influence of its lossy interconnect. With Section 6 we extrapolate parallel scalability beyond the size of our prototype, envisioning future systems built around the same approach. Related work is discussed in Section 7. Finally, with Section 8 we conclude the paper.

2. The Mont-Blanc Prototype

In this section we present the architecture of the Mont-Blanc prototype (shown in Figure 1). We highlight peculiarities of each building block as we introduce them.

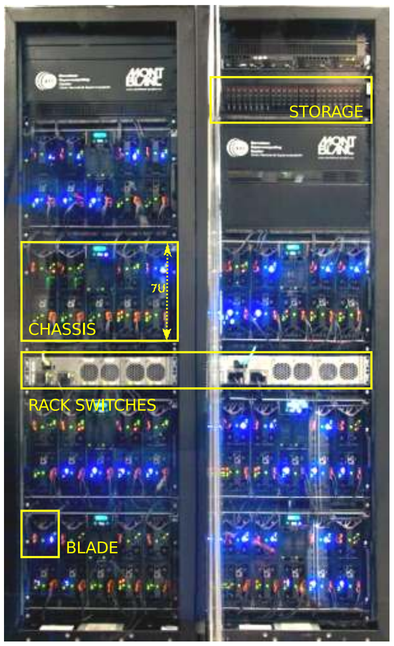


Figure 1: Physical view of the Mont-Blanc system.

2.1. The Mont-Blanc Compute Node

The Mont-Blanc compute node is a *Server-on-Module* architecture. Figure 3 depicts the Mont-Blanc node card

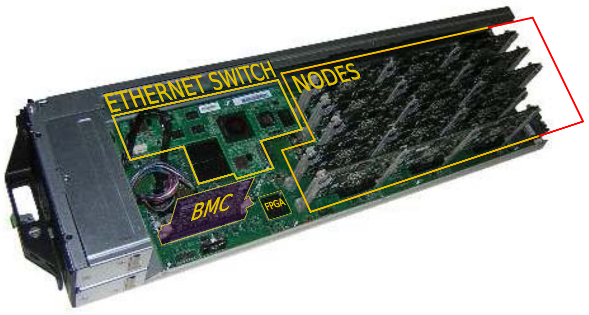


Figure 2: Physical view of the Mont-Blanc blade.

(Samsung Daughter Board or SDB) and its components. Each SDB is built around a Samsung Exynos 5250 SoC integrating two ARM Cortex-A15[1] CPUs @ 1.7 GHz sharing 1 MB of on-die L2 cache, and a mobile 4-core ARM Mali-T604 GPU @ 533MHz. The SoC connects to the on-board 4 GB of LPDDR3-1600 RAM through two 32-bit memory channels shared among the CPUs and GPU, providing a peak memory bandwidth of 12.8 GB/s.

The node interconnect is provided by the ASIX AS88179 USB 3.0 to 1 Gb Ethernet bridge, and an Ethernet PHY. An external 16 GB μSD card provides the boot-loader, OS system image, and local scratch storage.

The node connects to the blade through a proprietary bus using a PCI-e 4x form factor edge connector (EMB connector).

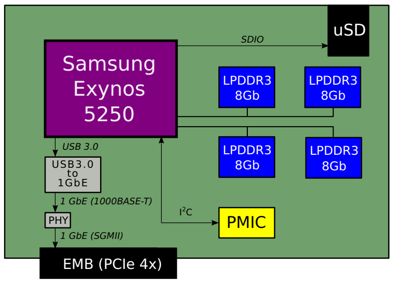


Figure 3: The Mont-Blanc node block scheme (not to scale).

2.2. The Mont-Blanc Blade

Figure 4 describes the architecture of the Mont-Blanc blade (Ethernet Mother Board, or EMB, depicted in Figure 2). The blade hosts 15 Mont-Blanc nodes which are interconnected through an on-board 1 GbE switch fabric. The switch provides two 10 GbE up-links. In addition, the EMB provides management services, power consumption monitoring of SDBs, and blade level temperature monitoring. The EMB enclosure is air-cooled through the fans installed on the front side.

1. Implementation of the ARMv7-a architecture.

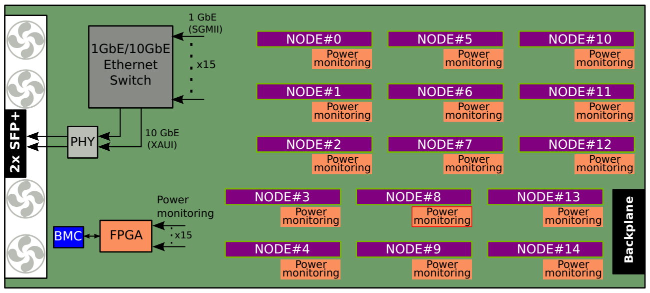


Figure 4: The Mont-Blanc blade block scheme.

2.3. The Mont-Blanc System

The entire Mont-Blanc prototype system (shown in Figure 1) fits into two standard 42U-19″ racks. Each Mont-Blanc rack hosts up to four 7U Bullx chassis which in turn integrate nine Mont-Blanc blades each. In addition, racks are populated with two 2U 10 GbE Cisco Nexus 5596UP top-of-the-rack (TOR) switches, one 1U prototype management 1GbE switch[2], and two 2U storage nodes.

2.3.1. System interconnect. The Mont-Blanc prototype implements two separate networks: the GbE management network, and the 10 GbE MPI network. The management network is out of the scope of this paper, thus we depict the implementation of the MPI interconnect in Figure 5.

The first level of switching is provided inside the blades using a 1 GbE switch fabric providing two 10 GbE up-links. Switching between the blades occurs at the TOR switches with a switching capacity of 1.92 Tbps per switch. The racks are directly connected with four 40 GbE links.

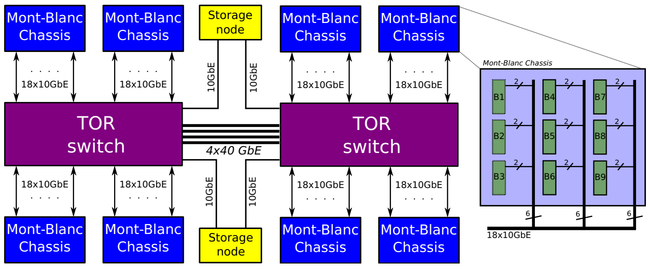


Figure 5: The Mont-Blanc system interconnect.

2.3.2. Storage. The Lustre parallel filesystem is built on a Supermicro Storage Bridge Bay based on x86-64 architecture, with a total capacity of 9.6 TB providing 2-3.5 GB/s read/write bandwidth (depending on the disk zone). The storage system is connected to the top-of-the-rack switches with four 10 GbE links.

2.3.3. Cooling. Compute nodes are passively cooled using a top-mounted heat sink, while blades provide active air-cooling through variable speed front-mounted fans in a temperature control loop.

2. Not visible, mounted on the back.

2.4. The Mont-Blanc Software Stack

Compilers	
GNU JDK Mercurium	
Scientific libraries	
ATLAS LAPACK SCALAPACK FFTW BOOST clBLAS clFFT PETSc HDF5	
Performance analysis	**Debugger**
EXTRAE Paraver Scalasca	Allinea DDT
Runtime libraries	
Nanos++ OpenCL OpenMPI MPICH3	
Cluster management	
SLURM Nagios Ganglia	
Hardware support	**Storage**
Power monitor	LustreFS
Operating System	
Ubuntu	

Figure 6: The Mont-Blanc software stack.

The work done during the Mont-Blanc project helped to mature the HPC software stack on the ARM architecture. Today, working with the Mont-Blanc prototype feels like working with any other HPC cluster.

