GPUBenchmark results for zape2

June 6, 2012

Abstract

This report shows the GPUBenchmark results obtained on zape2 on June 6, 2012.

Contents

1 Introduction

GPUBenchmark has been used to evaluate different aspects of the zape2 computer. Depending on its hardware architecture and the libraries available when GPUBenchmark was run, some or all of the following aspects will be reported in this document:

- Transfer speed between hard disk and main memory.
- Transfer speed between GPU and main memory.
- Transfer speed between two GPUs.
- Matrix-matrix multiplication performance.
- Matrix-vector multiplication performance.

The next section describes the hardware characteristics of zape2. Each one of the remainder sections will focus in one of the previously enumerated performance aspects.

2 Hardware description

This section shows the characteristics of the CPUs and GPU of zape2. The CPUs available at zape2 have the next characteristics:

```
AMD Phenom(tm) 9550 Quad-Core Processor
 cpu MHz    : 2200.176
 cache size : 512 KB
 cpu cores  : 4
 bogomips   : 4400.34

AMD Phenom(tm) 9550 Quad-Core Processor
 cpu MHz    : 2200.176
 cache size : 512 KB
 cpu cores  : 4
 bogomips   : 4400.31

AMD Phenom(tm) 9550 Quad-Core Processor
 cpu MHz    : 2200.176
 cache size : 512 KB
 cpu cores  : 4
 bogomips   : 4400.28

AMD Phenom(tm) 9550 Quad-Core Processor
 cpu MHz    : 2200.176
 cache size : 512 KB
 cpu cores  : 4
 bogomips   : 4400.29
```

The GPU Device installed on zape2 that has been used to perform some of the tests has the next characteristics:

```
Tesla C1060
 CUDA driver version            : 4000
 CUDA Runtime version           : 4000
 Multiprocessors                : 30
 Global memory (total)          : 4294770688 bytes
 Constant memory (total)        : 65536 bytes
 Shared memory per block (total) : 16384 bytes
 Available registers per block  : 16384
 Threads per block              : 512
 Max. dimension of a block      : 512 x 512 x 64
 Max. dimension of a grid       : 65535 x 65535 x 1
 Clock rate                     : 1.30 GHz
```

3 Transfer speed between hard disk and main memory

To obtain the transfer speed from hard disk to main memory, and from main memory to hard disk, several matrices of floats with different number of rows and columns have been created, and the

time required to transfer these matrices from hard disk to main memory, and viceversa, has been measured.

In order to obtain a more accurate measure, for each number of rows and columns, ten transfers were carried out in both directions. The median of the results for each case is reported.

Table 1 shows the transfer speed obtained for several numbers of rows and columns, from hard disk to main memory, and from main memory to hard disk. Note that the transfer speed is given in Mebibytes per second (MiB/s)[1]. These results are reported graphically in Figure 1.

Rows	Columns	Size (MiB)	Hard disk → Main memory (MiB/s)	Main memory → Hard disk (MiB/s)
128	128	0.06	129.67	55.41
256	256	0.25	174.83	89.19
512	512	1.00	162.89	113.21
1024	1024	4.00	35.45	134.97
2048	2048	16.00	69.24	88.25
4096	4096	64.00	82.42	80.24
8192	8192	256.00	90.38	80.41
10240	10240	400.00	91.85	80.85

Table 1: Transfer speed from hard disk to main memory, and from main memory to hard disk, for different matrix sizes

