Karlsruhe Institute of Technology

GPU Accelerated Scientific Computing: Evaluation of the NVIDIA Fermi Architecture; Elementary Kernels and Linear Solvers

Hartwig Anzt
Tobias Hahn
Vincent Heuveline
Björn Rocker

No. 2010-04

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

KIT – University of the State of Baden-Wuerttemberg and
National Research Center of the Helmholtz Association

www.emcl.kit.edu

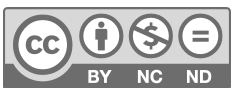
Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191–0693
No. 2010-04

Impressum

Karlsruhe Institute of Technology (KIT)
Engineering Mathematics and Computing Lab (EMCL)

Fritz-Erler-Str. 23, building 01.86
76133 Karlsruhe
Germany

KIT – University of the State of Baden Wuerttemberg and
National Laboratory of the Helmholtz Association



GPU Accelerated Scientific Computing: Evaluation of the NVIDIA Fermi Architecture; Elementary Kernels and Linear Solvers

Hartwig Anzt, Tobias Hahn, Vincent Heuveline, Björn Rocker

Karlsruhe Institute of Technology (KIT)
Institute for Applied and Numerical Mathematics 4
76128 Karlsruhe, Germany
hartwig.anzt@kit.edu,tobias.hahn@kit.edu,vincent.heuveline@kit.edu,
bjoern.rocker@kit.edu

Abstract. This paper evaluates the latest GPU generation from NVIDIA, called "Fermi", with respect to the previous generation. The experiments include benchmarks of elementary kernels as well as of linear solvers applied to problems arising in the area of computational fluid dynamics.
Both the consumer version of the hardware (GeForce GTX 480 and GTX 280) as well as the professional line (Tesla C2050 and C1060) are taken into account.

1 Introduction

Recently, the number of users and lines of code taking advantage of the computational power of accelerators, especially GPUs, grew enormously. One reason is the facilitated programmability of GPUs by NVIDIA's CUDA and OpenCL. With the introduction of full double precision support on GPUs, many scientific projects started using e.g. finite difference and finite element techniques for the solution of systems of partial differential equations (PDEs) using GPU hardware. Since 2003, several papers described the solution of the Navier-Stokes equations for incompressible fluid flow on the GPUs [1,2] or other boundary value problems [4]. An analysis of a meteorological simulation for tropical cyclones and an implementation of rigid particle flows on GPUs can be found in [3].

With the introduction of built-in double-precision support and IEEE754 compatibility, GPUs evolve towards universally usable processing units. Still, their paradigm is related to former graphics stream processing: The same series of operations is applied to every element of a set of data (i.e. a stream). Operations of a kernel are pipelined, such that many stream processors can process the stream in parallel. The limiting factor in this context is memory latency, especially when data dependency is high and data locality is low. GPUs always try to hide memory latency by executing many kernel instances in parallel on the same core. Switching these lightweight "threads" and operating on other register sets can be done in just a few cycles, whereas the cost of fetching data from the global memory extends several hundreds of cycles.

While the problem described above is often inherent for many-core computing, other restrictions of stream processing techniques have been addressed in CUDA, which offer e.g. gather and scatter operations on the global graphics memory. Furthermore, CUDA-capable devices can be programmed with slightly extended C and runtime libraries, including hardware support for double precision (obeying IEEE 754).

In fall 2009 NVIDIA released their new chip architecture "Fermi". This paper compares the performance of this new architecture to former generations.

2 Hardware and Software Environment

2.1 Hardware Overview

As hardware platform for the evaluation we have chosen the accelerators from the actual and previous generation both from the NVIDIA professional line (Tesla), as well as from the NVIDIA consumer line (GeForce).