The Mont-Blanc prototype nodes run Ubuntu 14.04.1 Linux on top of the customized Linaro Kernel version 3.11.0 which enables a user space driver for OpenCL programming of the ARM Mali-T604 GPU. The rest of the software stack components are shown in Figure 6.

A very relevant part of the Mont-Blanc software stack is the OmpSs programming model [2], provided by the Mercurium compiler and the Nanos++ runtime. OmpSs is a task-based programming model with explicit inter-task dataflow that allows the runtime system to orchestrate out-of-order execution of the tasks, selectively off-loading of tasks to the GPU when possible, or running them on the CPU if the GPU is busy. Applications ported to OmpSs can make simultaneous use of the CPU and the GPU, dynamically adapting to load imbalance situations.

2.5. Power Monitoring Infrastructure

The Mont-Blanc prototype provides a unique infrastructure for high-frequency measurements of power consumption at the granularity of a single compute node, scaling to the whole size of the prototype.

The Mont-Blanc system features a digital current and voltage meter in the power supply rail to each SDB. An FPGA on each EMB accesses the power sensors in each SDB via I2C and stores the averaged values every 1,120ms in a FIFO buffer. The Board Management Controller (BMC) on the EMB communicates with the FPGA to collect the power data samples from the FIFO buffer before storing them in its DDR2 memory along with a timestamp of the reading. User access to the data is then provided by the BMC over the management Ethernet through a set of custom Intelligent Platform Management Interface (IPMI) commands.

To provide application developers with power traces of their applications, the power measurement and acquisition process is conveniently encapsulated and automated in a

TABLE 1: Mont-Blanc compute performance summary.

	CPU	GPU
Compute Node		
Compute element	2×ARM Cortex-A15	1×ARM Mali-T604
Frequency	1.7 GHz	533 MHz
Peak performance (SP)	27.2 GFLOPS	72.5 GFLOPS
Peak performance (DP)	6.8 GFLOPS	21.3 GFLOPS
Memory (shared)	4 GB LPDDR3-800	
Blade = 15×Node		
Peak performance (SP)	408 GFLOPS	1.08 TFLOPS
Peak performance (DP)	102 GFLOPS	319.5 GFLOPS
Memory	60 GB	
Chassis = 9×Blade		
Peak performance (SP)	3.67 TFLOPS	9.79 TFLOPS
Peak performance (DP)	0.92 TFLOPS	2.88 TFLOPS
Memory	540 GB	
System = 8×Chassis		
Peak performance (SP)	29.38 TFLOPS	78.3 TFLOPS
Total (SP)	*107.7 TFLOPS*	
Peak performance (DP)	7.34 TFLOPS	23 TFLOPS
Total (DP)	*30.3 TFLOPS*	
Memory	4.32 TB	

custom-made system monitoring tool. The tool is developed with a focus on simplicity and scalability by respectively employing MQTT [3], for lightweight transport messaging, and Apache Cassandra, a scalable, distributed database for storing the acquired power data along with other time-series based monitoring data. A set of command line tools and a special API provide users with the ability to access the raw monitoring data or to plot and correlate information from different data sources throughout the system.

2.6. Performance Summary

Table 1 shows the performance figures of the Mont-Blanc prototype. The two Cortex-A15 cores provide a peak performance of 27.2 GFLOPS in single-precision (SP) and 6.8 GFLOPS in double-precision (DP). The performance disparity comes from the fact that the SIMD unit, code-named NEON, only supports SP floating-point (FP) operations, thus DP FP instructions execute in a scalar unit.

The on-chip 4-core Mali-T604 GPU provides 72.5 GFLOPS SP and 21.3 GFLOPS DP [4]. The total node performance is 99.7 GFLOPS SP and 28.1 GFLOPS DP.

Table 1 shows the peak performance at the blade, chassis and entire system levels for CPU and GPU separately. The whole system has a peak performance of 107.7 TFLOPS SP and 30.3 TFLOPS DP.

Due to the 32-bit nature of the SoC architecture, each node integrates only 4 GB of memory. The high node integration density of 1080 nodes (2160 cores) in 56U (over 19 nodes per U) adds up to 4.32 TB of memory, and an aggregate 13.8 TB/s memory bandwidth.

3. Compute Node Evaluation

In this section, we present a comparison between the Samsung Exynos 5250 SoC[3] used in the Mont-Blanc prototype and its contemporary 8-core Intel Xeon E5-2670 [4]

3. Introduced in Q3 2012
4. Introduced in Q1 2012

TABLE 2: Mont-Blanc vs MareNostrum III: node performance comparison.

	Mont-Blanc		*MareNostrum III*	
Frequency [GHz]	1.7		2.6	
# sockets	1		2	
	CPU	GPU	CPU	GPU
Peak FP-64 [GFLOPS]	6.8	21.3	332.8	*-n/a-*
Memory BW [GB/s]	12.8		51.2	
Network BW [Gb/s]	1		40	
Intersocket BW [GB/s]	*-n/a-*		32	

TABLE 3: List of Mont-Blanc benchmarks

Tag	Full name
2dc	2D convolution
amcd	Markov Chain Monte Carlo method
dmm	Dense matrix-matrix multiplication
hist	Histogram calculation
ms	Generic merge sort
nbody	N-body calculation
3ds	3D volume stencil computation
fft	One-dimensional Fast Fourier Transform
red	Reduction operation
vecop	Vector operation

server processor running at 2.6 GHz used in the MareNostrum III supercomputer [5]. A MareNostrum node is a dual-socket implementation using DDR3-1600 memory DIMMs. See more details in the side-by-side comparison in Table 2.

Methodology: We present and discuss both core-to-core, and node-to-node performance as well as energy figures when executing the Mont-Blanc benchmark suite [6](see Table 3). We report performance (execution time) and energy differences by normalizing to that of MareNostrum. We obtain node power using the power monitoring infrastructure of the Mont-Blanc prototype (see Section 2.5) , and the node energy consumption in MareNostrum provided through LSF job manager.

3.1. Core Evaluation

In Figure 7, we present the performance comparison on a core-to-core basis between the Mont-Blanc prototype and MareNostrum supercomputer. This comparison using single-threaded runs gives a sense of the performance difference between both cores without the interference of scheduling and synchronization effects of parallel runs.

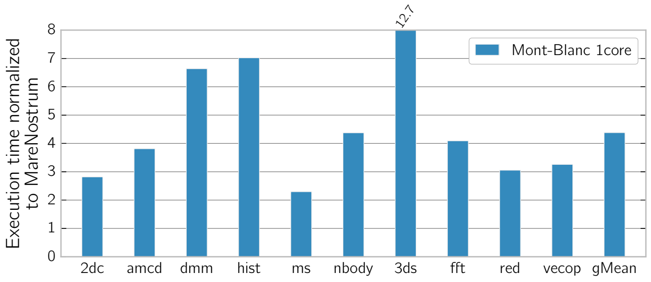
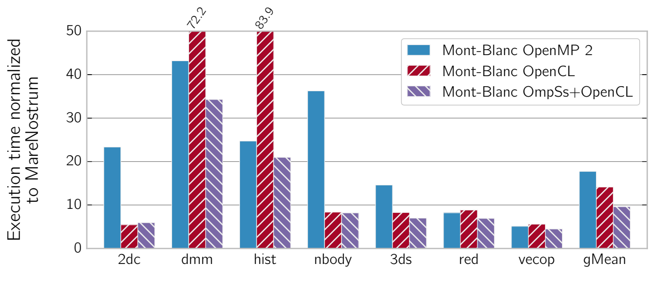
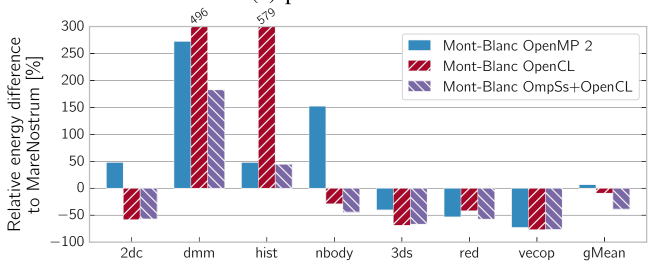


Figure 7: Mont-Blanc benchmarks: core-to-core performance comparison.