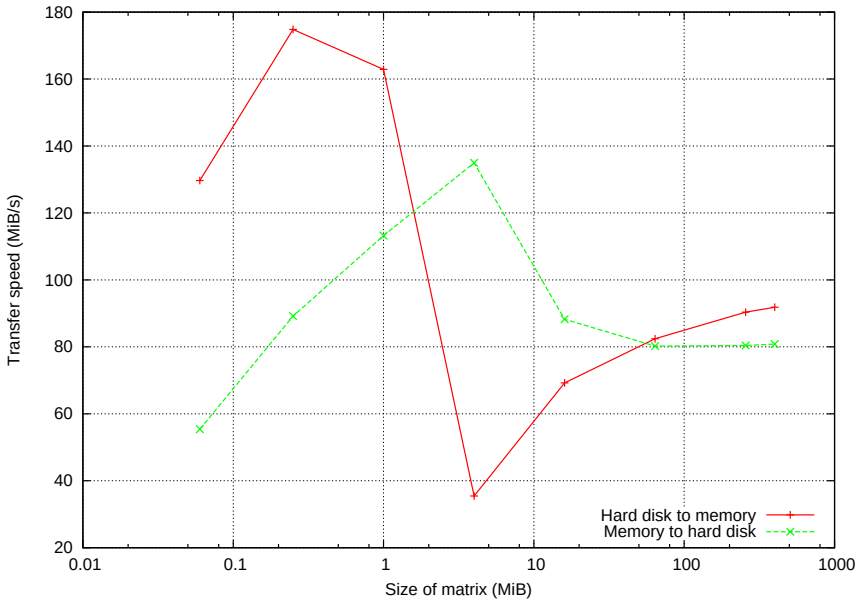


Figure 1: Transfer speed from hard disk to main memory, and from main memory to hard disk, for different matrix sizes

[1]A Mebibyte is defined as 2^{20} bytes

4 Transfer speed between GPU and main memory

To obtain the transfer speed from GPU internal memory to main memory, and from main memory to GPU internal memory, several matrices of floats with different number of rows and columns have been created, and the time required to transfer these matrices from GPU internal memory to main memory, and viceversa, has been measured.

In order to obtain a more accurate measure, for each number of rows and columns, ten transfers were carried out in both directions. The median of the results for each case is reported.

Table 2 shows the transfer speed obtained for several numbers of rows and columns, from GPU to main memory, and from main memory to GPU. Note that the transfer speed is given in Mebibytes per second (MiB/s)[2].

Rows	Columns	Size (MiB)	GPU → Main memory (MiB/s)	Main memory → GPU (MiB/s)
128	128	0.06	854.01	940.81
256	256	0.25	1000.19	1396.34
512	512	1.00	1504.65	1560.00
1024	1024	4.00	2131.11	1889.84
2048	2048	16.00	2398.53	2365.26
4096	4096	64.00	875.48	2521.11
8192	8192	256.00	899.00	2573.25
10240	10240	400.00	901.02	2574.16

Table 2: Transfer speed from GPU to main memory, and from main memory to GPU, for different matrix sizes

The same tests have been done using padding to allocate the matrices in the GPU internal memory. When padding is applied, it may be necessary to reserve additional storage to ensure that corresponding pointers in any given row will continue to meet the alignment requirements for coalescing. Padding is the recommended allocation method for 2D arrays.

Table 3 shows the transfer speed obtained for several numbers of rows and columns, from GPU to main memory, and from main memory to GPU, when padding is in use. Note that the transfer speed is given in Mebibytes per second.

Rows	Columns	Size (MiB)	GPU → Main memory (MiB/s)	Main memory → GPU (MiB/s)
128	128	0.06	1410.20	1290.90
256	256	0.25	2541.48	2132.81
512	512	1.00	3050.27	2497.20
1024	1024	4.00	3201.11	2600.70
2048	2048	16.00	3260.28	2637.97
4096	4096	64.00	3276.14	2647.71
8192	8192	256.00	3283.43	2650.05
10240	10240	400.00	3284.61	2650.34

Table 3: Transfer speed from GPU to main memory, and from main memory to GPU, when using padding, for different matrix sizes

[2] A Mebibyte is defined as 2^{20} bytes

Figure 2 shows the transfer speed from GPU to main memory, and from main memory to GPU, with and without padding.

