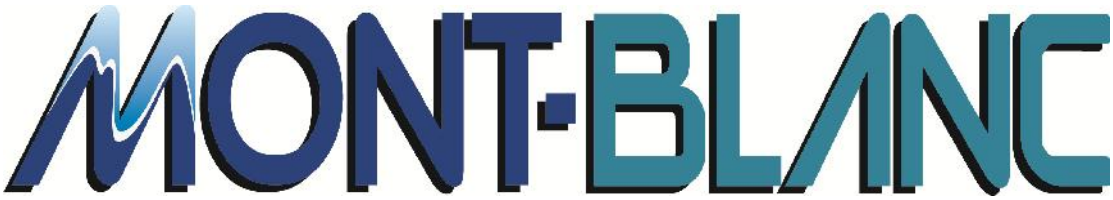

D3.5 Benchmarking progress report and submission of SPEC CPU2006 results Version 1.0

Document Information

Contract Number	288777
Project Website	www.montblanc-project.eu
Contractual Deadline	M36
Dissemination Level	PU
Nature	Report
Author	Hervé GLOAGUEN (BULL), Chris ADENIYI-JONES (ARM), Xavier MARTORELL (BSC)
Contributors	Marc SIMON (BULL)
Reviewer	Alejandro Rico (BSC)
Keywords	Benchmarks, performance evaluation, OmpSs

Notices:
The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 288777.

Change Log

Version	Description of Change
v0.1	Report on OmpSs benchmarking included
v0.2	Report on Standard basic benchmarks included
v1.0	Initial Draft released to the European Commission

Table of Contents

Executive Summary

This deliverable shows the evaluation of the Mont-Blanc node using two sets of benchmarks: Standard and Mont-Blanc benchmarks. The first set consists of linear algebra benchmarks Dgemm, Sgemm, and Linpack, Stream and the Standard SPEC CPU2000. They show the raw performance of the ARM cores, and allow a comparison with the performance of the Arndale board, which results in being very similar. The second set consists of 2D Convolution, 3D Stencil, Atomic Monte-Carlo Dynamics, Sparse Matrix-Vector Product, N-body, Dense Matrix Multiplication, Histogram, Reduction, Vector Operation and Merge-Sort. They have been developed in the project as plain serial version, and parallel version with the OpenMP, OpenCL and OmpSs programming models. For the ones that have more regular memory accesses, the performance of the MALI GPU is larger than that of the to ARM Cortex-A15 cores. For the algorithms that are more irregular regarding memory access, the performance of the ARM cores and the MALI GPU is similar.

N Benchmark descriptions

During this period we have ported a set of benchmarks to run onto the Mont-Blanc prototype. This section shows the benchmark descriptions and their evaluation.

1.1 Standard basic benchmarks

This section lists the benchmarks that have been executed on the platform to verify the basic performance of the Mont-Blanc prototype. This ensures at least that there is no major performance issue. It mainly covers the CPU raw performance and the main memory bandwidth. Dgemm (double precision matrix multiply), Sgemm (single precision matrix multiply), Linpack and Stream are part of the HPC Challenge Benchmark suite [1], and are used for comparing the performance of HPC systems. We have also ported the SPEC CPU2000 benchmarks [2].

- Dgemm

The matrix-matrix multiplication algorithm on double precision floating point numbers. It measures the number of floating point operations per second (GFlops/s).

- Sgemm

This is the single precision variant of the Dgemm benchmark. It also measures the number of floating point operations per second (GFlops/s).

- Linpack

This is a benchmark introduced by Jack Dongarra in 1979 that measure the performance of a system to solve a dense NxN system of linear equations. It is used to compare HPC systems for the top500 ranking [3].

- Stream

This is a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.

- Spec

This is a test suite developed and licensed by SPEC [2]. There are several revisions of the test suite, the latest one is SPEC CPU2006. A previous version is the SPEC2000. It is designed to provide performance measurements that can be used to compare compute-intensive workloads on different computer systems.

1.2 Mont-Blanc benchmarks

We have ported the Mont-Blanc benchmarks to OmpSs [4], in order to evaluate them. This is the description of the Mont-Blanc benchmarks:

- 2D Convolution

It implements a two-dimensional stencil computation with fixed width and height distances, on a 4096x2048 image. It uses 4 values on each side of the central number to be computed, and 8 from top and bottom. The benchmarks are executed for 500 iterations. It reports Mpixels per second.