Across the benchmark suite, Mont-Blanc is between 2.2 and 12.7 times slower on a per core basis. The Cortex-A15



(a) performance



(b) energy

Figure 8: Mont-Blanc benchmarks: node-to-node a) performance and b) energy comparison.

core underperforms the Intel Sandy Bridge mainly due to: lack of SIMD DP FP extensions (vectorization observed in *dmm*, *3ds*, *fft*, *red*, *vecop*); lower per socket memory bandwidth (12.8 vs 51.2 GB/s); and limited memory subsystem resources geared towards low power[5] (more off-chip accesses observed in *2dc*, *amcd*, *hist*, *nbody*). On average, across the entire suite, a Mont-Blanc processor core is 4.3x slower than that of MareNostrum.

3.2. Node Evaluation

In Figure 8, we compare performance and energy consumption on a node-to-node basis between the Mont-Blanc prototype and the MareNostrum III supercomputer.

Given the characteristics of the Mont-Blanc SoC and its software stack, we evaluate three different computing scenarios: homogeneous CPU computing with OpenMP (blue bars), heterogeneous CPU + GPU with OpenCL (red bars), and heterogeneous with OmpSs (violet bars).

Comparing CPU-only computing, a dual-core Mont-Blanc node is on average 18x slower than a 16-core MareNostrum node. When using OpenCL to off-load all compute tasks to the GPU, Mont-Blanc is 14x slower than MareNostrum. Finally, using OmpSs to efficiently offload computation to both the GPU and the CPU, we significantly reduce the gap to only 9x across the benchmark suite.

Energy-wise, when using only CPUs, a Mont-Blanc node consumes 7% more energy compared to a MareNostrum node. As we close the performance gap, Mont-Blanc nodes become more energy efficient on average: from consuming 10% less energy when using only GPU, to consuming 40% less energy when using both GPU and CPU cores.

5. Intel Xeon E5-2670 features 20 MB of a third level cache.

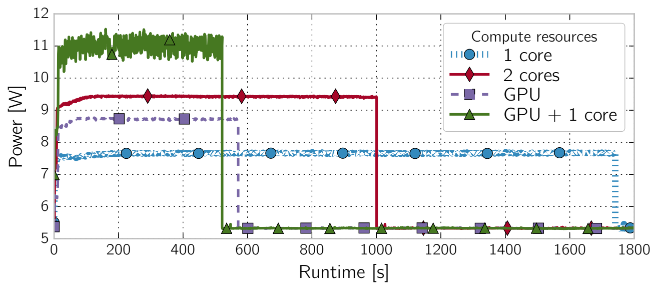


Figure 9: Power profile of different compute to hardware mappings for 3D-stencil computation. Note: markers are only to distinguish lines, not actual sampling points.

Our results show that, when using the embedded GPU, Mont-Blanc can be significantly more energy-efficient than a homogeneous cluster like MareNostrum III. However, Mont-Blanc needs applications to scale to 10-15x more nodes in order to match performance, and interconnection network performance is critical for that.

3.3. Node Power Profiling

Energy has two dimensions: power and time. Execution time depends on how the application performs on the underlying architecture. Power depends on how much the application stresses compute resources, processor physical implementation and SoC power management. The power monitoring infrastructure in the Mont-Blanc prototype (Section 2.5) helps the user to reason about both factors. Comparing the power of different mappings[6] (CPU, GPU, or CPU+GPU), the user can estimate the speed-up required to compensate the power differences and run the system at the best energy efficiency point.

Figure 9 shows a high sampling rate power profile of one Mont-Blanc node for different mappings of the execution of the 3D-stencil benchmark. The different mappings include one CPU core (sequential), dual core (OpenMP), GPU (OpenCL), and GPU + 1 CPU (OmpSs).

Node idle power is 5.3W. This includes the static power of all components given that frequency scaling is disabled for benchmarking purposes. The average power consumption when running on one and two CPU cores is 7.8W and 9.5W respectively. This includes the power consumption of the SoC, memory subsystem and network interface.

Node power when using the GPU and the GPU + 1 CPU is 8.8 and 11W, respectively. When running on the GPU alone, one of the cores is still active as a helper thread that synchronously launches kernels to the GPU and therefore blocks until they complete. When running OmpSs on the GPU + 1 CPU, one of the cores is the GPU helper and the other one runs a worker thread and contributes to computation, thus adding that extra power.

Our results show that the extra power required by OmpSs because of adding one CPU core to GPU computation could outweight the performance improvement, in turn leading to a

6. Counting only elements contributing to the computing.

higher energy-to-solution in the 3D stencil (3ds) benchmark (as we show in Figure 8).

From our results with other benchmarks, node power varies across different workloads although it remains in the same range seen in Figure 9. The maximum power seen for executions with two CPU cores is 14W, and 13.7W for executions with GPU plus one CPU core.

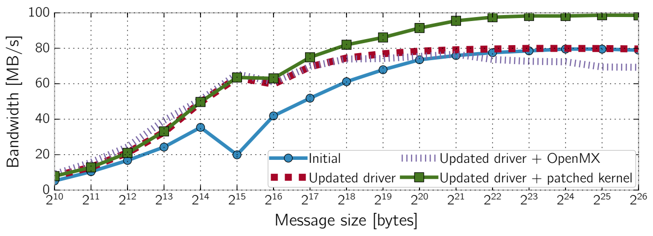
This shows the relevance of the power measurement infrastructure in the Mont-Blanc prototype. It allows us to explain where and how the power is being spent, even at high frequencies. The ability to visualize power over time is even more valuable for applications showing different phases that may benefit of different CPU-GPU mappings. This way, the user can identify the best mapping for each application phase.

In systems without a power profile (which just provide the total job energy consumption), such analysis requires a less accurate and time-consuming trial-and-error approach looking at power deltas over multiple runs of different configurations.

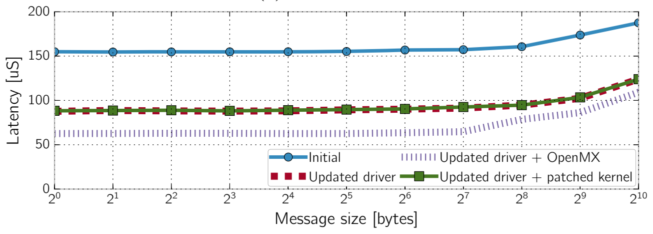
4. Interconnection Network Tuning and Evaluation

In this section, we quantify the latency and bandwidth of the Mont-Blanc interconnection network. Since the Mont-Blanc interconnect is implemented using a lossy Ethernet technology, it is of a paramount importance that every layer is properly tuned. Thus we discuss the improvements in different parts of the interconnect stack which in turn affect the overall interconnect performance.

In Figure 10, we present both bandwidth and latency measurements obtained from the Mont-Blanc prototype using the Intel MPI PingPong benchmark. We present four curves per graph, each corresponding to incremental improvements on the node network interface.



(a) Bandwitdh



(b) Latency

Figure 10: Inter-node bandwidth and latency of the Mont-Blanc prototype.

