Accelerating Symmetric Matrix-Vector Product on Fermi GPU

Ahmad Abdelfettah[1], Jack Dongarra[2], David Keyes[1] and HatemLtaief[3]
Division of Mathematical and Computer Sciences and Engineering, King Abdullah University of Science and Technology, Kingdom of Saudi Arabia[1]
Innovative Computing Laboratory, University of Tennessee, Knoxville USA[2]
Supercomputing Laboratory, King Abdullah University of Science and Technology, Kingdom of Saudi Arabia[3]

Abstract

Hardware accelerators are becoming ubiquitous in high performance scientific computing. Although challenging to program, GPU-based systems occupy three of the top five fastest machines in the world and more than thirty systems of the last TOP500 list contains accelerators. A principal explanation to this new trend is the capability of GPUs to deliver an unprecedented amount of **concurrent execution contexts**. High-level programming languages (e.g., CUDA), profiling tools (e.g., PAPI-CUDA, CUDA Profiler)as well as auto tuning frameworks are paramount to improve **productivity**, while **effectively exploiting the underlying hardware**.

We aim here to describe an optimized numerical kernel computing the symmetric matrix-vector product (Level 2 BLAS) on the latest nVidia TESLA GPU family, codenamed **Fermi**. Due to its inherent memory-bound nature, this kernel represents **one of the most critical operations** in computing the tridiagonal form of a symmetric dense matrix, which is a preprocessing step toward calculating the eigenpairs. Using a novel design to address **the irregular memory accesses** by hiding latency and increasing bandwidth, our preliminary asymptotic results show **3.5 and 2.5 fold** speedups over the similar **CUBLAS 4.0** kernel, and **7-8% and 30% fold** improvement over the Matrix Algebra on GPU and Multicore Architectures (**MAGMA**) library in single and double precision arithmetics, respectively.

Rules of Thumb

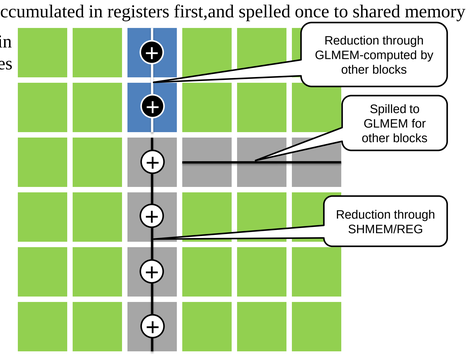
- Load balancing between computational blocks
- Data caching for reused data
- Data prefetching (to hide memory latency)
- Avoid going to SLOW global memory as much as possible
- Memory coalesced access (per warp)
- Avoid shared memory bank conflicts
- Avoid divergent branches (within same warp)
- Avoid spilling registers to local memory (63 registers/thread max.)
- Wisely use SM resources to increase occupancy

Profiling Tools

- PAPI CUDA Component to extract performance counters during kernel execution e.g.,shared memory bank conflict, global memory misses (load/store), divergent branches, local memory usage.
- NVIDIA compute profiler, additional information unavailable/hard to get through PAPI CUDA component e.g., registers per thread, GPU time, occupancy analysis, Kernel memory bandwidth.

Core Design

- 64×64 block size.
- Symmetric diagonal blocks are isolated from non-diagonal ones.
- Each computation block is responsible for one vertical column of submatrices, offering better use of locality for column major format.
- Partial contributions are accumulated in registers first,and spelled once to shared memory for reduction.
- Double buffering is used in all global memory accesses to hide the latency.