- 3D Stencil

It implements a three-dimensional stencil computation, where each element of the resulting matrix is computed based on the value of the element itself, and its 6 neighbors in the 3 dimensions. The benchmark is executed for 100 iterations. It reports the number of elements computed per second.

- Atomic Monte-Carlo Dynamics

It implements the Monte-Carlo computation on a set of atoms, with the goal of obtaining an atom configuration taking less energy. It has been implemented using both float and double precision numbers, but the versions with double precision numbers fails to compile the OpenCL kernel. It computes 100 iterations of the method, and it reports the average execution time in seconds taken by those 100 iterations.

- Sparse Matrix – Vector multiplication

It implements the sparse matrix-vector multiplication. It is executed with the MatrixMarket cant.mtx, a matrix of size 62451x62451, with 2034917 non-zeros. It reports the Gflops obtained.

- N-body

It implements the computation of the gravitational forces among a set of 4096 bodies, over 100 iterations. It reports the Gflops obtained.

- Dense Matrix Multiplication

It implements the matrix product of two matrices to give a third one (C += A*B). For this benchmark, we have implemented the OmpSs@OpenCL versions in such a way that tasks are executed on both the CPU and the GPU.

- Histogram

It implements the computation of a histogram, classifying 16M elements on 256 bins. It reports bandwidth of data processed in MB/s.

- Reduction

It implements the reduction operation of the elements of a vector, into a number. The vector contains 32M elements. It reports bandwidth of data processed in MB/s.

- Vector operation

It implements the vector addition (C = A + B) for vectors of 16M elements. It reports bandwidth of data processed in MB/s.

- Merge-sort

It implements the sorting of a vector of floating point numbers. Vector size is set to 8 million elements. It reports the number of elements sorted per second.

0 Evaluation of the benchmark

2.1 Standard basic benchmarks

The measurement is done on a single node, activating only the ARM cores (not the MALI GPU), except for some measurements on Sgemm. The results are compared with the Arndale board which is the development kit for the Exynos 5250 at 1.7GHz. The purpose is to check that the performance of the Mont-Blanc node is at least as good as on the Arndale board.

The benchmarks are executed with the following sizes:

Problem sizes	*Mont-Blanc*	*Arndale*
Matrix size for dgemm	12880 x 12880 (k = 320)	7672 x 7672 (k = 320)
Matrix size for sgemm:	18011 x 18011 (k = 640)	10611 x 10611 (k = 640)
Linpack	N=19438, NB=320	N=10595, NB=320
Stream vector size	154 MB	156 MB

- HPCC benchmarks (ARM core only)

The following table shows the results for both the Mont-Blanc node and the Arndale board on the HPCC benchmarks with the libatlas library.
Note that specific cooling has been done on the Arndale board otherwise it shuts down due to overheating. The Mont-Blanc node used its standard cooling system.

Benchmark	*Mont-Blanc node*	*efficiency*	*Arndale board*
dgemm	5.99 GFlops/s	88%	5.90 GFlops/s
sgemm	6.35 GFlops/s	93%	6.33 GFlops/s
linpack	5.54 GFlops/s	81%	5.26 GFlops/s
Stream read	3.58 GB/s		3.54 GB/s
Stream write	5.52 GB/s		5.50 GB/s
Stream triad	7.59 GB/s		7.22 GB/s

Memory latency	112 ns		114 ns

- Sgemm (MALI)

We tried to run the basic Sgemm program which is provided with the Mali SDK v1.1.0. The result is shown below, with several matrix sizes N.

N	512	1024	2048	4096
GFlops/s	1.79	1.88	1.77	0.37

The results show low performance, but the program is unlikely optimized, if we compare these results with the results obtained in Section 2.2 from the Dense Matrix Multiplication algorithm, which are from 5.16 to 6.71 Gflops/s out of the OpenCL, and OmpSs@OpenCL versions.

- Spec

Due to the size of the disk (micro SD), we are unable to run SPEC CPU2006 on the Mont-Blanc node. So the measurement has been done on SPEC CPU2000.

The results are provided below.

