Berkeley's Dwarfs on CUDA

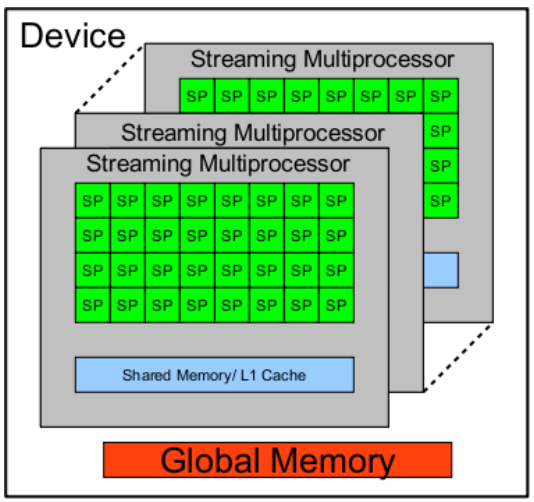
Paul Springer

23.07.2011

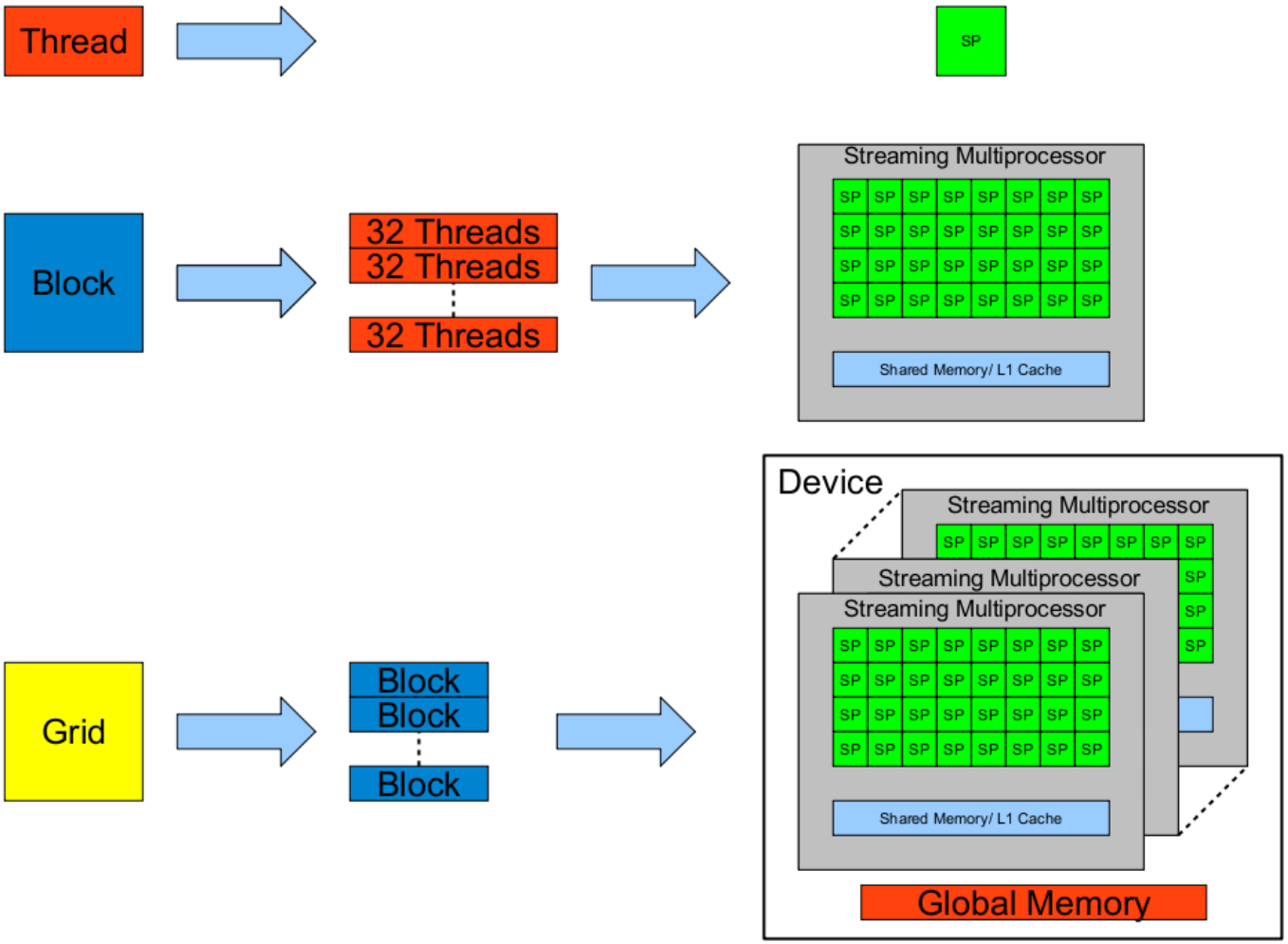
Contents

Modern Graphics Cards

- Peak single-precision performance of up to 1331 GFLOP/s
- Peak double-precision performance of up to 665 GFLOP/s
- Peak global-memory bandwidth of up to 177 GB/s
- Increasing general purpose capabilities
  - ECC support
  - Improved double precision performance
  - Improved debugging capabilities
- 512 cores (scalar processors SP)



Execution Model



Berkeley's Dwarfs

Dwarfs

Definition

A **Dwarf** is an algorithmic method that captures a pattern of computations and communication [1].

- Dwarfs are believed to be important for future scientific applications
- An application can comprise several Dwarfs
- Guide the development of new architectures and programming models

Dwarfs

Original Dwarfs

- Dense Linear Algebra
- Sparse Linear Algebra
- Spectral Methods
- N-Body Methods
- Structured Grids
- Unstructured Grids
- Monte Carlo

Additional Dwarfs

- Combinational Logic
- Graph Traversal
- Dynamic Programming
- Back-track and Branch & Bound
- Graphical Models
- Finite State Machine

GPU Benchmark Suites

Rodinia

- Open Source
- 20 applications covering five Dwarfs
- Designed for CUDA and OpenMP
- Few OpenCL versions

Parboil

- Open Source
- 12 applications covering six Dwarfs
- CUDA implementation only

SHOC

- Freely available but not open source
- 16 applications covering five Dwarfs
- Auto-parallelizes across multiple GPUs and nodes
- Targets CUDA and OpenCL alike

GPU Dwarfs

Dense Linear Algebra

- Many available applications
- Available libraries
  - CUBLAS: NVIDIA's implementation of BLAS
  - CULA: Commercial implementation of LAPACK
  - MAGMA: open source LAPACK implementation
- Typical problem
  - CUDA lacks an inter-block synchronization [13], [12]
- Good fit for the GPU