Name	Tesla C2050	Tesla C1060	GTX 480	GTX 280a
Chip	T20	T10	GF100	GT200
Transistors	ca. 3 Mrd.	ca. 1,4 Mrd.	ca. 3 Mrd.	ca. 1,4 Mrd.
Core frequency	1.15 GHz	1.3 GHz	1.4 GHz	1.3 GHz
Shaders (MADD)	448	240	480	240
GFLOPs (single)	1030	933	1.345	933
GFLOPs (double)	515	78	168	78
Memory	3 GB GDDR5	4 GB GDDR3	1.5 GB GDDR5	1 GB GDDR3
Memory Frequency	1.5 GHz	0.8 GHz	1.8 GHz	1.1 GHz
Memory Bandwidth	144 GB/s	102 GB/s	177 GB/s	141 GB/s
ECC Memory	yes	no	no	no
Power Consumption	247 W	187 W	250 W	236 W
IEEE double/single	yes/yes	yes/partial	yes/yes	yes/partial

Table 1. Key system characteristics of the four GPUs used for the tests. Computation rate and memory bandwidth are peak respectively theoretical values.

Host				Device				
CPU	MEM [GB]	BW [GB/s]	H2D [GB/s]	GPU	MEM [GB]	BW [GB/s]	D2H [GB/s]	CC ECC
2 x Intel Xeon (E5520, 4 cores)	32	12.07	PA: 3.25 PI: 5.86	Tesla T20	3	BT: 91.28 daxpy: 82.5 ddot: 88.3	PA: 2.51 PI: 4.75	2.0 Yes
2 x Intel Xeon (E5450, 4 cores)	16	6.14	PA: 1.92 PI: 5.44	Tesla T10	4	BT: 71.80 daxpy: 83.1 ddot: 83.3	PA: 1.55 PI: 3.77	1.3 No
1 x Intel Core2 (6600, 2 cores)	2	3.28	PA: 1.76 PI: 2.57	GTX 480	1.5	BT: 108.56 daxpy: 135.0 ddot: 146.7	PA: 1.38 PI: 1.82	2.0 No
1 x Intel Core i7 (920, 4 cores, SMT on)	6	12.07	PA:5.08 PI:5.64	GTX 280	1.0	BT: 111.54 daxpy: 124.3 ddot: 94.81	PA: 2.75 PI: 5.31	1.3 No

Table 2. Systems' configurations. The abbreviations are as follows: MEM is the amount of memory, BW the Bandwidth, H2D denotes the Host to Device bandwidth via PCIe and D2H the other way round, CC is the CUDA 'compute' capability and ECC is the availability of error correcting memory. PA means pageable memory is allocated, PI denotes the usage of pinned memory.

3 Numerical Experiments

In a first step, we perform benchmarks for some elementary kernels: we execute dot-products, vector updates, scalar-products, matrix-vector and matrix-matrix operations both in single and double precision.
In a second test, we evaluate the performance of a CG solver applied to a stencil-discretization of the Laplace equation.
Finally, we apply mixed precision iterative refinement solvers to linear systems arising in

the field of fluid dynamics. These implementations use the GPU as well as the CPU of the same system, enabling a performance evaluation for a more complex application.

3.1 Elementary Kernels Performance Evaluation

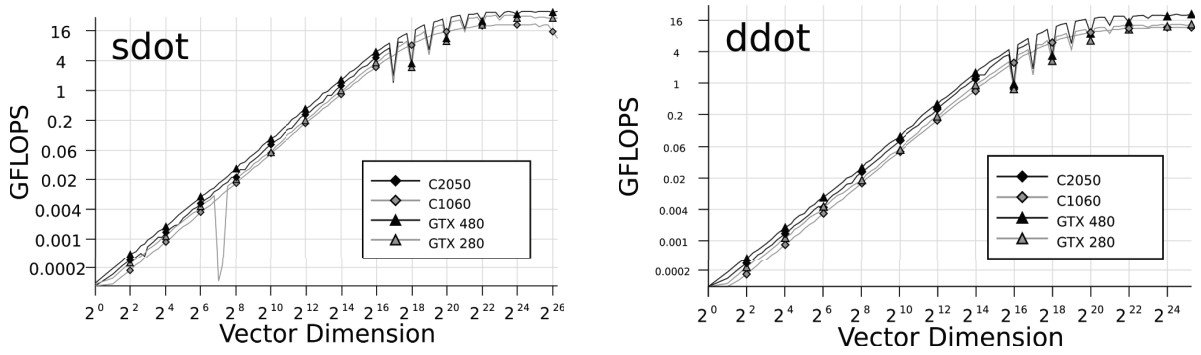


Fig. 1. Performance of the dot routine performed in single (sdot) and double precision (ddot).