SpecInt

#	test	#cores	sec.	ratio
164.	gzip	2	288	11.3
175.	vpr	2	255	12.8
176.	gcc	2	103	24.7
181.	mcf	2	309	13.5
186.	crafty	2	128	18.1
197.	parser	2	329	12.7
252.	eon	2	104	28.9
253.	perlbmk	2	183	22.8
254.	gap	2	100	25.4
255.	vortex	2	220	20.0
256.	bzip2	2	215	16.2
300.	twolf	2	555	12.5

The final result is the geometric average of the basic ratio:
Est.SPECint_rate_base 17.4

Looking at the results published in the SPEC website [5], these results are equivalent to the performance obtained by 1 core of an Intel Xeon @3.2 Ghz, 2MB L3 cache, on a 2004 Dell PowerEdge 2650 (SPECint_rate_base 16.0).

SpecFP

#	test	#cores	sec.	ratio
168.	wupwise	2	149	24.9
171.	swim	2	409	17.6
172.	mgrid	2	419	9.98
173.	applu	2	288	16.9
177.	mesa	2	117	27.7
178.	galgel	2	209	32.1
179.	art	2	317	19.0
183.	equake	2	195	15.5
187.	facerec	2	159	27.7
188.	ammp	2	435	11.7
189.	lucas	2	161	28.9
191.	fma3d	2	239	20.4
200.	sixtrack	2	269	9.47
301.	apsi	2	339	17.8

The final result is the geometric average of the basic ratio:
Est.SPECfp_rate_base 18.7

Looking at the results published in the SPEC website [5], these results are a little less than those obtained by an Intel Xeon @3.8 Ghz on a 2005 Dell Precision Workstation (SPECfp_rate_base 21.8).

2.2 Mont-Blanc benchmarks

Along the evaluation, for each benchmark we show the performance obtained from the sequential version running on one of the Cortex-A15 of the Exynos chip, the performance of the OpenCL version, the performance of the OmpSs@OpenCL version using data transfers, and the performance of the OmpSs@OpenCL version when allocating the data in memory mapped on the OpenCL device.

Figure 1 shows the results obtained from the 2D Convolution benchmark. It shows that the three versions, OpenCL, OmpSs@OpenCL using memory transfers and OmpSs@OpenCL using mapped memory behave similarly in both float and double precisions. In this application, which accesses data in a regular fashion, the performance obtained from the MALI GPU is larger than the performance obtained from the serial execution on a single Cortex-A15 ARM core, showing the benefits of using the GPU in both single and double precision.

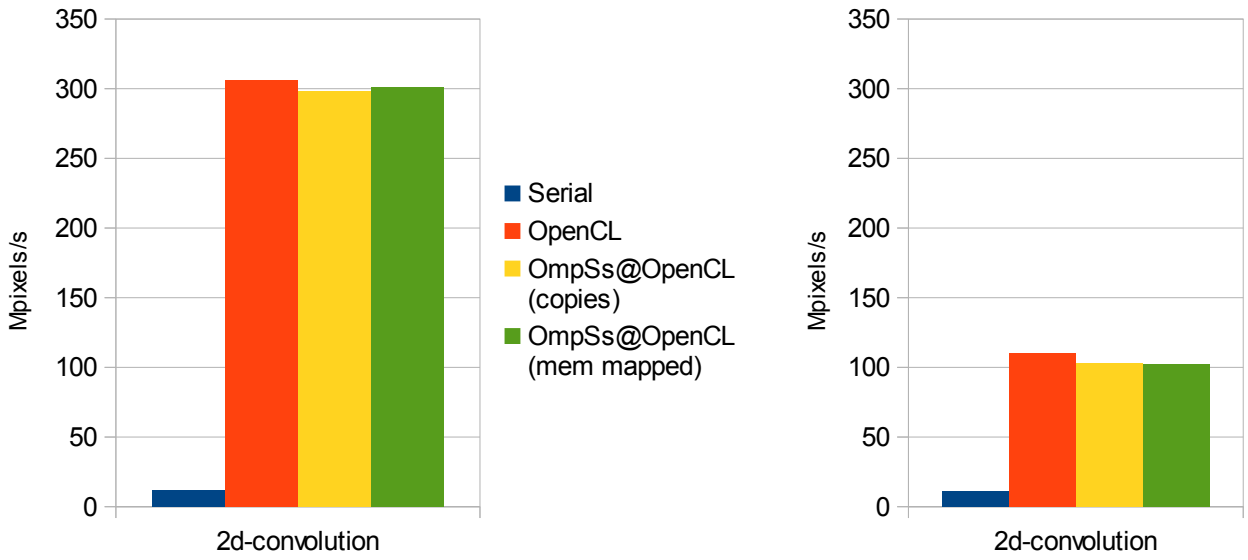

Figure 1: Performance obtained in 2D Convolution - MPixels/s in float (left) and double precision (right). Higher is better.