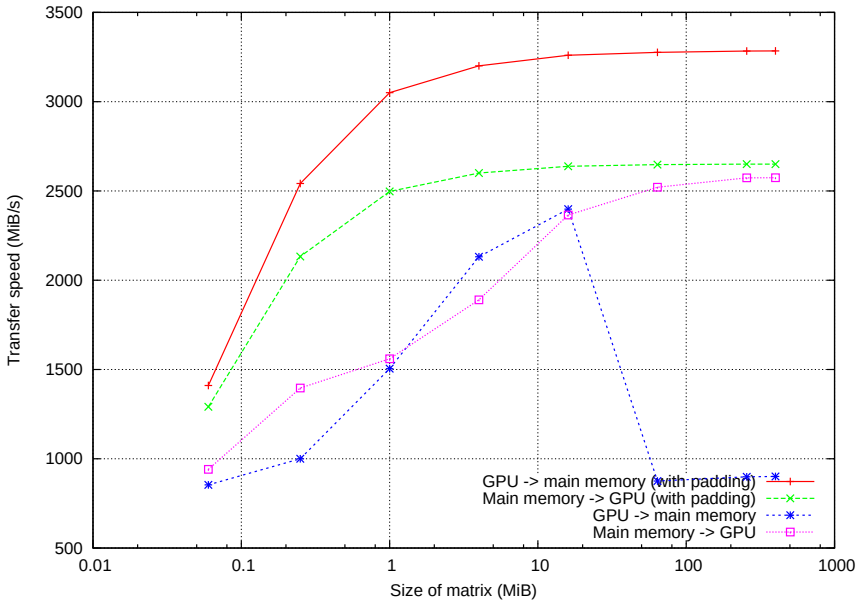


Figure 2: Transfer speed from GPU to main memory, and from main memory to GPU, with and without padding, for different matrix sizes

5 Matrix-matrix multiplication performance

This section shows both the time required and the GFLOPS obtained by the CPU and the GPU on zape2 when computing the operation:

$$C = \alpha AB + \beta C,$$

where $A \in \mathcal{R}^{m \times k}$, $B \in \mathcal{R}^{k \times n}$, $C \in \mathcal{R}^{m \times n}$, and α and β are scalars.

This operation has been performed on the CPU by calling the CBLAS `cblas_sgemm()` function, and on the GPU by calling the CUBLAS `cublasSgemm()` function. Different m, n, and k values have been used. For each value of m, the n and k values have been obtained as 25, 50, 75, and 100% of m.

In order to obtain a more accurate measure, for each combination of m, n, and k, ten operations have been carried on in both CPU and GPU. Then, the median of the outcomes for each case has been computed.

The GFLOPS for each combination of m, n, and k has been computed as:

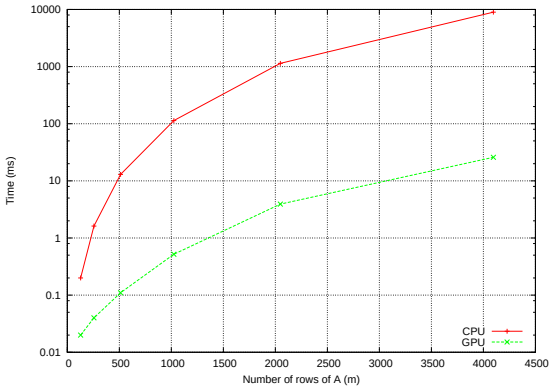
$$GFLOPS = \frac{2mnk \cdot 10^{-9}}{time}$$

Table 4 shows the time and GFLOPS results obtained for the given m, n, and k values.

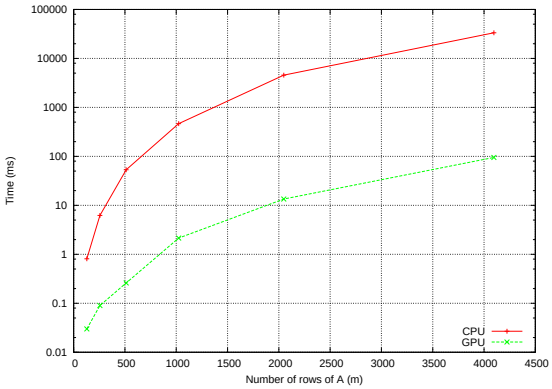
Figure 3 shows the time to compute the operation $C = \alpha AB + \beta C$ on the CPU and on the GPU (notice that n and k are 25, 50, 75, and 100% of m). Figure 4 shows the CPU and GPU GFLOPS for these values of m, n and k.