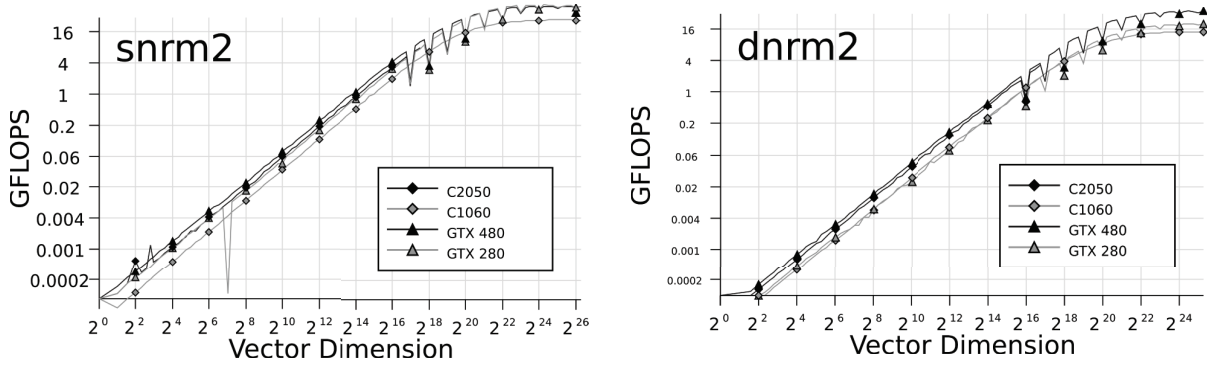


Fig. 2. Performance of the 2-norm routine performed in single (snrm2) and double precision (dnrm2).

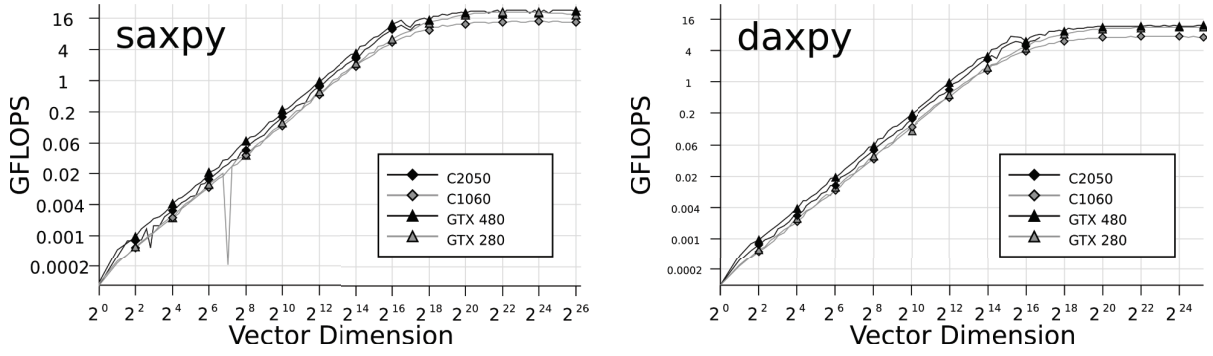


Fig. 3. Performance of the vector scale and add routine axpy performed in single (saxpy) and double precision (daxpy).

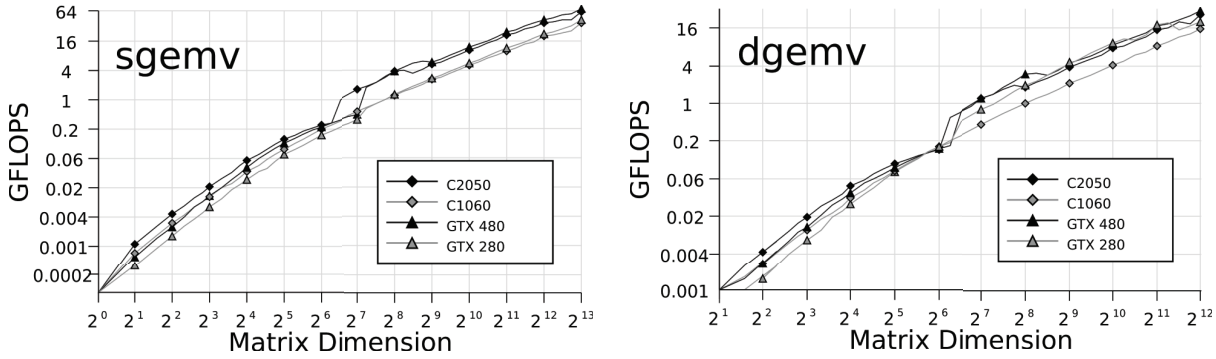


Fig. 4. Performance of dense matrix vector multiplication routine performed in single (sgemv) and double precision (dgemv).