In addition, there is no difference in performance comparing the OmpSs@OpenCL versions with memory transfers and memory mapped memory because the application works most of the time with the data in the GPU memory, performing the transfers only at the beginning and the end. This was the transfers have no impact on the overall performance obtained.

Figure 2 shows the results of the 3D Stencil benchmark. As it is the case of the 2D Convolution benchmark, the results obtained from the 3D Stencil benchmark show that the performance obtained with the OmpSs versions is comparable to the one obtained from the OpenCL implementation. Also, as it is the case with 2D Convolution, the data transfers are performed only once at the beginning to move data to the GPU memory, and once at the end to move the results back to the host. This is the reason why the performance of the OmpSs@OpenCL versions is the same.

Figure 3 presents the performance obtained from the Atomic Monte-Carlo Dynamics benchmark on single precision floating point values, reported in seconds. It shows that the OmpSs implementations are equivalent in performance with the OpenCL version, and they also obtain better performance running on the GPU than the serial version running on the Cortex-A15.

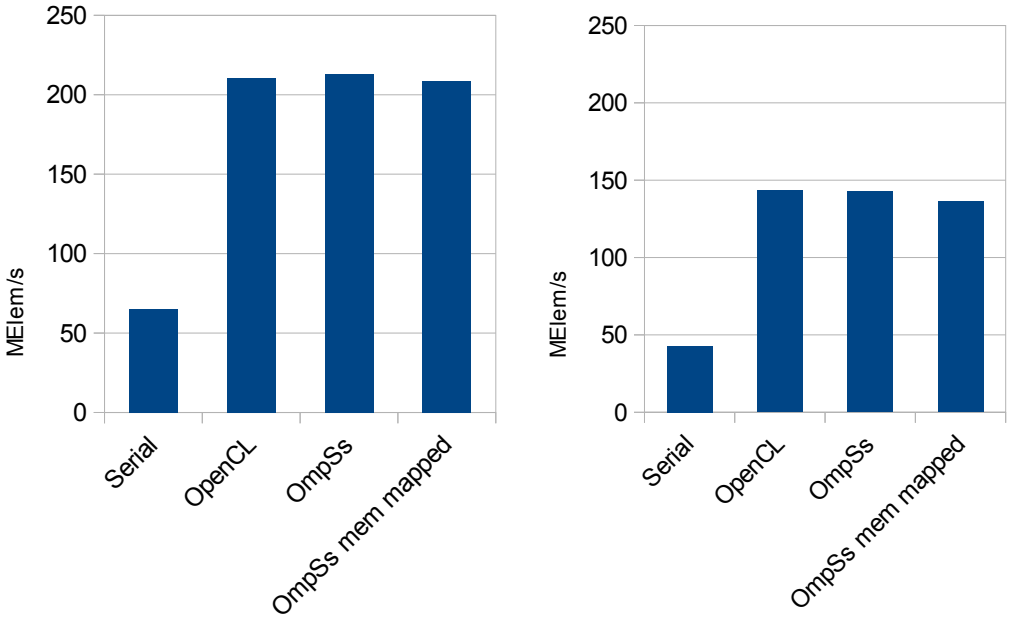

Figure 2: Performance obtained in 3D Stencil - MElem/s in float (left) and double precision (right). Higher is better.