TABLE 4: MPI applications used for scalability evaluation.

Application	Domain
BigDFT [11], [12]	Electronic Structure
BQCD [13]	Quantum Chromodynamics
MP2C [14]	Multi-Particle Collision Dynamics
QuantumESPRESSO [15]	Electronic Structure and Materials Modeling
SMMP [16], [17], [18]	Molecular Thermodynamics
Alya [19], [20]	Biomedical Mechanics
COMD [21]	Proxy for Molecular Dynamics
LULESH [22], [23]	Proxy for Hydrodynamics
miniFE [24]	Proxy for Finite Element Method

After the initial deployment of the Mont-Blanc prototype we measured the achievable MPI throughput and latency of 80 MB/s and 156 μs respectively (blue line, solid with circles). The results were obtained using the NIC driver built into the Linux Kernel.

We updated the driver using a proprietary version provided by the USB-to-GbE bridge maker[7], achieving significant improvements in both throughput and latency for small messages: up to 3.4x better throughput for messages under 64KB, and only 88μs latency for zero-sized messages (red line, dashed). However, throughput for larger messages stayed the same as in the initial configuration. The new driver provides a configurable wait interval between the consecutive bulk transfers on the USB bus, which we reduced to the bare minimum.

Additionally, we did a back-port of a Linux Kernel patch [7] which improves throughput in USBNET driver for USB 3.0 compliant devices. This patch improved throughput for messages larger than 64 KB, achieving a maximum throughput of 100 MB/s (green line, solid with squares). For reference purposes, the same benchmark run on a server class x86_86 system (with integrated 1 GbE NIC) achieves 112.5 MB/s and a 46.5μs latency [8], so we achieve 89% of the potential bandwidth, but our latency is still 1.9x higher.

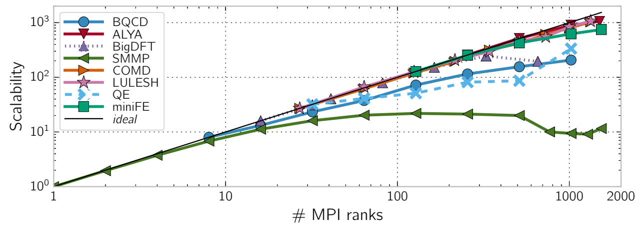
Most of the ping-pong latency is due to the TCP/IP protocol stack, which runs on the ARM Cortex-A15 CPU. We also deployed the Open-MX [9] protocol stack (a free implementation of the Myricom protocol) to replace TCP/IP. The lighter-weight protocol reduced latency for small messages to 65 μs, which also increases bandwidth for messages under 32 KB. However, it degrades the throughput for the larger message sizes (violet line, dotted).

Since most of our MPI applications will exchange large messages, we prefer to optimize bandwidth over latency and select the proprietary driver + patched USBNET kernel for the stable network configuration of the prototype.

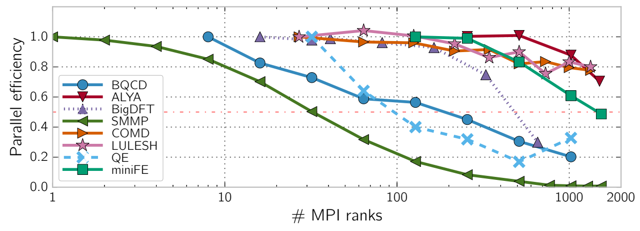
5. Overall System Evaluation

In this section, we evaluate the Mont-Blanc cluster using full-scale, production MPI applications, as listed in Table 4, plus three reference mini-apps used by US DOE National labs [10]. All test applications use OpenMPI, and run on the CPU only.

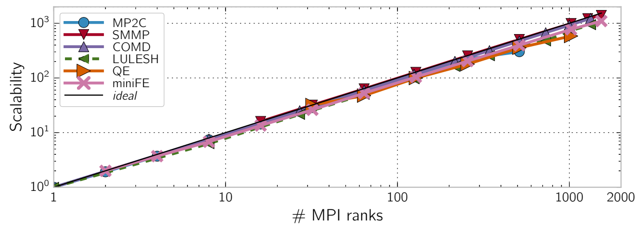
7. ASIX chip AX88179



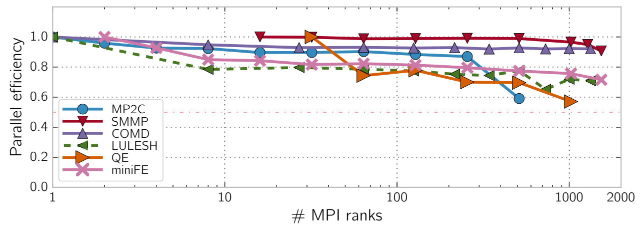
(a) Strong scaling.



(b) Strong efficiency.



(c) Weak scaling.



(d) Weak efficiency.

Figure 11: Scalability and parallel efficiency of MPI applications on the Mont-Blanc prototype. Two MPI ranks per node.

5.1. Application scalability

In Section 3.2, we show that a Mont-Blanc node is 18x slower than a MareNostrum-III node when using only the CPU cores. This means we have to be able to linearly scale a workload to 18x more compute nodes to achieve equivalent performance.

In Figure 11, we show both *strong* and *weak* scaling figures for MPI applications on the Mont-Blanc prototype. Each graph is accompanied with the corresponding parallel efficiency graph to provide more details about the application's scalability. Note that 16 Mont-Blanc nodes already span 2 EMB blades, and 32 nodes span 3 blades. Also, most applications had their baseline run with >1 node due to the 4GB/node DRAM limitation.

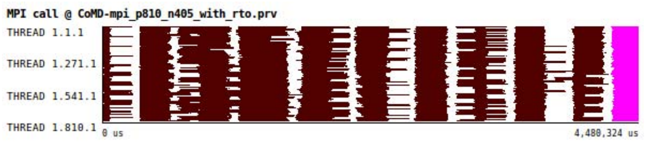
Strong scaling for SMMP quickly degrades starting at 32 nodes. Parallel efficiency drops to 50%, and performance flattens, and even degrades at 512 nodes. BQCD and QE also exhibit quick strong scaling degradation, but still run at >50% efficiency on 64 nodes. The rest of the applications scale linearly to hundreds of nodes, with 4 of them still running at >50% efficiency at the full scale of the system.

Our results show that it is reasonable to scale applications to 16 nodes to compensate for the difference with a MareNostrum III node. However, not all applications will scale further to compensate for multiple MareNostrum III nodes.

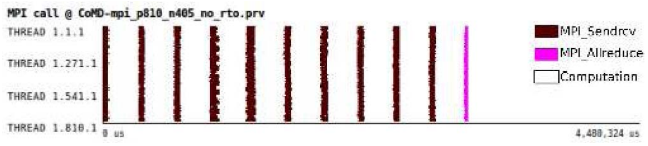
Weak scaling results are much better. Most of the applications still run at >70% efficiency at the maximum problem size. Notably, CoMD and SMMP run at >90% efficiency, but QE and MP2C degrade to 60% efficiency.

Detailed performance analysis reveals the causes for lack of scalability: besides the low bandwidth / high latency GbE network, the system suffers from lost packets in the interconnect, each incurring at least one Retransmission Time Out (RTO), and load imbalance introduced by scheduler preemptions.

5.1.1. Lost packets. In Figure 12, we show an execution profile for the real CoMD run, and a simulated run (using Dimemas [25]) eliminating network retransmissions. Both traces have the same time scale.



(a) Packet loss in place.



