GPUBenchmark results for testbed

September 17, 2012

Abstract

This report shows the GPUBenchmark results obtained on testbed on September 17, 2012.

Contents

1 Introduction

GPUBenchmark has been used to evaluate different aspects of the testbed 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 testbed. 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 testbed. The CPUs available at testbed have the next characteristics:

```
Intel(R) Core(TM)2 CPU 6320 @ 1.86GHz
 cpu MHz    : 1866.650
 cache size : 4096 KB
 cpu cores  : 2
 bogomips   : 3736.13

Intel(R) Core(TM)2 CPU 6320 @ 1.86GHz
 cpu MHz    : 1866.650
 cache size : 4096 KB
 cpu cores  : 2
 bogomips   : 3733.46
```

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

```
GeForce 8800 Ultra
 CUDA driver version            : 4000
 CUDA Runtime version           : 4000
 Multiprocessors                : 16
 Global memory (total)          : 804978688 bytes
 Constant memory (total)        : 65536 bytes
 Shared memory per block (total) : 16384 bytes
 Available registers per block  : 8192
 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                     : 1.51 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.

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

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

Rows	Columns	Size (MiB)	Hard disk → Main memory (MiB/s)	Main memory → Hard disk (MiB/s)
128	128	0.06	53.93	1.27
256	256	0.25	72.03	2.82
512	512	1.00	130.48	9.34
1024	1024	4.00	167.78	22.91
2048	2048	16.00	45.85	37.98
4096	4096	64.00	52.50	47.43
8192	8192	256.00	56.04	53.21
10240	10240	400.00	57.53	54.61

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