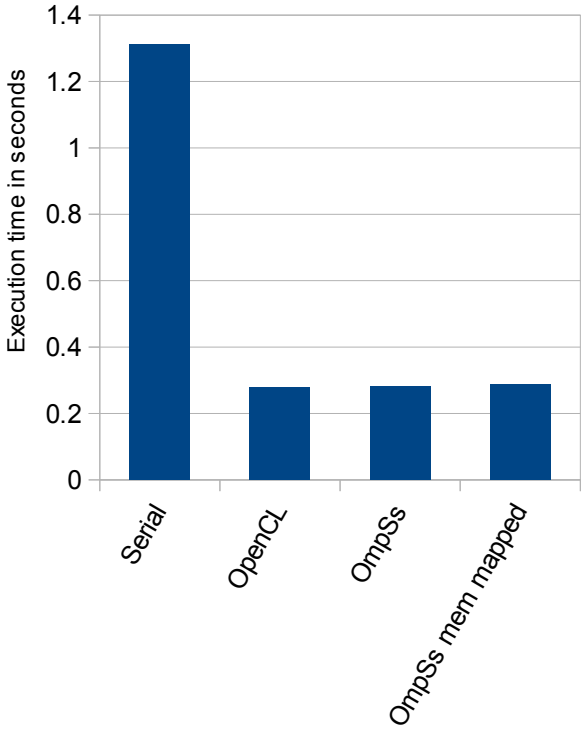

Figure 3: Performance obtained in Atomic Monte-Carlo Dynamics – Execution time in seconds on single precision data values. Lower is better.

Figure 4 and Figure 5 report the results obtained in Sparse Matrix-Vector Product, n-body and Dense Matrix multiplication, in Gflops. Figure 4 shows the single precision version of the benchmarks, and Figure 5 shows the double precision version.

For the Sparse Matrix-Vector product, the performance benefit of using the GPU in single precision is low. Instead, for the double precision version, the OpenCL implementation gets substantially better performance. The fact that the performance increase with respect the serial version is not so noticeable is due to the irregular nature of the algorithm. In both cases, the OmpSs@OpenCL implementations are comparable to the OpenCL one.

For the n-body benchmark, instead, the OpenCL implementations get higher performance that the version running on the Cortex-A15 core. For this benchmark, the serial version uses NEON vector instructions and the OpenCL and OmpSs@OpenCL versions use vector types. Overall, the performance obtained from the OpenCL versions is much higher than that obtained from

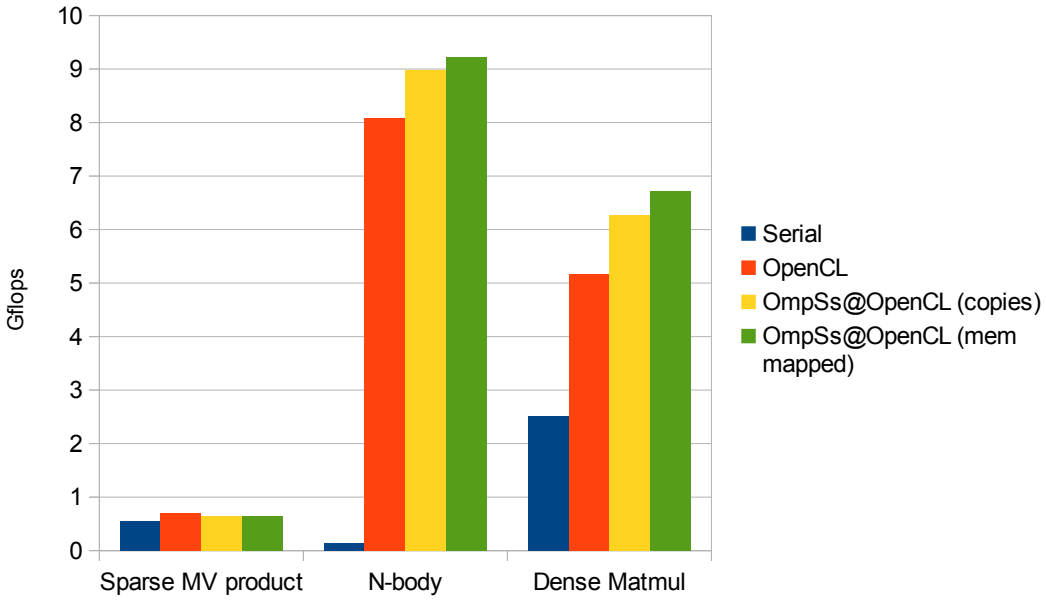

Figure 4: Performance obtained in Sparse MV product, N-body and Dense Matrix Multiplication – Gflops in the single precision (float) version. Higher is better.

the serial version. Further evaluation will be done to determine the reason for this performance benefit.

For the Dense Matrix Multiplication benchmark, we observe that the OpenCL implementations also get better performance than the version run in the Cortex-A15. Comparing the OpenCL version with the OmpSs versions, we observe that the latter outperform OpenCL with the technique of exploiting tasks both on the GPU and the CPU. The CPU is effectively helping in achieving better performance in both single and double precision versions.