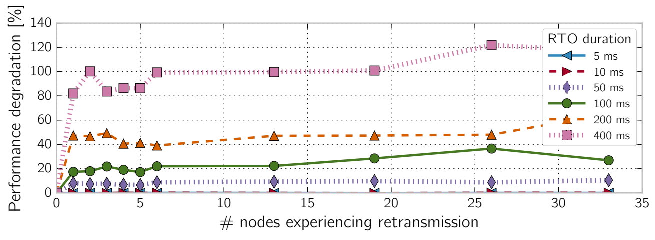
(b) No packet loss.

Figure 12: Illustration of the packet loss effect on MPI parallel applications: a) trace with, and b) without packet loss. The X axis represents time, the Y axis represents the process number.

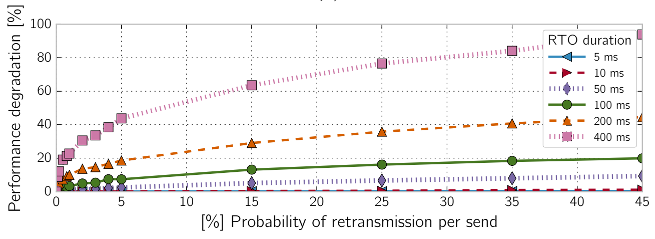
The simulated profile shows that the native execution suffered from a lot of lost packets (most application's communication phases suffer from at least 1 retransmission), reducing performance by 1.47x. This is of course application dependent, and depends on the communication patterns, message sizes, volume of communication, etc.

Further analysis of the duration of MPI send operations shows that CoMD experiences multiple retransmissions per packet, with and average MPI send duration of 158ms (compared to 50ms optimum), and often reaching 400ms.

Figure 13a shows the performance degradation as a function of how many nodes experience a retransmission



(a)



(b)

Figure 13: Performance degradation due to retransmissions: a) every message is affected for selected nodes; b) random messages are affected.

penalty on every message they send. The results show that the penalty is linear with respect to the retransmission delay. But more important, the results show that as soon as one node has to retransmit, the whole applications pays almost the full penalty.

Figure 13b shows the performance degradation as a function of how many messages need to be retransmitted (by any node). The results show that the penalty is linear with the retransmission delay and the retransmission probability. Both results combined indicate that it is important to avoid retransmissions in the whole system, or to cluster retransmissions in time, because as soon as one node has to retransmit, it does not matter if others also have to retransmit. For example, a glitch in a switch that causes all nodes connected to it to retransmit would have a similar penalty to a glitch in the NIC of one of the nodes, forcing it alone to retransmit.

To minimize the penalty of retransmissions, we reduce the RTO_{min} parameter in the TCP/IP stack from the default 200ms to 5ms (the lowest possible in our system). While the lowering of RTO_{min} parameter reduces retransmission penalties, it would be desirable implementing Retransmission Early Detection (RED) to reduce the effects of retransmissions. However, packet loss does not exclusively happen at switch buffers, but we also observed nodes can drop packets. In addition, our blade switches which forward most of the network traffic do not support Explicit Congestion Notification (ECN) markings, thus not being able to control transmission rates.

5.1.2. Pre-emptions. Figure 14 shows a histogram of the duration of computational phases in the real CoMD execution. The gradient color shows the total time spent in computation phase of a given duration (green/light is low, blue/dark is high).

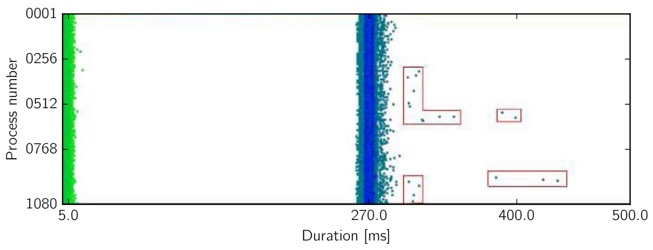


Figure 14: 2D Histogram of computational phases duration. X axis represents bins of durations, Y axis represents process number. Gradient coloring: green-blue. Coloring function: logarithmic.

The figure shows two main regions of 5ms and 270ms durations. We match the 5ms regions to the TCP/IP retransmissions (matching the 5ms RTO setting, and confirming that many processes suffer retransmissions). Then, the remaining time is spent in 270ms regions, matching the duration of one inner iteration of the application. Beyond the 270ms boundary, we identify a set of outliers taking significantly more time (marked with red polygons).

Checking the IPC of these computation phases, we confirm that the divergence in execution time is not related to load imbalance in the application. There are external factors introducing this variation. We attribute them to scheduler preemptions, and from now on treat them as OS noise in discussions to come.

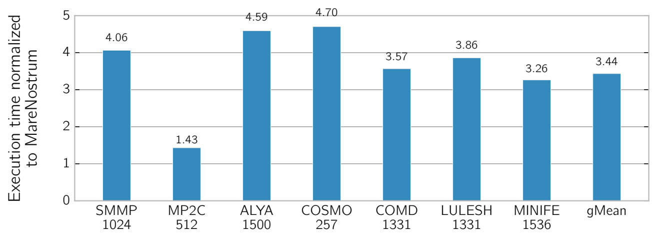
Further simulations of different noise injection frequencies and noise duration (graph not shown) indicate that the performance impact of OS noise is linear with the probability of noise being injected, and the ratio for the noise duration to the computational burst. That is, applications with short computational bursts are more prone to suffer OS noise performance degradation than applications with long computational bursts.

5.2. Comparison with traditional HPC

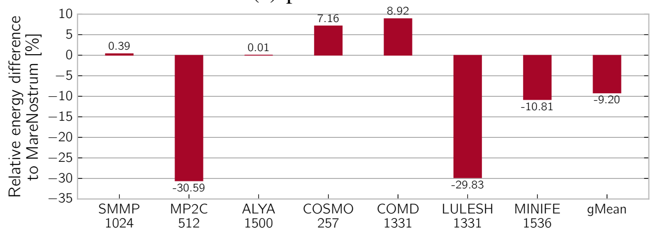
Figure 15 shows a comparison between Mont-Blanc and MareNostrun III when using the same number of MPI ranks (same number of cores). As a reference, we remind the reader of the node performance comparison of both systems in Table 2. Since applications are not completely malleable in the number of MPI ranks they can use, the reported number of cores is different for each application, ranging from 257 to 1536.

Our results show that Mont-Blanc is 3.5x slower on average (matching the Mont-Blanc benchmarks evaluation in Section 3.1), and requires 9% less energy to run the applications. However, none of the applications is optimized to use the GPU or OmpSs. Following the results in Section 3, we would expect Mont-Blanc to result in better energy efficiency once the GPU is used alongside the CPU.

Table 5 shows a comparison of the Mont-Blanc prototype and MareNostrum III when aiming to equalize their execution time. For this experiment we exercise the strong-scaling capability of applications on the Mont-Blanc proto-



(a) performance



(b) energy

Figure 15: Mont-Blanc vs MareNostrum III: a) performance and b) energy comparison for a fixed number of MPI ranks. In both cases lower is better for Mont-Blanc.

TABLE 5: Mont-Blanc vs MareNostrum III: same input, same execution time.

	CoMD		*miniFE*	
	MN[a]	MB[b]	MN	MB
# MPI ranks	64	240	64	224
Execution time [s]	70.72	68.05	71.66	72.19
Avg. power [W]	992	1083	1065	1034
Energy [Wh]	195	205	212	207
# rack units	8	7	8	7

[a] MareNostrum III
[b] Mont-Blanc

type, such that we keep input set constant and increase the number of MPI ranks to get the same execution time on MareNostrum III with 64 MPI ranks (4 nodes).