m	n	k	CPU (ms)	GPU (ms)	CPU GFLOPS	GPU GFLOPS
128	32	32	0.20	0.02	1.29	11.89
128	64	64	0.81	0.03	1.30	34.17
128	96	96	1.86	0.05	1.27	48.38
128	128	128	3.35	0.06	1.25	70.13
256	64	64	1.62	0.04	1.29	55.82
256	128	128	6.22	0.09	1.35	90.21
256	192	192	14.27	0.13	1.32	150.43
256	256	256	25.61	0.18	1.31	186.58
512	128	128	12.99	0.11	1.29	156.74
512	256	256	53.32	0.26	1.26	257.26
512	384	384	122.53	0.49	1.23	305.21
512	512	512	265.08	1.17	1.01	229.94
1024	256	256	113.51	0.52	1.18	259.42
1024	512	512	465.04	2.15	1.15	250.22
1024	768	768	1262.50	3.55	0.96	340.43
1024	1024	1024	2342.70	7.28	0.92	295.12
2048	512	512	1137.88	3.92	0.94	273.91
2048	1024	1024	4551.10	13.46	0.94	319.08
2048	1536	1536	10280.07	28.42	0.94	340.04
2048	2048	2048	18131.85	48.61	0.95	353.42
4096	1024	1024	8959.34	25.81	0.96	332.75
4096	2048	2048	33418.17	95.02	1.03	361.60
4096	3072	3072	74729.53	210.43	1.03	367.38
4096	4096	4096	132820.35	371.98	1.03	369.48

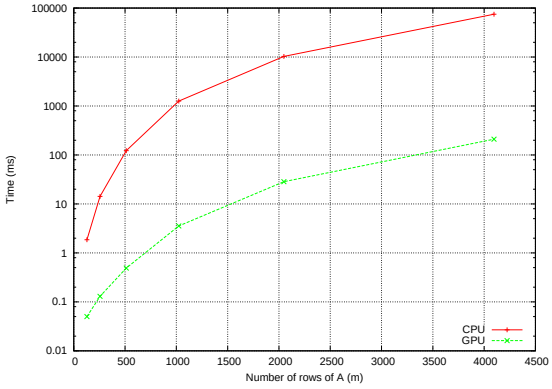
Table 4: Time and GFLOPS for the operation $C = \alpha AB + \beta C$ on the CPU and on the GPU for different m, n and k values



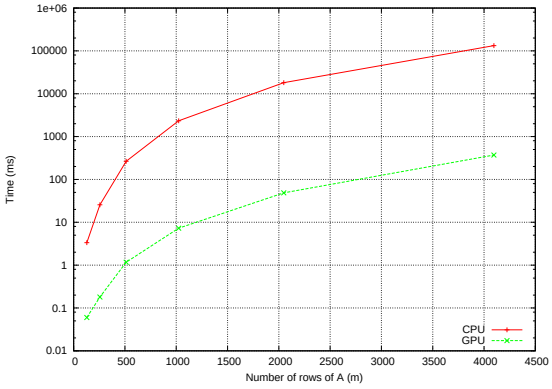
(a) n and k are 25% of m



(b) n and k are 50% of m

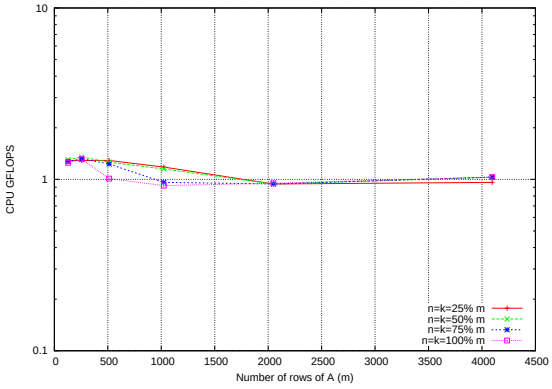


(c) n and k are 75% of m



(d) n and k are 100% of m

Figure 3: Time to compute the operation $C = \alpha AB + \beta C$ on the CPU and on the GPU