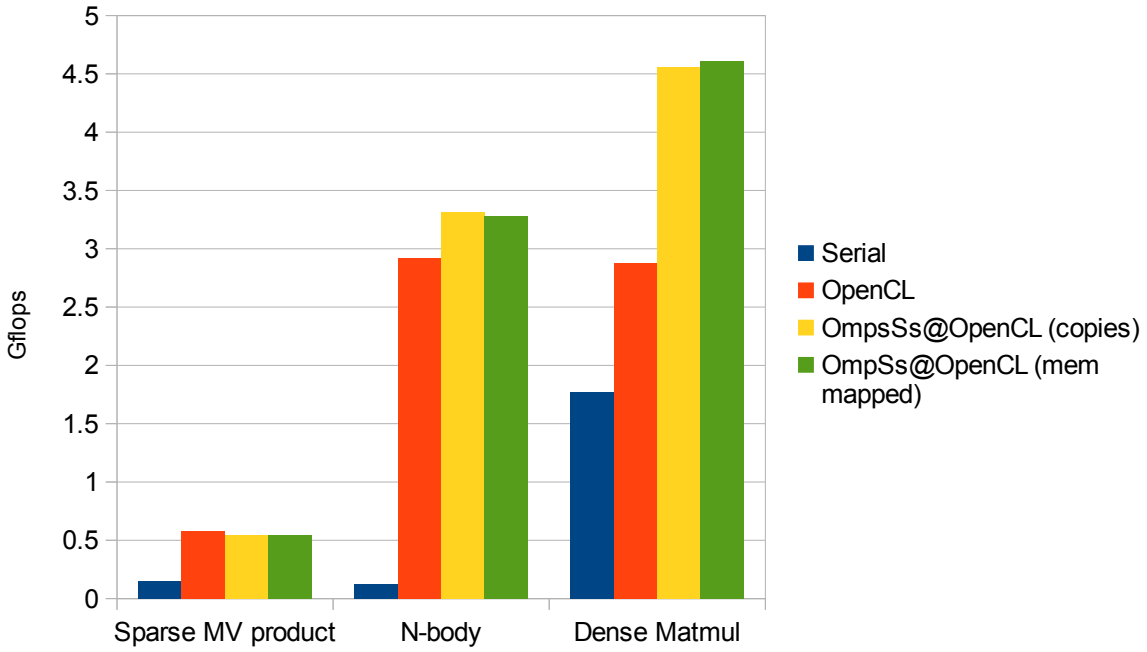

Figure 5: Performance obtained in Sparse MV product, N-body and Dense Matrix Multiplication – Gflops in the double precision version. Higher is better.

Figure 6 and Figure 7 show the performance obtained in the benchmarks Histogram, Reduction and Vector operation. These benchmarks report the bandwidth achieved in MB/s accessing the full set of data to perform the computation. Figure 6 shows the results obtained by the single precision versions, and Figure 7 shows the results obtained by the double precision version.

Both for Histogram and Reduction, we observe that the higher data access rate compared to computation is noticeable in the lower performance that the OmpSs@OpenCL version achieves compared to both OpenCL and OmpSs@OpenCL with mapped memory. This latter version avoids copying the data back to the host after each iteration. This is necessary in Histogram to re-initialize the data to be used in the next iteration. In the case of Reduction, it has two parallel tasks, one of them implemented in the SMP host. The data transfers this parallel task generates for each iteration makes the OmpSs@OpenCL with data copies version to have lower performance than the memory mapped version.

Vector operation, is another benchmark that shows regular memory accesses, and it is again a benchmark that shows the benefit of using the GPU to get better performance when compared to the use of the Cortex-A15 cores. In this benchmark, it also happens that the OpenCL, OmpSs@OpenCL and OmpSs@OpenCL with mapped memory perform very similarly because the data transfers can be done at the beginning and the end only.