Our results show that Mont-Blanc needs 3.5–3.75 more MPI ranks to match the MareNostrum III execution time. This is consistent with the 3.5x slowdown observed for constant number of MPI ranks, and shows similar scalability on both systems. In terms of energy consumption, both systems consume approximately the same amount of energy. Regarding rack space, Mont-Blanc requires 7 rack units (1 BullX chassis, 9 blades x 15 nodes, 270 cores), while MareNostrum III requires 8 rack units (4 2U nodes).

We conclude that, when using only the CPUs, Mont-Blanc and MareNostrum III are equally energy efficient, with Mont-Blanc having slightly higher integration density.

6. Scalability Projection

The results in Section 5 show that the scalability of the Mont-Blanc prototype is affected by the choice of node interconnect technology (Ethernet via USB), and potential load imbalance (introduced by the system, or intrinsic to the application). Further, in Section 5.2, we reveal a need for good parallel scalability to compensate for lower per node performance compared against the MareNostrum III supercomputer. These issues conceal the potential of the Mont-Blanc approach at scale.

To unveil the scalability of the prototype architecture to larger systems, we employ a state-of-the-art modeling methodology [26], [27] that allows us to project the scalability of the current deployment once certain issues have been fixed.

To validate our hypothesis about factors preventing applications to scale, we simulate weak scheduling scenarios where we have removed network retransmissions, OS preemptions, and improved load balance in the application. We remove retransmissions by having the Dimemas simulator assume that messages are always delivered. We remove OS preemptions by recomputing the CPU burst durations using the cycle counter (multiplying by the cycle time). Since the cycle counter is virtualized, it does not count while the application is preempted. To simulate a better load balance, we evenly redistribute the instruction count in a computation phase across all the MPI ranks, and compute the burst duration using the average IPC for the compute phase. Finally, we also simulate an ideal network (lossless, zero latency, infinite bandwidth) to determine if a better (hardware-supported) network would improve the system.

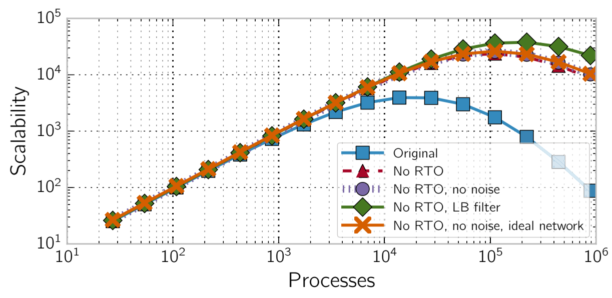
Figure 16 shows the results. The baseline setup (blue curve) shows that none of the 3 simulated applications scale beyond 100K MPI ranks with an efficiency over 50%. If we consider that the whole MareNostrum III system has 3,056 nodes (48.9K processors), and that Mont-Blanc is 3.5x slower at the same number of MPI ranks, a Mont-Blanc system with 50K nodes would be 1.75x slower than MareNostrum III, consume the same energy (but less power), and use 12.5% less space.

However, just removing the network packet loss (red line, triangle) already shows a significant improvement for CoMD, now scaling well to 30K processes and still improving performance up to 100K processes. LULESH and miniFE do not seem to be heavily affected by the lossy network.

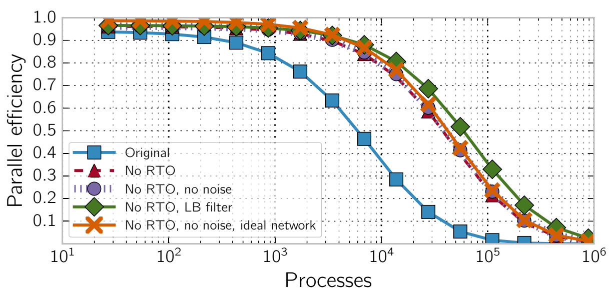
When we eliminate both the retransmissions, and the OS preemptions (purple line, circle), CoMD does not exhibit a significant improvement. It is clear that its scalability was dominated by the retransmissions. However, LULESH and miniFE show some improvement, that indicates that serialization introduced by OS noise was causing significant damage.

If we go one step further, and look at the ideal network simulations (orange line), we observe that none of the three applications is limited by network performance (after we remove the retransmissions and OS noise). At this point, we have obtained a scalability improvement of 7x for CoMD, 1.2x for LULESH, and 1.1x for miniFE, most of it due to using a lossless network.

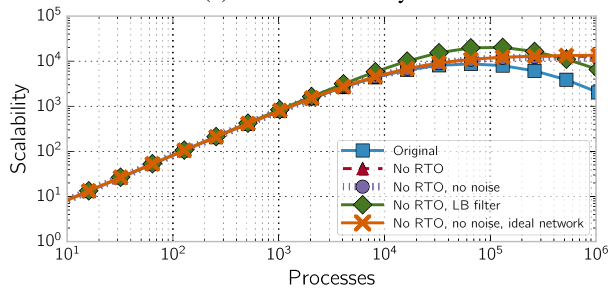
Further analysis reveals that load imbalance is the biggest issue affecting scalability of the prototype (green line, diamond). Improving load balance has a visible impact on all three applications, with LULESH and miniFE being the most affected (notably, the two apps that were insensitive to the network performance).



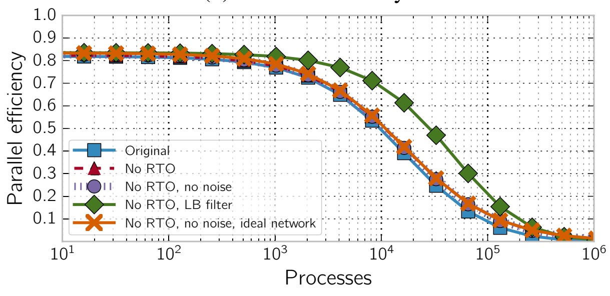
(a) CoMD scalability.



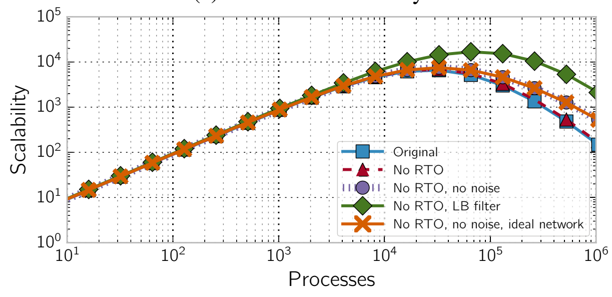
(b) CoMD efficiency.



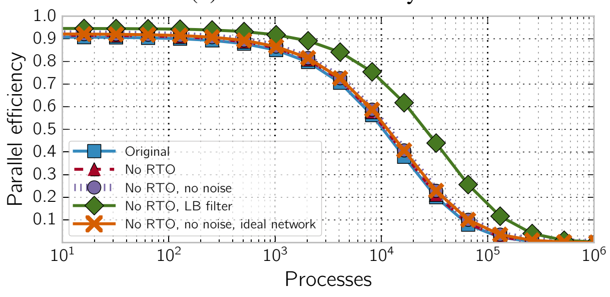
(c) LULESH scalability



(d) LULESH efficiency.



(e) miniFE scalability



(f) miniFE efficiency.

Figure 16: Measured and simulated scalability and parallel efficiency. Simulated configurations are as follows: ▲ - w/o retransmissions; ● - w/o retransmissions and preemptions; ♦ - w/o retransmissions + balanced load; ✖ - w/o retransmissions, preemptions,and ideal network parameters.

Improving the load balance in the application is beyond the capabilities of the hardware, and while it will affect Mont-Blanc systems and traditional systems like MareNostrum III in a similar way, Mont-Blanc needs to scale to a higher number of processes to compensate for the slower compute nodes, making it a most critical aspect of application development.