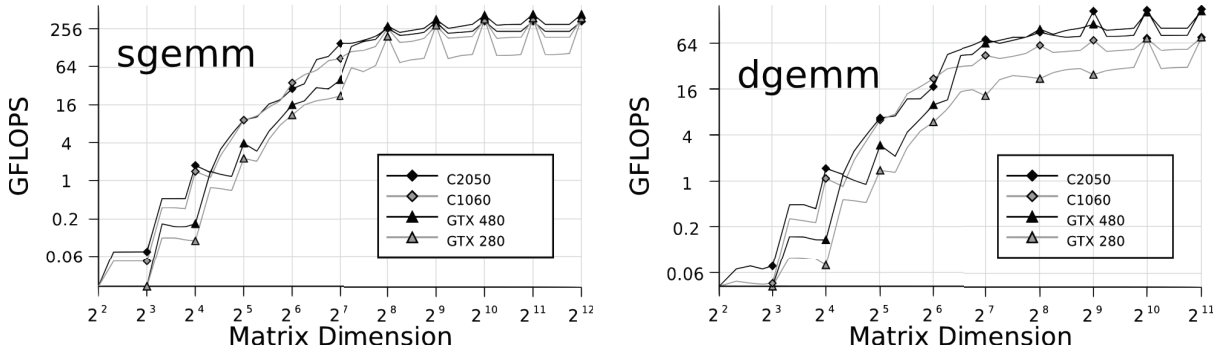


Fig. 5. Performance of the general matrix matrix multiplication routine gemm performed in single (sgemm) and double precision (dgemm).

Experiment Setup		Performance (GFLOP/s)			
Routine	Data size	C2050	C1060	GTX 480	GTX 280
sdot	185364	9.27	6.50	12.36	8.24
sdot	39903170	29.83	18.41	38.17	29.11
ddot	110218	5.38	3.50	6.68	4.16
ddot	39903170	18.79	11.03	19.38	12.48
snrm2	185364	7.72	4.88	9.76	7.41
snrm2	39903170	44.34	26.47	48.72	48.99
dnrm2	110218	3.06	1.79	3.39	1.79
dnrm2	39903170	22,74	13.49	33.22	19.22
saxpy	185364	11.96	8.62	14.83	11.23
saxpy	39903170	23.83	13.23	22.08	19.16
daxpy	110218	6.68	4.59	7.87	5.80
daxpy	39903170	12.54	6,92	11.33	10.52
sgemv	8192	58.19	34.63	68.72	40.49
dgemv	4096	25.39	14.74	29.69	19.52
sgemm	4096	330.17	367.61	430.40	368.75
dgemm	2048	174.21	74.40	161.97	73.76

Table 3. Performance for the elementary kernels for special data sizes.

3.2 CG Solver

We evaluated the performance of an implementation of the CG-algorithm (see e.g. [5]) based on the CSR-data format. The linear system is obtained from a finite element discretization

of the Laplace equation on a unit square using linear test-functions, which is equivalent to a finite differences discretization based on the 5-point-stencil. The matrix has the following characteristics: 4.000.000 degrees of freedom (dofs) and 19.992.000 nonzero entries (nnz).

All computations run exclusively on the accelerator and are performed in double precision, the stopping criteria for the residual is set to 10^{-6}.