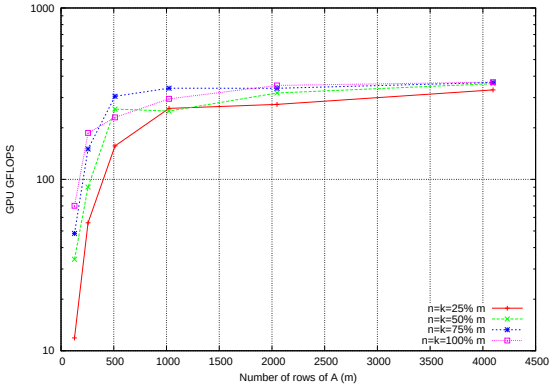


Figure 4: CPU and GPU GFLOPS for $C = \alpha AB + \beta C$ with several m, n and k values (n and k are 25, 50, 75, and 100% of m)

6 Matrix-vector multiplication performance

This section shows both the time required and the GFLOPS obtained by the CPU and the GPU on zape2 when computing the operation:

$$y = \alpha Ax + \beta y,$$

where $A \in \mathcal{R}^{m \times n}$, $x \in \mathcal{R}^n$, $y \in \mathcal{R}^m$, and α and β are scalars.

This operation has been performed on the CPU by calling the CBLAS `cblas_sgemv()` function, and on the GPU by calling the CUBLAS `cublasSgemv` function. Different m and n values have been used. For each value of m, the n values have been obtained as 25, 50, 75, and 100% of m.

In order to obtain a more accurate measure, for each combination of m and n, ten operations have been carried on in both CPU and GPU. Then, the median of the outcomes for each case has been computed.

The GFLOPS for each combination of m and n has been computed as:

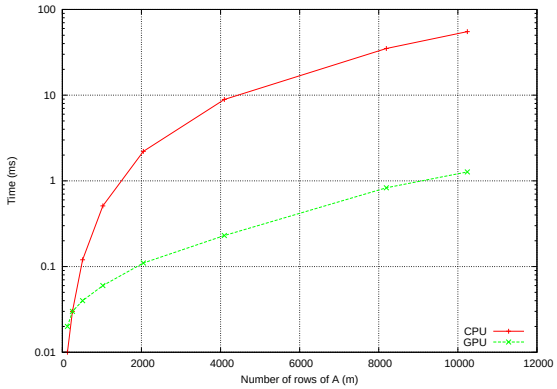
$$GFLOPS = \frac{2mn \cdot 10^{-9}}{time}$$

Table 5 shows the time and GFLOPS results obtained for the given m and n values.

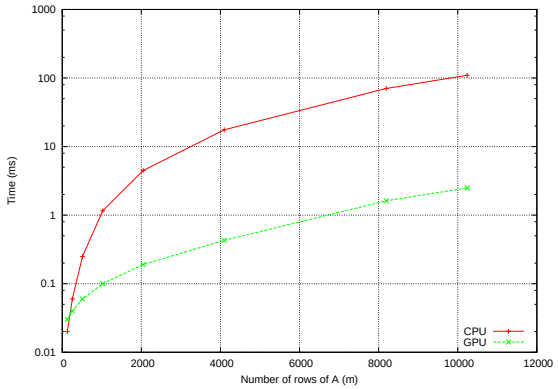
Figure 5 shows the time to compute the operation $y = \alpha Ax + \beta y$ on the CPU and on the GPU (notice that n is 25, 50, 75, and 100% of m). Figure 6 shows the CPU and GPU GFLOPS for these values of m and n.

m	n	CPU (ms)	GPU (ms)	CPU GFLOPS	GPU GFLOPS
128	32	0.01	0.02	1.02	0.37
128	64	0.02	0.03	0.96	0.61
128	96	0.03	0.03	0.85	0.77
128	128	0.03	0.04	1.02	0.89
256	64	0.03	0.03	1.02	1.22
256	128	0.06	0.04	1.04	1.62
256	192	0.10	0.05	1.02	1.99
256	256	0.13	0.06	1.00	2.22
512	128	0.12	0.04	1.07	3.26
512	256	0.25	0.06	1.05	4.30
512	384	0.38	0.08	1.04	4.91
512	512	0.51	0.10	1.03	5.31
1024	256	0.51	0.06	1.02	8.41
1024	512	1.16	0.10	0.90	10.05
1024	768	1.67	0.15	0.94	10.67
1024	1024	2.25	0.19	0.93	11.21
2048	512	2.21	0.11	0.95	18.72
2048	1024	4.48	0.19	0.94	21.78
2048	1536	6.62	0.27	0.95	23.01
2048	2048	8.92	0.36	0.94	23.59
4096	1024	8.91	0.23	0.94	36.39
4096	2048	17.58	0.43	0.95	39.03
4096	3072	26.43	0.63	0.95	40.01
4096	4096	35.03	0.83	0.96	40.53
8192	2048	35.06	0.83	0.96	40.58
8192	4096	70.34	1.62	0.95	41.45
8192	6144	104.93	2.41	0.96	41.77
8192	8192	140.65	3.20	0.95	41.88
10240	2560	55.13	1.27	0.95	41.34
10240	5120	109.65	2.49	0.96	42.04
10240	7680	164.42	3.72	0.96	42.25
10240	10240	219.27	4.95	0.96	42.39