7. Related work

A clear example of how commodity technology can bring a breakthrough in HPC systems architecture is the ASCI Red supercomputer [28]. The first top-tier supercomputer, breaking TFLOPS barrier, built with commodity x86 processors - namely 7,246 Intel Pentium Pro processors. With ASCI Red, a new era of HPC systems built with commodity PC processors began.

The three generations of the BlueGene family of supercomputers [29], [30], [31], first introduced in 2004, exploited a new approach for energy-efficient HPC. The BlueGene processors were based on IBM cores used in the embedded market segment and extended with powerful floating-point units with SIMD capabilities. However, system software and interconnection network were still HPC-specific.

Further prototype clusters built with low-power commodity processors include GreenDestiny [32] and MegaProto [33] built with Transmeta Crusoe, FAWN[34] built with Intel Atom, and the Apple TV cluster [35], the first small-scale 4-node cluster built with ARM Cortex-A8 processors .

The Tibidabo cluster prototype, deployed in 2011, was the first large-scale HPC cluster based on mobile processors [36], featuring 256 nodes of dual-core ARM Cortex-A9 processors. It was the first ARM-based cluster with a full HPC software stack including cluster management, job scheduler, scientific libraries and HPC performance analysis tools. The ARM Cortex-A9 was the first mobile processor with a floating-point unit allowing for single-cycle double-precision operations. Tibidabo also demonstrated a case for scale-out parallel processing with real scientific applications on mobile processors.

There have been multiple SoCs and commercial solutions using embedded processors and targeting server market: the Calxeda EnergyCore ECX-1000, AMD Opteron A1100 are ARM based, while the AMD SeaMicro SM10000-64 and the Quanta Computer S900-X31A are based on the Intel Atom. All extend the embedded multicore with high bandwidth networks (10GbE) and ECC memory protection.

Other companies have developed custom processors based on the ARMv8 architecture. Applied Micro (APM) X-Gene [37] is a server-class SoC with eight 64-bit ARMv8 cores and four 10 GbE links. Cavium, with large experience in networking processors, designed ThunderX [38], another server-class SoC with 48 ARMv8 cores and multiple 10/40GbE interfaces. Qualcomm and Phytium also announced ARMv8 server SoCs with 24 [39] and 64 [40] cores, respectively. Some successful deployments of some of these SoCs are already in place. CERN has published a comparison of APM X-Gene compared to Intel Xeon and IBM Power8 chips [41]. PayPal has deployed HP Moonshot servers with APM X-Gene processors claiming half the price, one seventh of the power consumption and 10x more nodes per rack compared to their traditional data center infrastructure [42].

These efforts, however, target the server market and there are still no large demonstrators of such mobile-technology-based processors for HPC. The Mont-Blanc prototype is thus the first demonstrator of an HPC cluster with full HPC software stack running real scientific applications, commodity networking, and standard system integration. Our experiments demonstrate the feasibility of the proposed alternative approach, assess system software maturity and project its scalability at larger scale.

At the ISC'16 conference, Fujitsu announced their plans to produce a processor for the Post-K Exascale Supercomputer based on their own microarchitecture implementation of ARMv8 [43]. This supports our approach of using commodity technology deployed in the mobile and embedded markets, in this case the ARMv8 architecture, for HPC.

8. Conclusions

In this paper, we have described the architecture of the Mont-Blanc prototype in detail, and compared it to a production supercomputer. Our results show that Mont-Blanc is 4x slower than MareNostrum III on the same number of MPI processes, however applications can weakly scale to 4x more nodes to compensate for that, and still run in approximately the same time and same energy. Since applications must scale to a higher number of nodes, load balancing is a critical design issue for the applications. Load imbalance, including that introduced by retransmissions, could be improved using MPI+OpenMPI and a runtime dynamic load balancing method [44], which detects idle cores finishing early and assigns them a part of the work of the core which suffered retransmission timeout.

Cost savings due to use of commodity embedded SoCs in Mont-Blanc is impossible to evaluate at this point. The Mont-Blanc prototype is dominated by Non-recurring engineering (NRE) costs, while the cost of MareNostrum III is the result of a negotiation and a competitive bid.

However, we do not necessarily advocate for using off-the-shelf mobile processors like the Samsung Exynos 5250, just like we do not use off-the-shelf desktop Intel Core i7 cores for MareNostrum III. We advocate for building workload-specific SoCs based on the IP developed for the embedded and mobile segments, adding the missing features required by HPC, such as a lossless network (as indicated by our results in Section 6), ECC memory protection, and the set of accelerators that the workload will exploit. Just like the Intel Xeon used in MareNostrum III builds on the desktop Intel Core i7 Sandy Bridge processor.

From the time when Mont-Blanc specifications were frozen, there have been many developments in the embedded computing space: increased multicore counts (4 and 8 cores per SoC), 64-bit ARM processors (Cortex-A72 and A57), CUDA capable embedded GPUs (NVIDIA Tegra K1 and X1 SoCs), and on-chip PCIe controllers are all available.

Our projections simulating CoMD, LULESH, and miniFE on an upgrade of the Samsung Exynos 5250 dual-core Cortex-A15 @ 1.7 GHz to the NVIDIA Tegra X1 quad-core Cortex-A57 @ 1.9 GHz show a 1.6-1.7$\times$ performance improvement while still relying only on the CPUs.

Based on our analysis in this paper, a next-generation Mont-Blanc system should have a lossless interconnection network, and a higher per-node core count to better amortize shared infrastructure costs such as cooling and power supply. Then, the burden falls on the applications having to fully utilize the SoC resources, such as the embedded GPU[8], and focus on load balancing to scale to a higher number of lower-performance nodes. Under these conditions, Mont-Blanc type systems would offer equivalent performance to contemporary systems, while saving 40% energy, and achieving higher integration density.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments. This project and the research leading to these results was supported by the Mont-Blanc project (European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n^o 288777), the Spanish Ministry of Science and Technology through *Computacion de Altas Prestaciones (CICYT) VI* (TIN2012-34557), and the Spanish Government through *Programa Severo Ochoa* (SEV-2011-0067).

References

[1] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3. IEEE Computer Society, 2012, pp. 37–48.

8. The use of OpenCL for HPC code acceleration has not ramped up since 2012 when the ARM Mali GPU was selected. CUDA and OpenMP with SIMD annotations seem to be the preferred way to use HPC accelerators today.

[2] A. Duran, E. Ayguadé, R. M. Badia, J. Labarta, L. Martinell *et al.*, "Ompss: a proposal for programming heterogeneous multi-core architectures," *Parallel Processing Letters*, vol. 21, no. 02, pp. 173–193, 2011.

[3] "The MQTT Protocol," http://mqtt.org, 2014.

[4] "ARM Connected Community Forums," https://community.arm.com/message/18218, 4 2014.

[5] Barcelona Supercomputing Center, "MareNostrum III (2013) System Architecture," https://www.bsc.es/marenostrum-support-services/mn3.

[6] N. Rajovic, A. Rico, J. Vipond, I. Gelado, N. Puzovic, and A. Ramirez, "Experiences with mobile processors for energy efficient HPC," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 464–468.

[7] Ming Lei, "USBNET: increase max rx/tx qlen for improving USB3 throuput," https://github.com/torvalds/linux/commit/452c447a497dce3c9faeb9ac7f2e1ff39232876b, 2013.

[8] S. N. Kandadio and X. He, "Performance of HPC Applications over Infiniband, 10 Gb and 1 Gb Ethernet," http://www.chelsio.com/assetlibrary/whitepapers/HPC-APPS-PERF-IBM.pdf.