First the performance results for the itemized kernels within the CG are presented and afterwards the complete runtime for the solver:

	C2050	C1060	GTX 480	GTX 280
ddot (sec)	0.000725	0.000768	0.000436	0.000675
ddot (Gbyte/s)	88.28	83.33	146.78	94.81
ddot (GFlops/s)	11.03	10.42	18.35	11.85
dscale+daxpy (sec)	0.00185	0.001916	0.001153	0.001285
dscale+daxpy (Gbyte/s)	51.89	50.10	83.26	74.71
dscale+daxpy (GFlop/s)	4.33	4.18	6.94	6.23
dcsrgemv (sec)	0.0187	0.019591	0.011527	0.013145
dcsrgemv (Gbyte/s)	17.10	16.33	27.75	24.34
dcsrgemv (GFlop/s)	2.14	2.04	3.47	3.04
daxpy (sec)	0.001163	0.001151	0.000711	0.000772
daxpy (Gbyte/s)	82.55	83.41	135.02	124.35
daxpy (GFlop/s)	6.88	6.95	11.25	10.36

Table 4. Performance evaluation of elementary kernels of the CG-algorithm on the four evaluated accelerators. All measurements are performed in double precision.

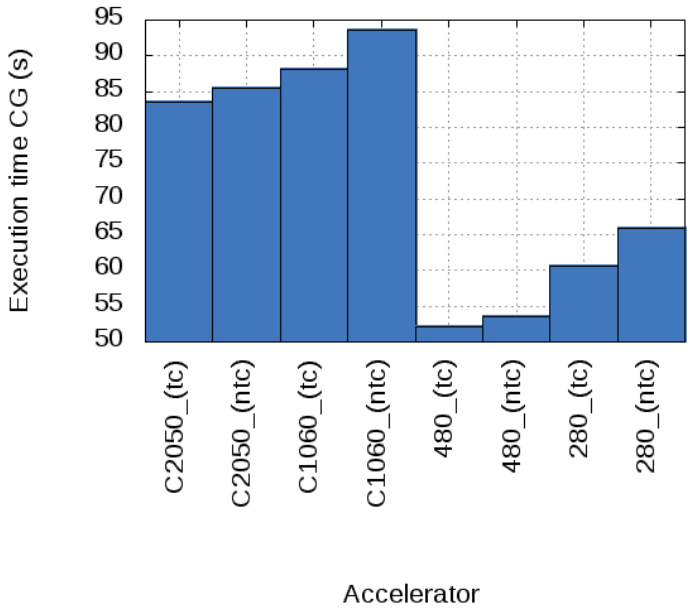


Fig. 6. Runtime of the CG algorithm for the Laplace test case for the four evaluated accelerators. *tc* employs the use of texture cache, *ntc* without using it.

3.3 Iterative Refinement Method

To be able to evaluate the computational power of the hardware platform in a more complex application, we use a GPU-implementation of a plain GMRES-(30) solver and a mixed-precision iterative refinement implementation based on the same solver. Mixed precision

iterative refinement solvers use a less complex floating point format for the inner error correction solver, and are therefore able to exploit the often superior low precision performance of GPUs and the double precision performance of the CPU [6], [8], [7].

Both, the plain double GMRES-(30) and the mixed precision variant use the relative residual stopping criterion of $\varepsilon = 10^{-10} \parallel r_0 \parallel_2$, while we choose $\varepsilon_{\text{inner}} = 10^{-1} \parallel r_i \parallel_2$ as inner stopping criterion for the error correction variant.

In case of the mixed precision iterative refinement implementation, the error correction solver is performed on the GPU, while the solution update is led to the CPU of the same system. This enables to handle larger problems, since the available memory on the GPU is usually very limited.

We compare the total needed computation time with the performance results on similar systems.

As test problems, we use three systems of linear equations CFD1, CFD2 and CFD3 affiliated with the 2D modeling of a Venturi Nozzle in different discretization fineness. The distinct number of supporting points leads to different matrix characteristics in terms of dimension, sparsity, and the condition number.