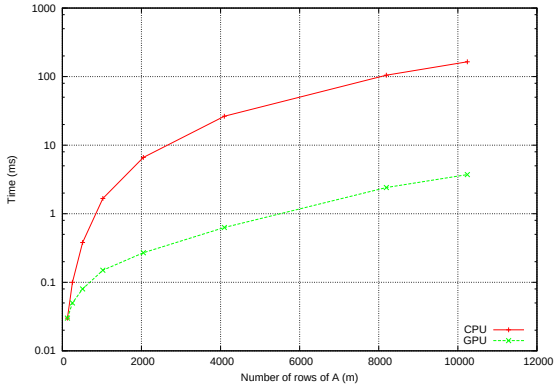
Table 5: Time and GFLOPS for the operation $y = \alpha Ax + \beta y$ on the CPU and on the GPU for different m and n values



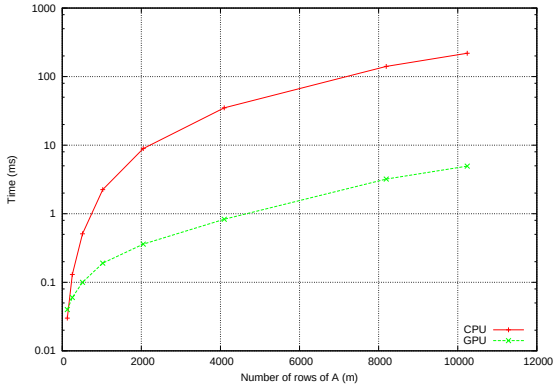
(a) n is 25% of m



(b) n is 50% of m

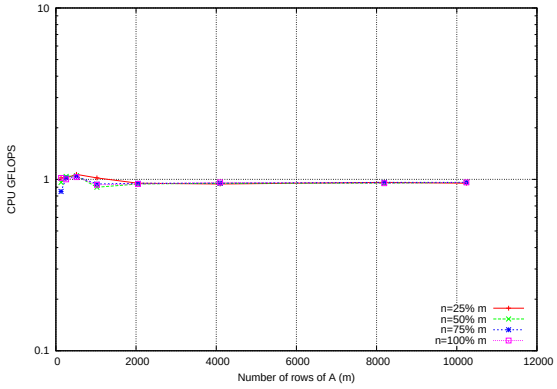


(c) n is 75% of m



(d) n is 100% of m

Figure 5: Time to compute the operation $y = \alpha Ax + \beta y$ on the CPU and on the GPU



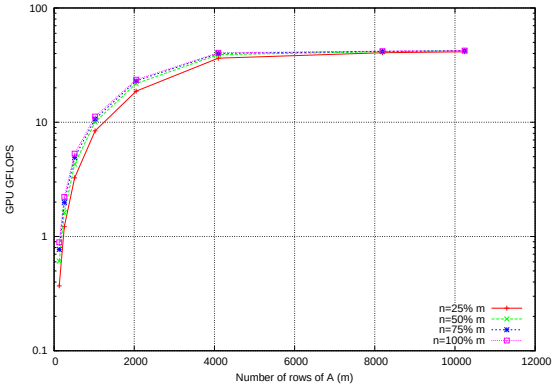


Figure 6: CPU and GPU GFLOPS for $y = \alpha Ax + \beta y$ with several m and n values (n is 25, 50, 75, and 100% of m)