[9] B. Goglin, "Design and implementation of Open-MX: High-performance message passing over generic Ethernet hardware," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–7.

[10] "Coral collaboration benchmark codes," https://asc.llnl.gov/CORAL-benchmarks/, 2013.

[11] "The BigDFT Scientific Application," http://bigdft.org/, 2015.

[12] L. Genovese, B. Videau, M. Ospici, T. Deutsch, S. Goedecker, and J.-F. Méhaut, "Daubechies Wavelets for High Performance Electronic Structure Calculations: the BigDFT Project." in *Compte-Rendu de l'Académie des Sciences, Calcul Intensif.* Académie des Sciences, 2010.

[13] Y. Nakamura and H. Stüben, "BQCD-Berlin quantum chromodynamics program," *arXiv preprint arXiv:1011.0199*, 2010.

[14] G. Sutmann, L. Westphal, and M. Bolten, "Particle based simulations of complex systems with mp2c : Hydrodynamics and electrostatics," *AIP Conference Proceedings*, vol. 1281, no. 1, pp. 1768–1772, 2010. [Online]. Available: http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.3498216

[15] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car *et al.*, "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, no. 39, p. 395502, 2009. [Online]. Available: http://stacks.iop.org/0953-8984/21/i=39/a=395502

[16] F. Eisenmenger, U. H. E. Hansmann, S. Hayryan, and C.-K. Hu, "[SMMP] A modern package for simulation of proteins," *Computer Physics Communications*, vol. 138, no. 2, pp. 192–212, 2001.

[17] ——, "An enhanced version of SMMP—open-source software package for simulation of proteins," *Computer Physics Communications*, vol. 174, no. 5, pp. 422–429, 2006.

[18] J. H. Meinke, S. Mohanty, F. Eisenmenger, and U. H. E. Hansmann, "[SMMP] v. 3.0—Simulating proteins and protein interactions in Python and Fortran," *Computer Physics Communications*, vol. 178, no. 6, pp. 459–470, 2008.

[19] M. Vazquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra *et al.*, "Alya: towards exascale for engineering simulation codes," *arXiv preprint arXiv:1404.4881*, 2014.

[20] M. Vázquez, R. Arís, J. Aguado-Sierra, G. Houzeaux, A. Santiago *et al.*, *Selected Topics of Computational and Experimental Fluid Mechanics*. Cham: Springer International Publishing, 2015, ch. Alya Red CCM: HPC-Based Cardiac Computational Modelling, pp. 189–207.

[21] ExMatEx, "Comd proxy application."

[22] I. Karlin, J. Keasler, and R. Neely, "Lulesh 2.0 updates and changes," Tech. Rep. LLNL-TR-641973, August 2013.

[23] I. Karlin, A. Bhatele, J. Keasler, B. L. Chamberlain, J. Cohen *et al.*, "Exploring traditional and emerging parallel programming models using a proxy application," in *27th IEEE International Parallel & Distributed Processing Symposium (IEEE IPDPS 2013)*, Boston, USA, May 2013.

[24] M. A. Heroux, D. W. Doerfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards *et al.*, "Improving performance via mini-applications," *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, vol. 3, 2009.

[25] R. M. Badia, J. Labarta, J. Gimenez, and F. Escale, "Dimemas: Predicting mpi applications behavior in grid environments," in *Workshop on Grid Applications and Programming Tools (GGF8)*, vol. 86, 2003, pp. 52–62.

[26] M. Casas, R. M. Badia, and J. Labarta, "Automatic analysis of speedup of MPI applications," in *Proceedings of the 22nd Annual International Conference on Supercomputing, ICS 2008*, 2008, pp. 349–358.

[27] C. Rosas, J. Giménez, and J. Labarta, "Scalability prediction for fundamental performance factors," *Supercomputing frontiers and innovations*, vol. 1, no. 2, 2014.

[28] T. Mattson and G. Henry, "An Overview of the Intel TFLOPS Supercomputer," *Intel Technology Journal*, vol. 2, no. 1, 1998.

[29] N. R. Adiga, G. Almási, G. S. Almasi, Y. Aridor, R. Barik *et al.*, "An overview of the BlueGene/L supercomputer," in *ACM/IEEE 2002 Conference on Supercomputing*. IEEE Computer Society, 2002.

[30] S. Alam, R. Barrett, M. Bast, M. R. Fahey, J. Kuehn *et al.*, "Early evaluation of IBM BlueGene/P," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, ser. SC '08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 23:1–23:12.

[31] IBM Systems and Technology, "IBM System Blue Gene/Q Data Sheet," November 2011.

[32] M. Warren, E. Weigle, and W. Feng, "High-density computing: A 240-processor Beowulf in one cubic meter," in *Supercomputing, ACM/IEEE 2002 Conference*. IEEE, 2002, pp. 61–61.

[33] H. Nakashima, H. Nakamura, M. Sato, T. Boku, S. Matsuoka *et al.*, "Megaproto: 1 TFlops/10kW rack is feasible even with only commodity technology," in *Proceedings of the ACM/IEEE SC 2005 Conference on Supercomputing*. IEEE, 2005.

[34] V. Vasudevan, D. Andersen, M. Kaminsky, L. Tan, J. Franklin, and I. Moraru, "Energy-efficient cluster computing with fawn: Workloads and implications," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. ACM, 2010, pp. 195–204.

[35] K. Fürlinger, C. Klausecker, and D. Kranzlmüller, "Towards energy efficient parallel computing on consumer electronic devices," in *Information and Communication on Technology for the Fight against Global Warming*. Springer, 2011, pp. 1–9.

[36] N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramirez, "Tibidabo: Making the case for an ARM-based HPC system," *Future Generation Computer Systems*, vol. 36, pp. 322–334, 2014.

[37] Applied Micro, "APM "X-Gene" Launch Press Briefing," https://web.archive.org/web/20120813151248/http://www.apm.com/global/x-gene/docs/X-GeneOverview.pdf, 2012.

[38] Cavium, "ThunderX^{TM}," https://web.archive.org/web/20160310114848/http://www.cavium.com/pdfFiles/ThunderX_PB_p12_Rev1.pdf, 2013.

[39] PCWorld, "Qualcomm enters server CPU market with 24-core ARM chip," http://www.pcworld.com/article/2990868/qualcomm-enters-server-cpu-market-with-24-core-arm-chip.html, 2015.

[40] Charles Zhang, Phytium Technology Co., Ltd, "Mars: A 64-core ARMv8 Processor," http://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.24-Monday-Epub/HC27.24.30-HP-Cloud-Comm-Epub/HC27.24.321-64core-Zhang-phytium-v1.0.pdf, 2015.

[41] D. Abdurachmanov, B. Bockelman, P. Elmer, G. Eulisse, R. Knight, and S. Muzaffar, "Heterogeneous high throughput scientific computing with apm x-gene and intel xeon phi," in *Journal of Physics: Conference Series*, vol. 608, no. 1. IOP Publishing, 2015, p. 012033.

[42] Data Center Knowledge, "PayPal Deploys ARM Servers in Data Centers," http://www.datacenterknowledge.com/archives/2015/04/29/paypal-deploys-arm-servers-in-data-centers/, 2015.

[43] D. Cepulis, "ISC16 Recap - Fujitsu Takes the Stage," https://community.arm.com/groups/processors/blog/2016/06/27/isc16-recap-fujitsu-takes-the-stage.

[44] M. Garcia, J. Corbalan, and J. Labarta, "Lewi: A runtime balancing algorithm for nested parallelism," in *2009 International Conference on Parallel Processing*. IEEE, 2009, pp. 526–533.