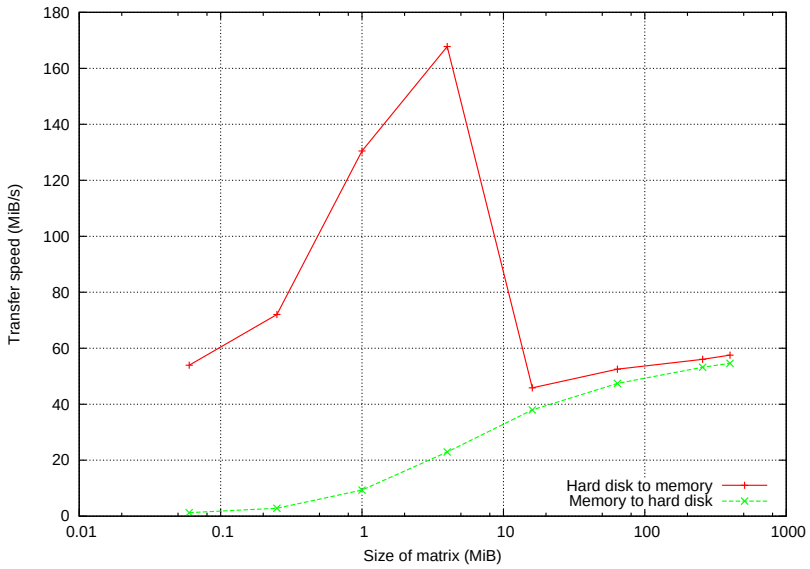


Figure 1: Transfer speed from hard disk to main memory, and from main memory to hard disk, for different matrix sizes

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	731.78	1085.67
256	256	0.25	851.78	1552.87
512	512	1.00	562.81	1620.77
1024	1024	4.00	711.35	1645.62
2048	2048	16.00	723.87	1487.04
4096	4096	64.00	760.90	1626.25
8192	8192	256.00	829.95	1671.85
10240	10240	400.00	814.16	1669.34

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	1289.19	1610.16
256	256	0.25	1688.09	2236.62
512	512	1.00	1834.03	2490.64
1024	1024	4.00	1892.39	2567.90
2048	2048	16.00	2002.03	2615.87
4096	4096	64.00	2043.36	2629.54
8192	8192	256.00	2051.94	2632.92
10240	10240	400.00	2052.37	2633.16

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

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

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

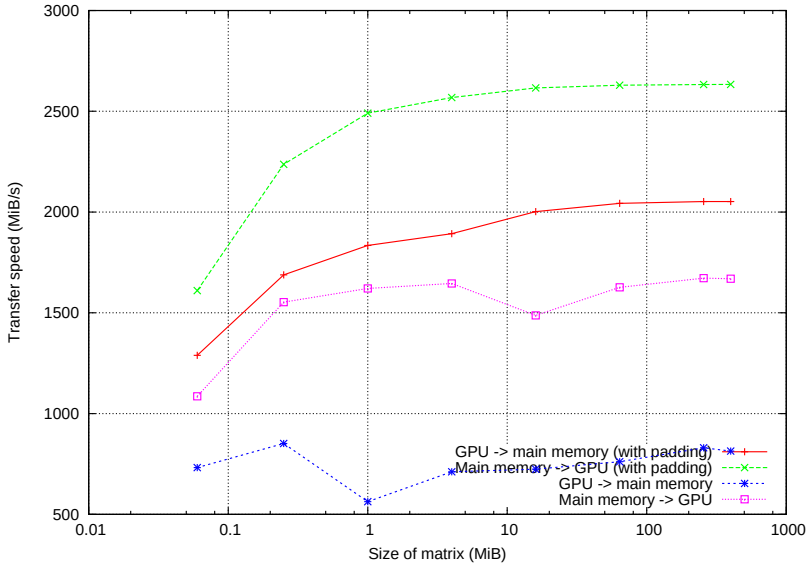


Figure 2: Transfer speed from GPU to main memory, and from main memory to GPU, with and without padding, for different matrix sizes

5 Matrix-matrix multiplication performance

This section shows both the time required and the GFLOPS obtained by the CPU and the GPU on testbed 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.

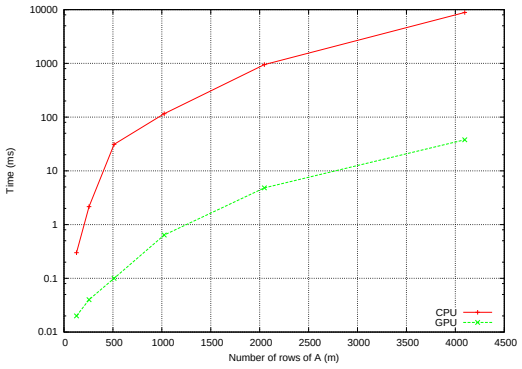
6 Matrix-vector multiplication performance

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

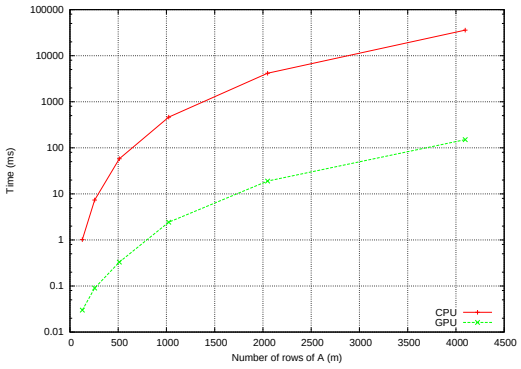
$$y = \alpha Ax + \beta y,$$

m	n	k	CPU (ms)	GPU (ms)	CPU GFLOPS	GPU GFLOPS
128	32	32	0.30	0.02	0.88	16.22
128	64	64	1.02	0.03	1.03	40.50
128	96	96	2.33	0.04	1.01	56.58