Dense Linear Algebra

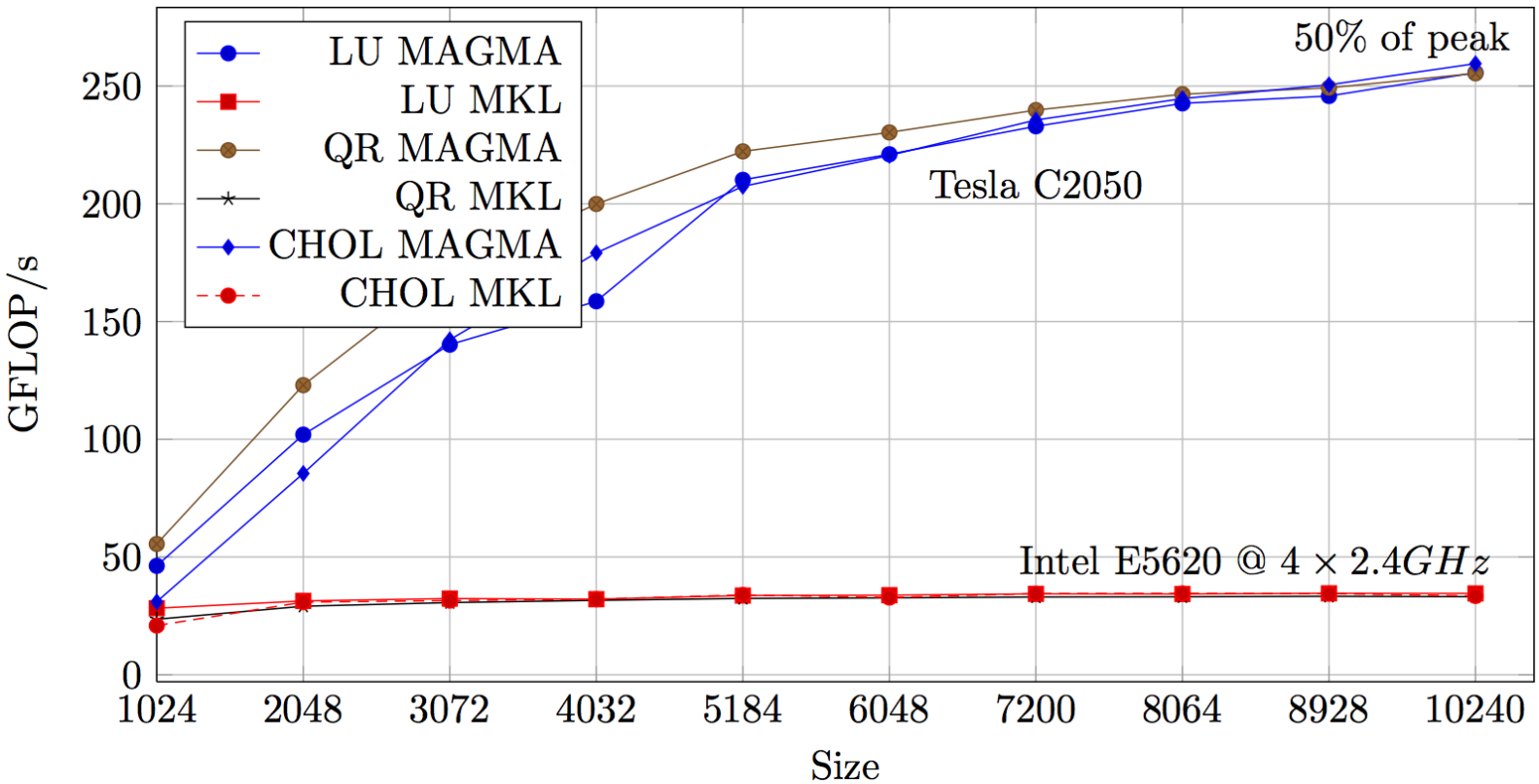


Figure : Performance of LU, Cholesky and QR decomposition of a square matrix using double precision.

Sparse Linear Algebra

- Fewer applications available than for DLA
- Typical problem [2]:
  - Indirect and irregular memory accesses
  - Few floating point operations per memory access
  - Data reuse
- Available libraries
  - CUSPARSE: NVIDIA's implementation of sparse BLAS
  - CUSP: open source implementation of sparse BLAS, includes iterative solvers based on Krylov subspaces

