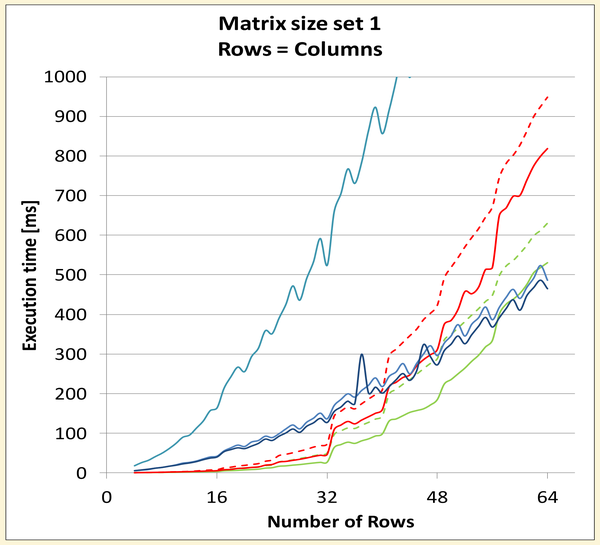
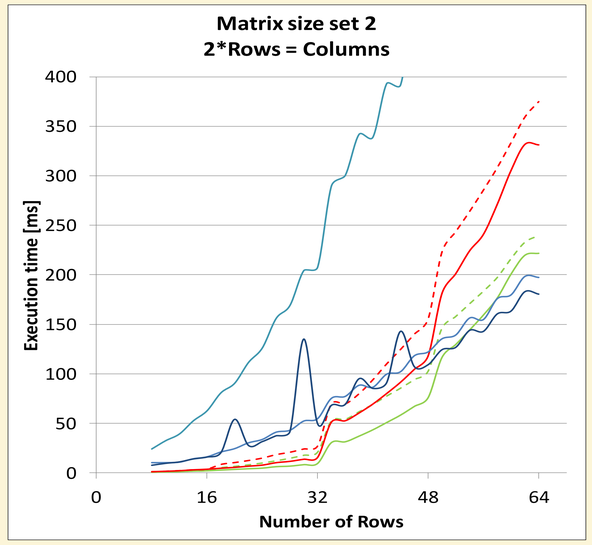
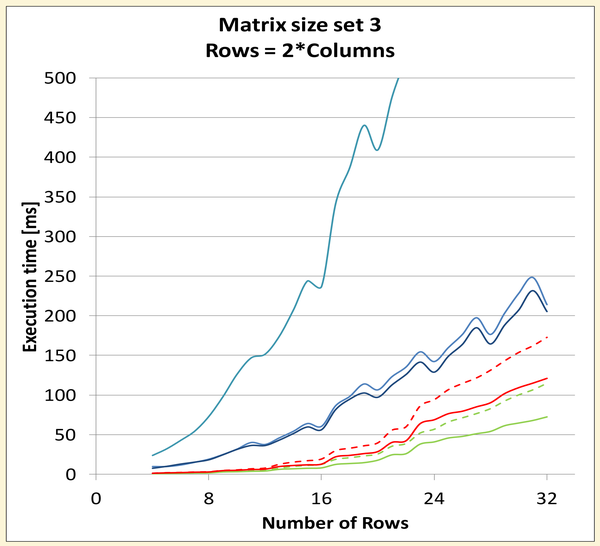
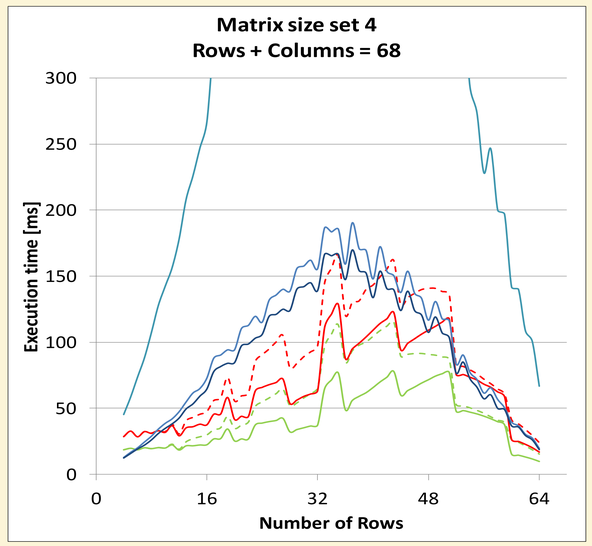
CUDA-based LU Factorization with pivoting for 10,000s of small dense matrices vs. Intel MKL