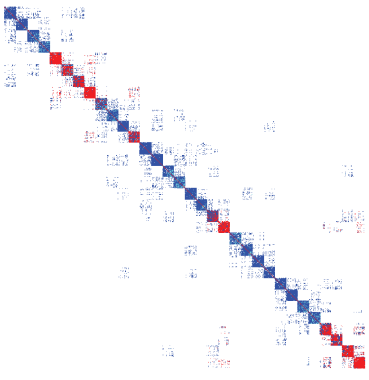
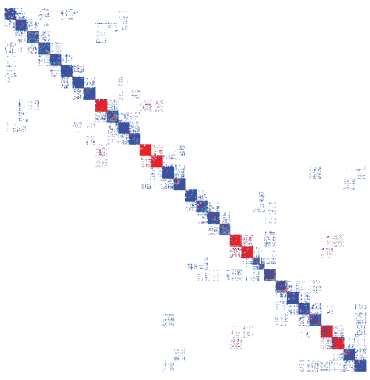
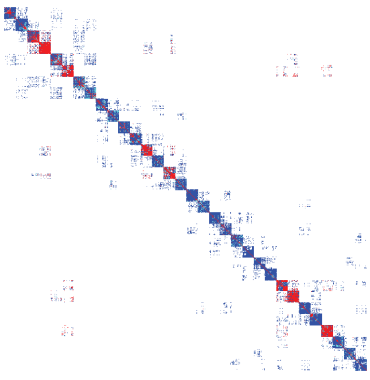
CFD1	**CFD2**	**CFD3**
		
problem: 2D fluid flow matrix dimension: $n = 395009$ sparsity: $nnz = 3544321$ storage format: CRS	problem: 2D fluid flow matrix dimension: $n = 634453$ sparsity: $nnz = 5700633$ storage format: CRS	problem: 2D fluid flow matrix dimension: $n = 1019967$ sparsity: $nnz = 9182401$ storage format: CRS

Table 5. Sparsity plots and properties of the CFD test-matrices.

Experiment setup		Computation Time (s)			
problem	solver type	C2050	C1060	GTX 480	GTX 280
CFD 1	plain double GMRES-(30)	164.845	252.749	145.237	183.375
	mixed precision GMRES-(30)	80.489	129.191	60.985	98.462
CFD 2	plain double GMRES-(30)	473.385	778.753	456.176	518.492
	mixed precision GMRES-(30)	273.995	510.385	256.432	301.411
CFD 3	plain double GMRES-(30)	993.638	1921.640	1145.082	1046.493
	mixed precision GMRES-(30)	554.287	1555.360	669.574	697.120

Table 6. Computation time (s) for problem CFD 1, CFD 2 and CFD 3 based on a GMRES-(30).

4 Energy efficiency

Beside the computational performance energy efficiency becomes more and more important for customers from academia and industry. We present results based on the theoretical peak power consumption of the accelerators from Tabular 1 and the runtime for our CG solver.

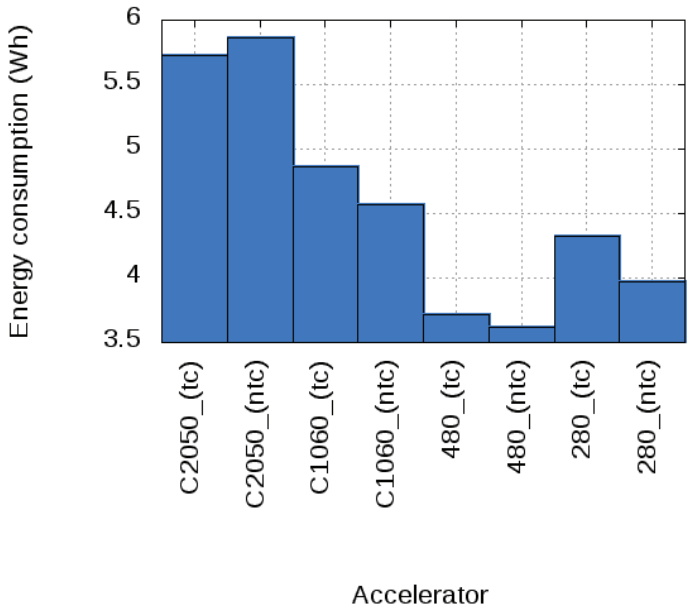


Fig. 7. Energy consumption in Watt hours (Wh) for the Laplace test case for four evaluated accelerators. *tc* employs the texture cache, *ntc* does not.

5 Conclusion

The current GPU generation offers enormous potential that can be utilized not only in constructed examples, but also CFD applications with academic and non-academic background. A basic condition for this is that the underlying mathematical model combined with the numerical schemes for solving it offers enough parallelism to create a sufficient number of threads for the GPU to cover waiting time for memory calls of one GPU-thread by executing another thread. Exchanging threads is cheap compared to the time a global memory operation takes.