Sparse Linear Algebra

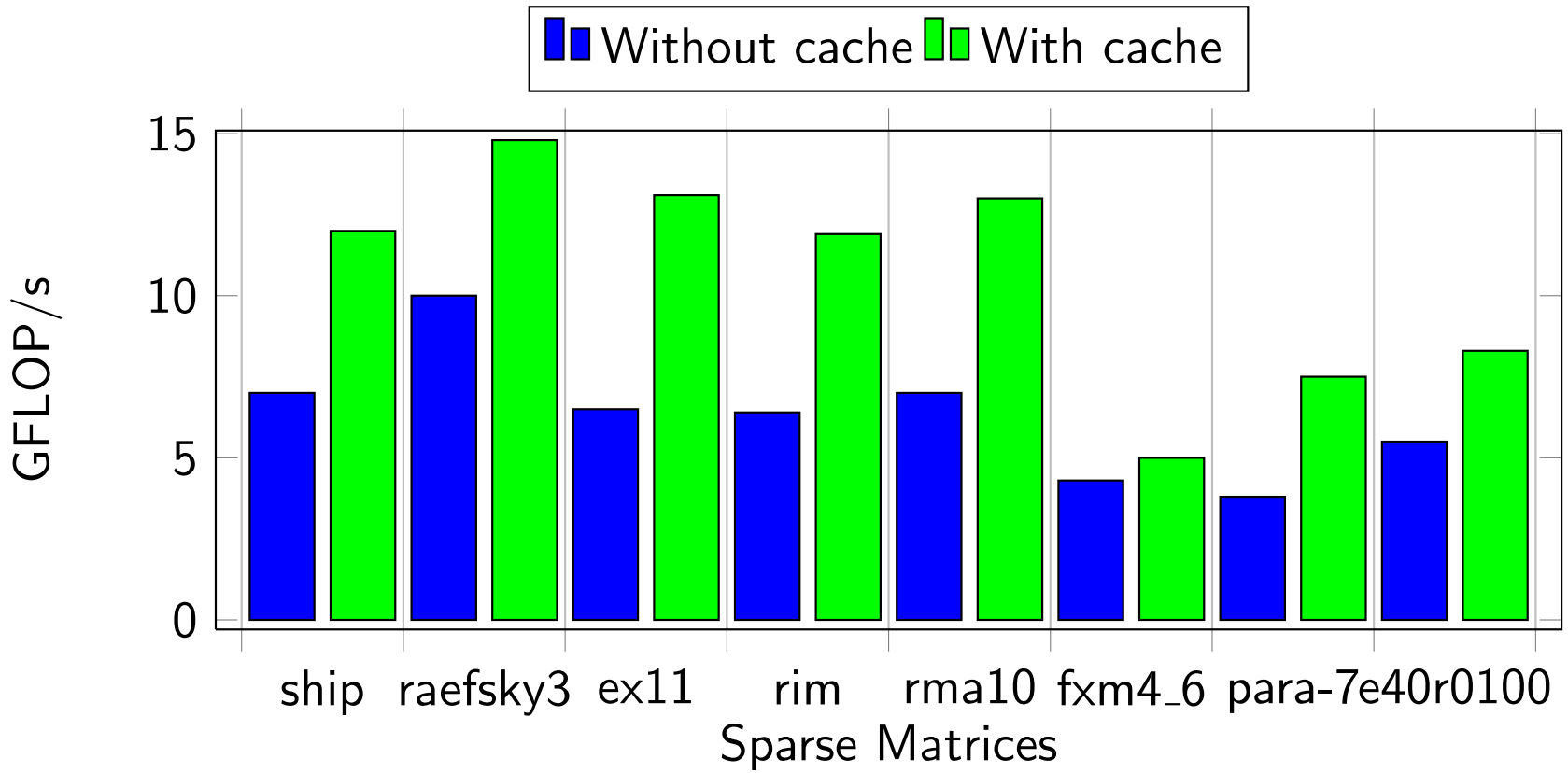


Figure : SpMV Performance shown by Baskaran et. al [2] running on a GeForce GTX 280.

N-Body Methods

Description

A system of N bodies which induce forces upon each other.

- Folding@home: Biophysical simulations such as protein folding [3]
  - Distributed application
  - GPUs contribute over 65% to the total peak performance
- Direct Coulomb Summation: Calculating electrostatic potential maps
  - Approach by Kerk et al. [8, p. 173f] is based on a regular grid
  - Achieve speedups of 40x

N-Body Methods

- Astrophysical simulations: Movement of stars and galaxies [11, p.677f]
  - All-pair approach yielding $\mathcal{O}(N^2)$ complexity
  - Achieve up to 60% of peak performance on a GTX 8800.
- Tree Based Barnes Hut (TBBH)
  - Subdivides the space into small cells [7, p.75f]
  - Approximation algorithm, that reduces the complexity to $\mathcal{O}(NlogN)$
  - Involved problems
    - Building and traversing an irregular data structure
    - Cope with indirect memory accesses
    - Avoiding recursion

Tree Based Barnes Hut (TBBH)