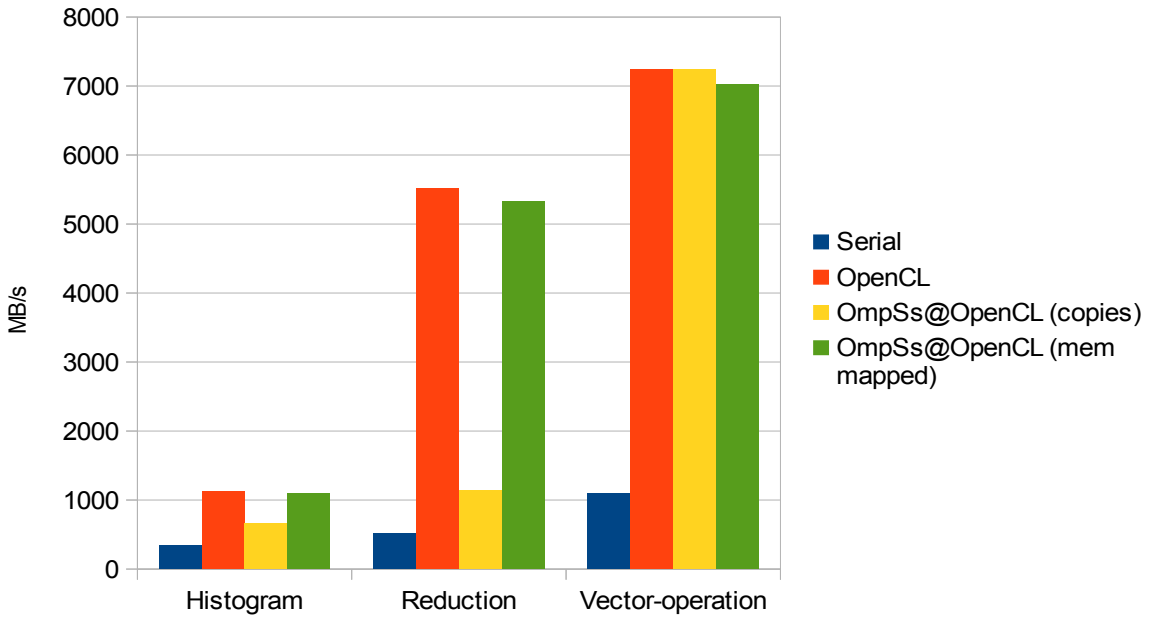

Figure 6: Performance obtained in Histogram, Reduction, and Vector operation – Bandwidth in MB/s in the single precision (float) version. Higher is better.

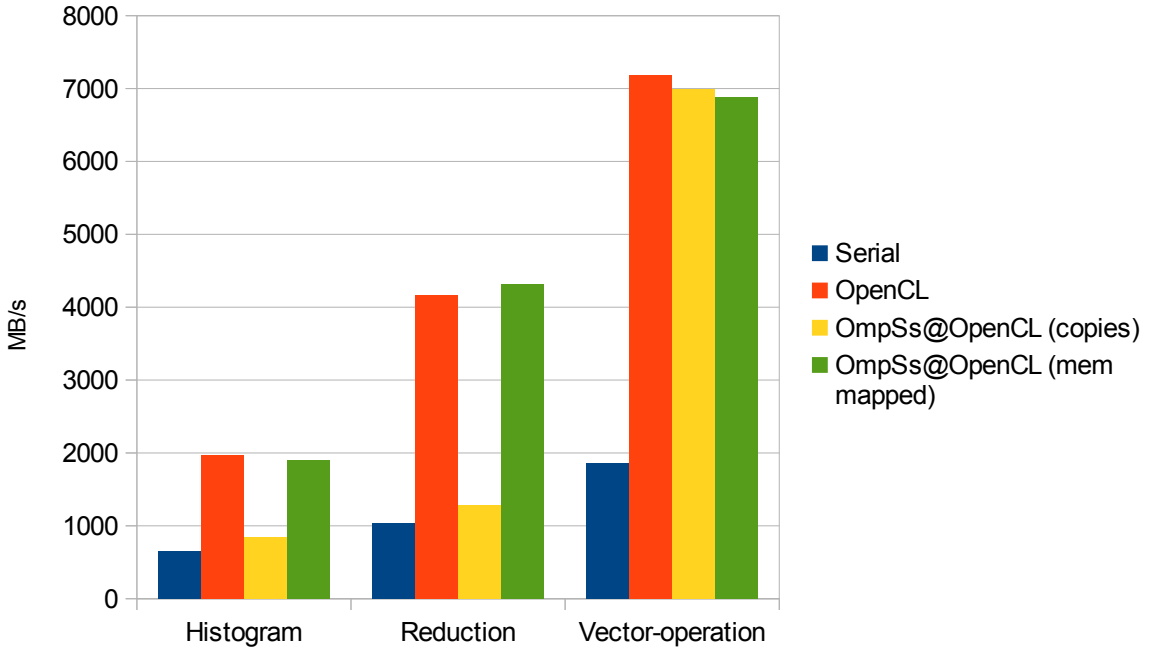

Figure 7: Performance obtained in Histogram, Reduction, and Vector-operation – Bandwidth in MB/s in the double precision version. Higher is better.