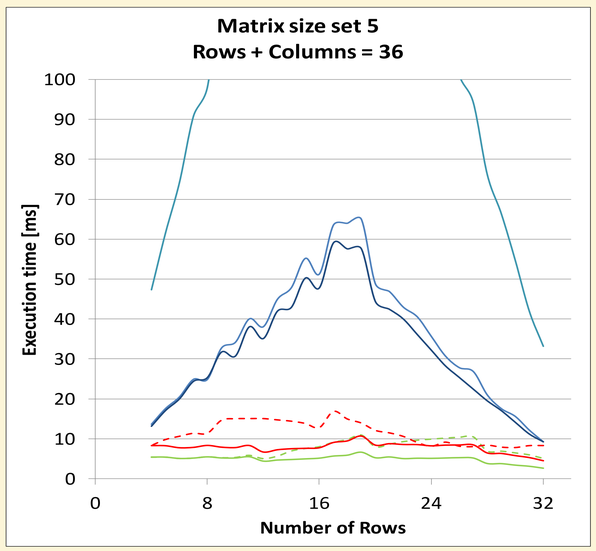
Fredrik Hellman (HPC), Jimmy Pettersson (HPC), Ian Wainwright (HPC)

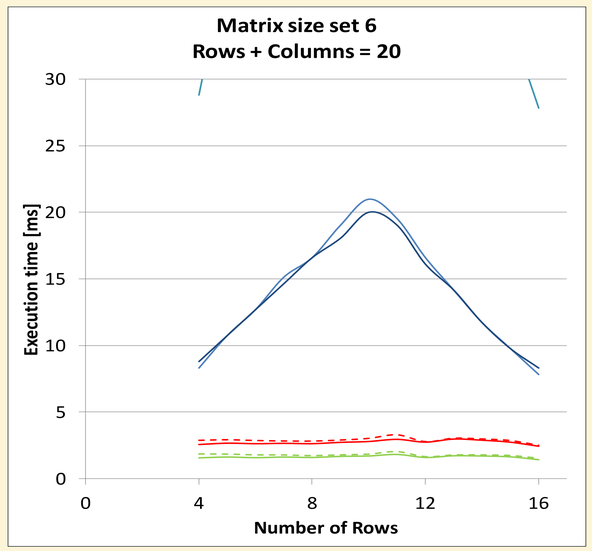












LU Factorization with pivoting

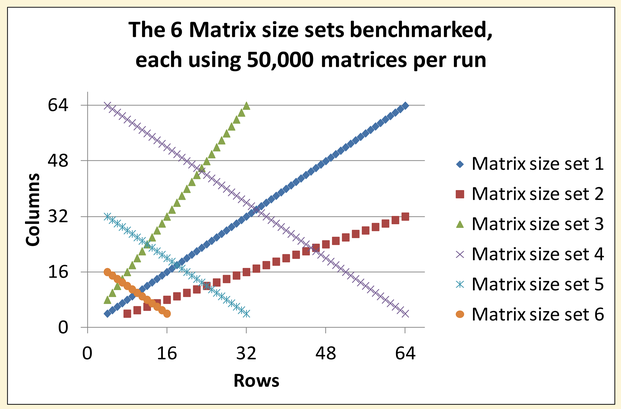
LU factorization is a ”high-level” algebraic description for Gaussian elimination. It is a fundamental operation performed in linear algebra when for example solving systems of linear equations, inverting a matrix or computing its determinant.

Implementation

The CUDA-based implementation stores almost all data in registers, each thread servicing one column. This minimizes the need for memory-access and allows for high performance until registers start to spill into memory. To fully utilize the CPU, function calls to Intel MKL have been parallelized using OpenMP and optimal thread affinity.

Conclusions

- GPUs are more than capable of performing dense linear algebra on small matrices, **achieving a speed-up factor of more than 10** vs. an 8-threaded Intel MKL implementation on a high-end CPU.
- When register spills occur, a discrete loss in performance often follows.
- Selecting 48 KiB L1 Cache instead of Shared memory helps minimize the impact of excessive register spills.



Non-GPU Hardware	Software
Intel Core i7 960, 3.20 GHz, 8 Logical Cores	Windows 7 64-bit SP1
ASUS P6T Deluxe V2	CUDA 286.19 Driver
12 GB RAM	CUDA Toolkit 4.1

GPUs	GTX 480	GTX 560 Ti
Core count	480	384
Core clock	1.40 GHz	1.64 GHz
Max Theoretical Compute Power	1344 GFLOPS	1259 GFLOPS
Memory Bus-Width	384-bit	256-bit
Memory clock	1.848 GHz	2.00 GHz
Max Bandwidth with SDK Bandwidth test	146.5 GB/s	105.8 GB/s