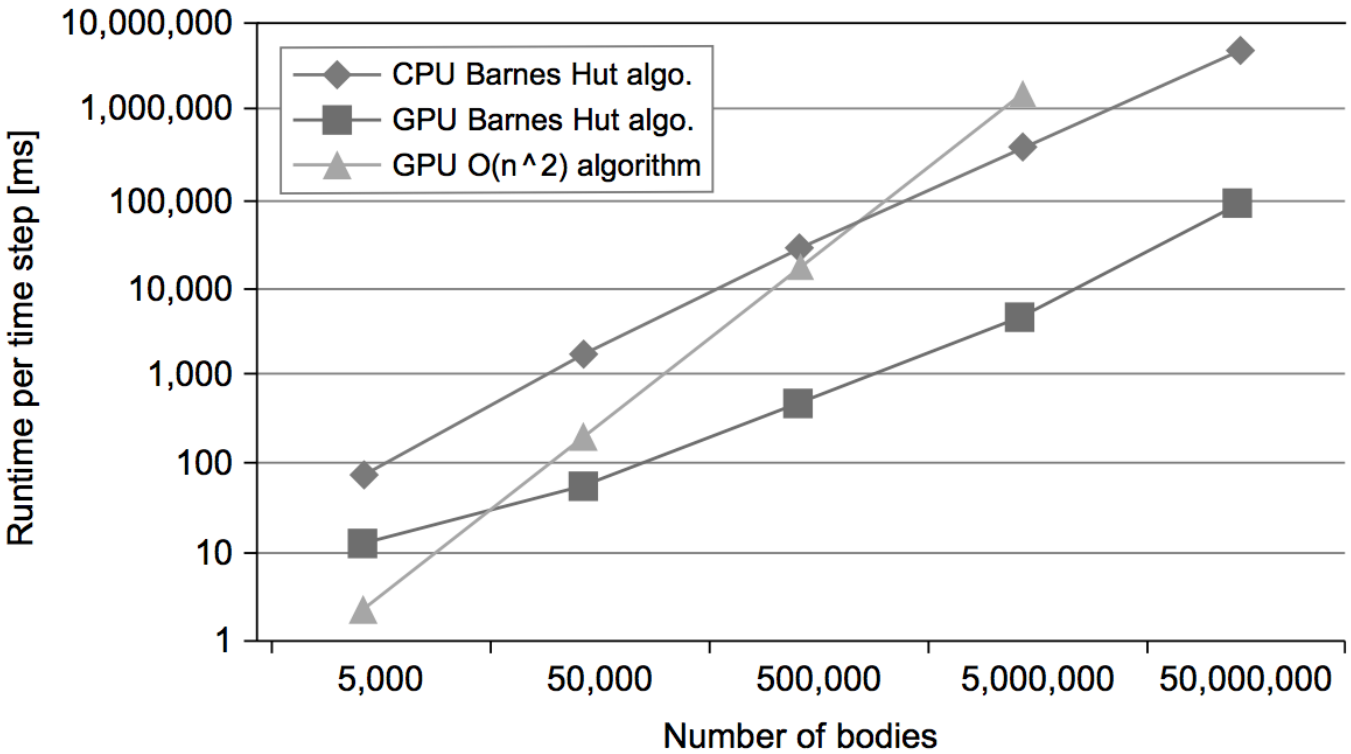


Figure : **TBBH**: Runtime for a singe time step in milliseconds w.r.t. varying input sizes. Running on NVIDIA Quadro FX 5800 and a 2.53GHz Intel Xeon E5540 respectively. [7, p.90]