Figure 8 shows the performance obtained in the Merge-sort benchmark. In this benchmark, the performance of the OpenCL versions is not better than that achieved in the Cortex-A15 core. Low data locality causes this low performance in the GPU. OmpSs@OpenCL versions are not capable of improving the OpenCL results.

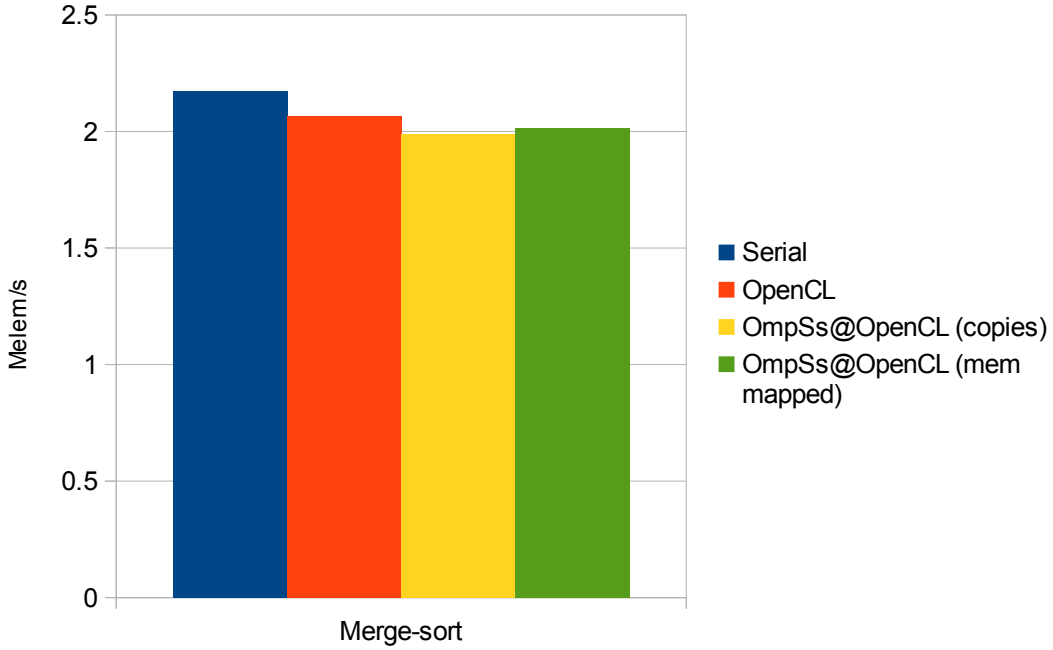

Figure 8: Performance obtained in Merge-sort – Melem/s in single precision. Higher is better.

P Conclusions

In this deliverable we have shown the evaluation of two sets of benchmarks on the Mont-Blanc node: Standard and Mont-Blanc benchmarks. With the first set we have determined that the performance of the Mont-Blanc node is equivalent to the performance of the Arndale board that we were using as a prototype.

With the second set, we have compared the performance of four versions of the benchmarks: serial, OpenCL, OmpSs@OpenCL with data transfers and OmpSs@OpenCL using memory mapped onto the device. Results show that for the algorithms that have more regular memory accesses, the performance of the MALI GPU is larger than that of the ARM Cortex-A15 cores. On the contrary, for the algorithms that are more irregular regarding memory access, the performance of the ARM cores and the MALI GPU is similar.

References

[1] HPC Challenge Benchmark, University of Tennessee, http://icl.cs.utk.edu/hpcc/.
[2] SPEC Benchmarks, Standard Performance Evaluation Corporation (SPEC), Gainesville, VA, http://www.spec.org/.
[3] TOP 500 Supercomputer Site, http://www.top500.org.
[4] Duran, A., Ayguade, E., Badia, R.M., Labarta, J., Martinell, L., Martorell, X., Planas, J., "OmpSs: A Proposal for Programming Heterogeneous Multi-Core Architectures", Parallel Processing Letters, 21, 173(2011), doi: 10.1142/S0129626411000151.
[5] SPEC CPU2000 results, http://www.spec.org/cpu2000/results/cpu2000.html.