128	128	128	4.15	0.05	1.01	81.77
256	64	64	2.16	0.04	0.97	53.02
256	128	128	7.36	0.09	1.14	91.47
256	192	192	19.35	0.11	0.98	167.95
256	256	256	34.42	0.17	0.97	193.46
512	128	128	31.15	0.10	0.54	169.95
512	256	256	57.81	0.33	1.16	203.19
512	384	384	131.04	0.72	1.15	210.05
512	512	512	233.55	1.23	1.15	218.08
1024	256	256	115.58	0.64	1.16	209.06
1024	512	512	463.67	2.43	1.16	220.83
1024	768	768	1053.65	5.40	1.15	223.58
1024	1024	1024	1963.61	9.52	1.09	225.46
2048	512	512	947.33	4.82	1.13	222.72
2048	1024	1024	4159.92	18.97	1.03	226.40
2048	1536	1536	9850.51	42.51	0.98	227.30
2048	2048	2048	17904.60	75.38	0.96	227.91
4096	1024	1024	8854.39	37.87	0.97	226.80
4096	2048	2048	35879.49	150.74	0.96	227.94
4096	3072	3072	80611.20	338.59	0.96	228.33
4096	4096	4096	143886.59	601.65	0.96	228.44

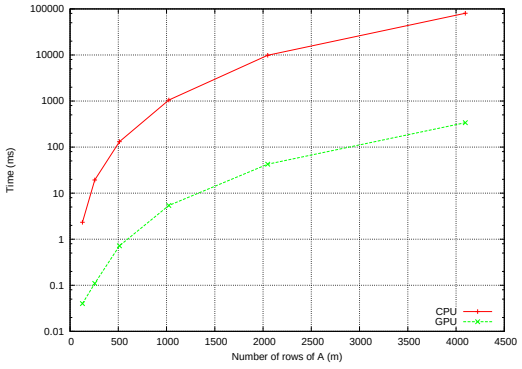
Table 4: Time and GFLOPS for the operation $C = \alpha AB + \beta C$ on the CPU and on the GPU for different m, n and k values



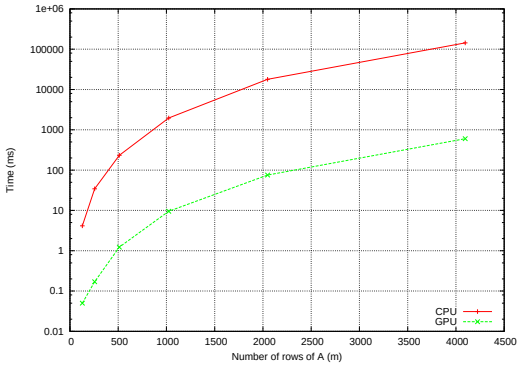
(a) n and k are 25% of m



(b) n and k are 50% of m

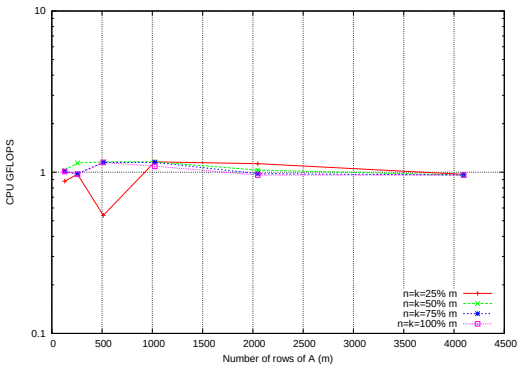


(c) n and k are 75% of m



(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



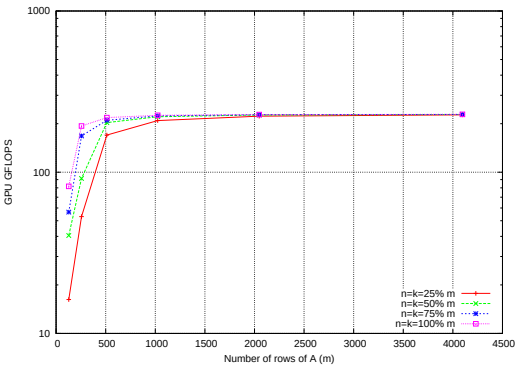


Figure 4: CPU and GPU GFLOPS for $C = \alpha AB + \beta C$ with several m, n and k values (n and k are 25, 50, 75, and 100% of m)

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:

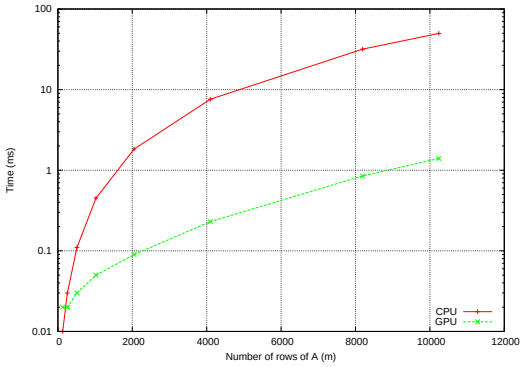
$$GFLOPS = \frac{2mn \cdot 10^{-9}}{time}$$

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

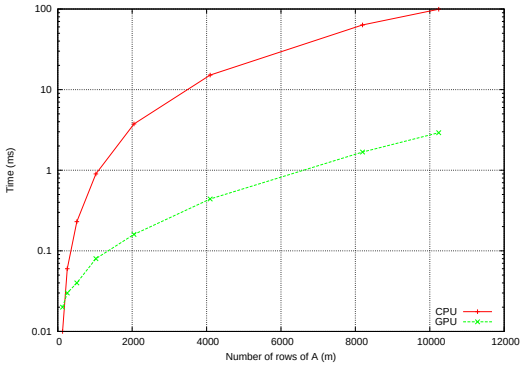
m	n	CPU (ms)	GPU (ms)	CPU GFLOPS	GPU GFLOPS