Motivations

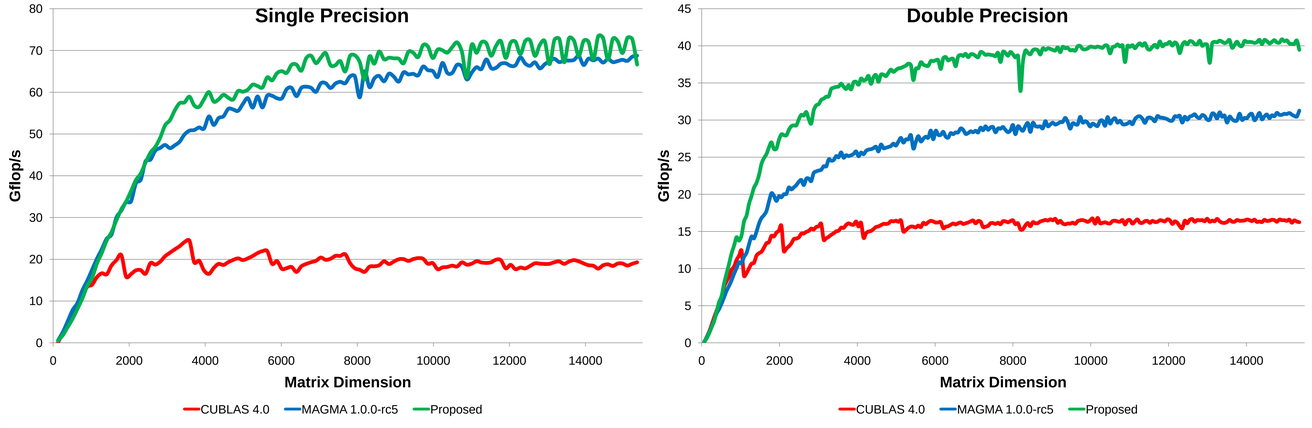
A Look at the last TOP500 List

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 / 2010 NUDT	186368	2566.00	4701.00	4040.0
3	DOE/SC/Oak Ridge National Laboratory United States	Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.0
4	National Supercomputing Centre in Shenzhen (NSCS) China	Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 / 2010 Dawning	120640	1271.00	2984.30	2580.0
5	GSIC Center, Tokyo Institute of Technology Japan	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.6

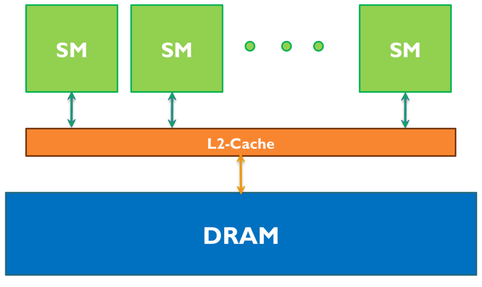
Test Case: Tridiagonal Reduction

- Calculating the eigenpairs of a dense symmetric matrix A
- Requires the reduction to tridiagonal form, which extensively involves symmetric Matrix-Vector products
- Kernel consists in computing a Level 2 BLAS operation
  $Y = \alpha \times A \times X + \beta \times Y$
  A is a symmetric matrix
  X and Y are vectors
  αand β are scalars
  Only lower/upper side of A should be referenced
- Data reuse possible only in the vector X (read mode)

Performance Results

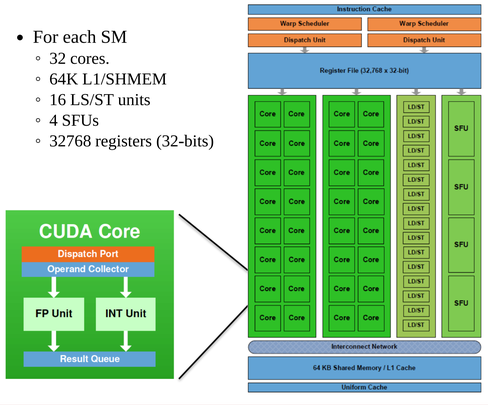


Fermi Hardware Specifications



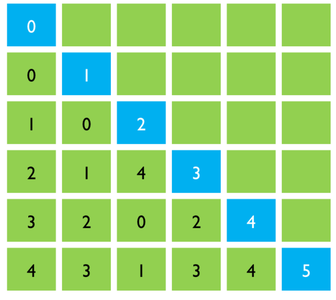
- Complete Memory Hierarchy(registers, L1 cache/SHMEM,L2 cache, DRAM)with ECC support
- Theoretical Peak Performance:
  - 1 Tflop/s (SP) and
  - ~ 500 Gflop/s (DP)

- For each SM
  - 32 cores.
  - 64K L1/SHMEM
  - 16 LS/ST units
  - 4 SFUs
  - 32768 registers (32-bits)



Future Works

- Integration into MAGMA/nVidia CUBLAS libraries
- Distribution of work among computation blocks is not balanced.Balancing load may lead to further improvement, butlocality will notbe exploited.1D Block cyclic assignment is intended:



References

- nVidiaCUBLAS Library. http://developer.download.nvidia.com.
- R. Nath, S. Tomov, T. Dong and J. Dongarra, *"Optimizing Symmetric Dense Matrix-Vector Multiplication on GPUs"*, International Supercomputing Conference, Seattle, 2011.
- Matrix Algebra on GPU and MulticoreArchitectures (MAGMA) http://icl.cs.utk.edu/magma/.
- Performance Application Programming Interface (PAPI).http://icl.cs.utk.edu/papi/.

Acknowledgements

- RajibNath (University of California, San Diego, USA)
- Heike Jagode(University of Tennessee, Knoxville, USA)
- Michael Young (KAUST IT)
- nVidia for the hardware donations