The programmability of NVIDIA-GPUs was heavily simplified by the introduction of CUDA compared to the most common former approaches. Due to this, significant performance increases can be achieved in very short time e.g. by extending existing applications with accelerated kernels in Fortran and C/C++, as the partially ported meteorological implementation from [3] shows. Highest performance is achieved when porting the code completely thus avoiding host-device communication as much as possible.

The new generation of Tesla and GeForce accelerators based on the Fermi architecture offer an enormous gain in computational performance by looking on the theoretical values compared to the previous generation. For many kernels and applications based on such kernels, the performance can be used in practice. But when memory bandwidth is the limiting factor the computational power can not be exploited completely due to the fact that the memory bandwidth did not increase to the same ratio (see table 1). We see speedups of about 1.2 for the CG-algorithm from the last generation to the new one. Similar results

can be observed for the mixed precision iterative refinement solvers, though the speed-up decreases for larger dimension, as then the memory bandwidth becomes the bottleneck.

The energy efficiency tests reveal that the performance gain in terms of execution time of the new Fermi generation comes with the price of a significantly higher energy consumption.

Finally, it should be mentioned, that besides the floating point performance, this paper focuses on, the resilience of hardware is of importance in cluster computing. It should be emphasized, that the higher performance of the consumer cards comes with an uncertainty in terms of correctness of long-lasting computations. The importance of ECC is a topic of further investigation.

Acknowledgement

The authors would like to thank Dimitar Lukarski from the Shared Research Group (SRG) [9] for his assistance while performing the benchmarks and his contributions for the content of this paper.

References

1. Bolz, J., Farmer, I., Grinspun, E., Schröder,P.: Sparse matrix solvers on the GPU: conjugate gradients and multigrid. ACM Transactions on Graphics, vol. 22, 2003, pp. 917-924.
2. Krüger, J., Westermann, R.: Linear algebra operators for gpu implementation of numerical algorithms. ACM Transactions on Graphics, vol. 22, 2003, pp. 908-916.
3. Hahn, T. and Heuveline, V. and Rocker, B.: GPU-based Simulation of Particulate Flows with CUDA: Proceedings of the PARS Workshop 2009, German Informatics Society, 2009
4. Goodnight, N., Lewin, G., Luebke, D., Skadron, K.: A multigrid solver for boundary-value problems using programmable graphics hardware. Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2003, pp. 102-111.
5. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edition: SIAM: Philadelpha, PA, 2003
6. Anzt, H. and Rocker, B. and Heuveline, V.: Mixed Precision Error Correction Methods for Linear Systems: Convergence Analysis based on Krylov Subspace Methods: Proceedings of PARA 2010 State of the Art in Scientific and Parallel Computing, 2010
7. Anzt, H. and Rocker, B. and Heuveline, V.: An Error Correction Solver for Linear Systems: Evaluation of Mixed Precision Implementations: Proceedings of VECPAR 2010 High Performance Computing for Computational Science, 2010
8. Anzt, H. and Rocker, B. and Heuveline, V.: Energy efficiency of mixed precision iterative refinement methods using hybrid hardware platforms: Computer Science - Research and Development, Springer Berlin / Heidelberg, 2010
9. Shared Research Group (SRG), Karlsruhe Institute of Technology (KIT), http://www.numhpc.math.kit.edu

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2010-03 Hartwig Anzt, Vincent Heuveline, Björn Rocker: Energy Efficiency of Mixed Precision Iterative Refinement Methods using Hybrid Hardware Platforms: An Evaluation of different Solver and Hardware Configurations

No. 2010-02 Hartwig Anzt, Vincent Heuveline, Björn Rocker: Mixed Precision Error Correction Methods for Linear Systems: Convergence Analysis based on Krylov Subspace Methods

No. 2010-01 Hartwig Anzt, Vincent Heuveline, Björn Rocker: An Error Correction Solver for Linear Systems: Evaluation of Mixed Precision Implementations

No. 2009-02 Rainer Buchty, Vincent Heuveline, Wolfgang Karl, Jan-Philipp Weiß: A Survey on Hardware-aware and Heterogeneous Computing on Multicore Processors and Accelerators

No. 2009-01 Vincent Heuveline, Björn Rocker, Staffan Ronnas: Numerical Simulation on the SiCortex Supercomputer Platform: a Preliminary Evaluation