128	32	0.01	0.02	1.02	0.48
128	64	0.01	0.02	1.09	0.77
128	96	0.02	0.02	1.07	0.98
128	128	0.03	0.03	1.09	1.16
256	64	0.03	0.02	1.13	1.57
256	128	0.06	0.03	1.13	2.29
256	192	0.09	0.04	1.13	2.72
256	256	0.11	0.04	1.14	2.99
512	128	0.11	0.03	1.16	4.56
512	256	0.23	0.04	1.15	5.94
512	384	0.34	0.06	1.16	6.60
512	512	0.45	0.07	1.16	7.02
1024	256	0.45	0.05	1.17	11.57
1024	512	0.90	0.08	1.17	13.68
1024	768	1.36	0.11	1.16	14.08
1024	1024	1.83	0.15	1.15	14.15
2048	512	1.83	0.09	1.14	22.80
2048	1024	3.75	0.16	1.12	25.65
2048	1536	5.65	0.23	1.11	27.09
2048	2048	7.58	0.30	1.11	27.71
4096	1024	7.59	0.23	1.10	36.09
4096	2048	15.17	0.44	1.11	38.27
4096	3072	22.80	0.65	1.10	38.94
4096	4096	30.28	0.85	1.11	39.30
8192	2048	31.77	0.85	1.06	39.39
8192	4096	63.44	1.68	1.06	40.03
8192	6144	95.37	2.53	1.06	39.80
8192	8192	126.83	3.37	1.06	39.79
10240	2560	49.81	1.40	1.05	37.41
10240	5120	99.46	2.92	1.05	35.87
10240	7680	148.56	4.37	1.06	36.00
10240	10240	199.28	5.82	1.05	36.04

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

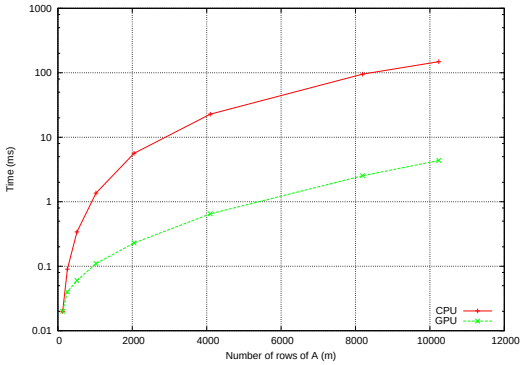
Figure 5 shows the time to compute the operation $y = \alpha Ax + \beta y$ on the CPU and on the GPU (notice that n is 25, 50, 75, and 100% of m). Figure 6 shows the CPU and GPU GFLOPS for these values of m and n.



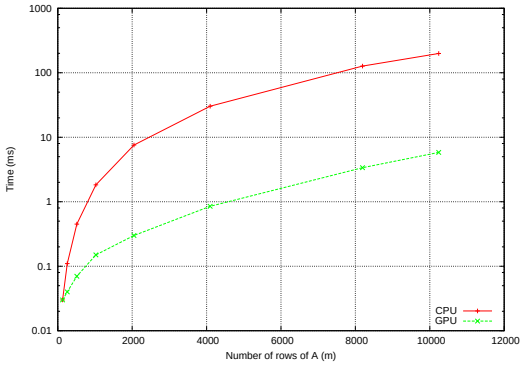
(a) n is 25% of m



(b) n is 50% of m

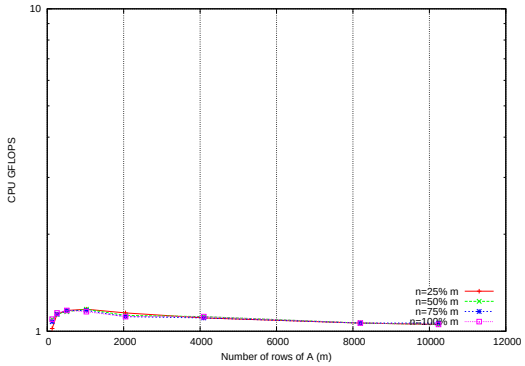


(c) n is 75% of m



(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



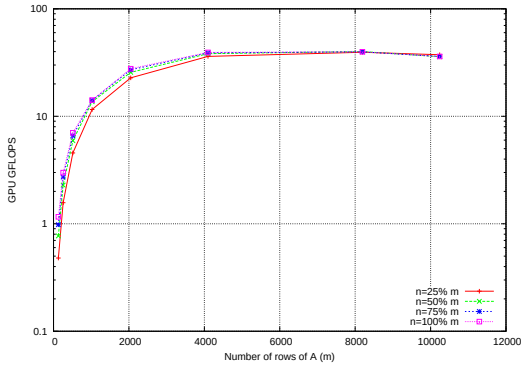


Figure 6: CPU and GPU GFLOPS for $y = \alpha Ax + \beta y$ with several m and n values (n is 25, 50, 75, and 100% of m)