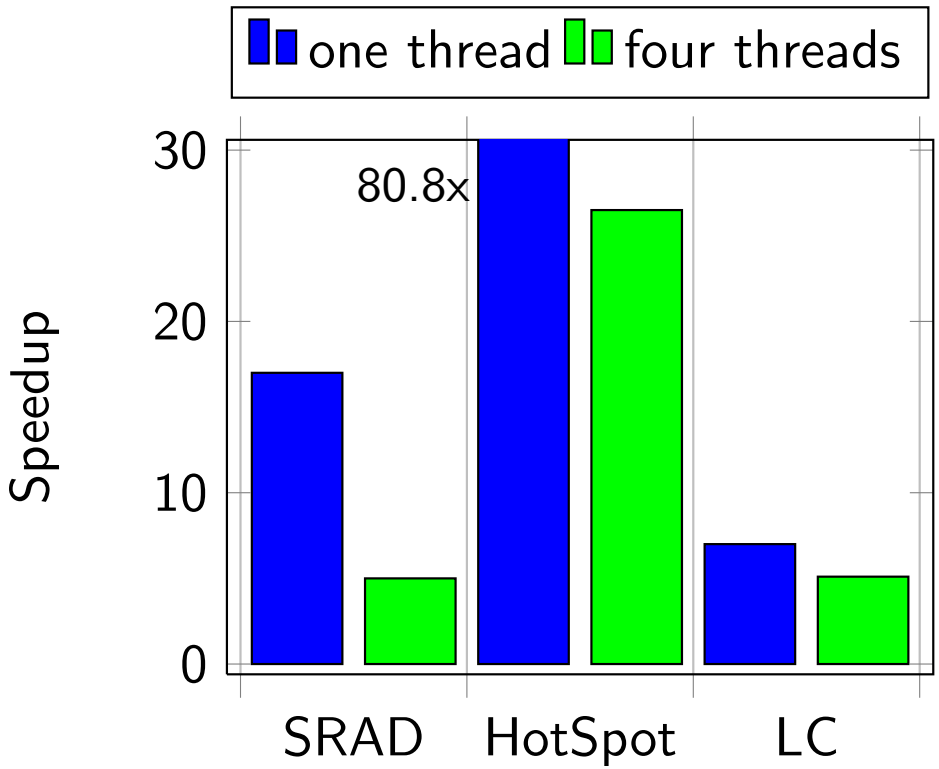
Structured Grids

Description

Structured grid algorithms subdivide the computational region into subspaces where updating an element depends on its neighboring elements.

- Speckle Reducing Anisotropic Diffusion (SRAD)
  - Image processing application for ultrasonic and radar images
  - Reduces noise while maintaining important image features
- HotSpot (HS)
  - Thermal simulation
  - Estimates the temperature of a processor
- Leukocyte tracking (LC) [4]
  - Detects and tracks leukocytes in intravital microscopy videos

Structured Grids



- Typical problems
  - Handle boundary elements [10]
  - Avoid global synchronizations

Figure : Speedup of SRAD, HotSpot and LC running on a GeForce GTX 280 over an optimized OpenMP implementation [5].

Graph Traversal

- All Pairs Shortest Path (up to **17x**) [6]
- Single Source Shortest Path (up to **70x**) [6]
- Maximum Flow (up to **30x**) [7, p.439f]
- Breadth First Search (up to **10x**) [9]
- Typical problems
  - Irregular data structures and memory accesses
  - Poor spatial locality
  - Avoid recursion
  - Avoid global synchronizations

Breadth First Search

Graph type	GPU1-BFS	GPU2-BFS
Regular	0.45x	10.3x
Real World	0.51x	5.6x
Scale-free	0.25x	1.05x

Table : **Speedup** of two different GPU BFS implementations over the best CPU algorihm. GPU1-BFS denote the GPU implementation of Harish et al. [6] and GPU2-BFS denotes GPU implementation of Luo et al[9]. Results taken from [9].

Conclusion

- CUDA is a good fit for many Dwarfs
- But it is more difficult to achieve high-performance
- Some speedups shown in research papers are not representative
- Future work
  - More reliable speedups
  - Metrics which account for the effort of a GPU implementation
  - Benchmark suites should comprise all 13 Dwarfs

Thank you for your attention.

References I


K. Asanovic, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, et al.
The landscape of parallel computing research: A view from berkeley.
Technical report, Citeseer, 2006.


M.M. Baskaran and R. Bordawekar.
Optimizing sparse matrix-vector multiplication on gpus.
IBM research report RC24704, IBM, 2009.


A.L. Beberg, D.L. Ensign, G. Jayachandran, S. Khaliq, and V.S. Pande.
Folding@ home: Lessons from eight years of volunteer distributed computing.
2009.

References II


M. Boyer, D. Tarjan, S.T. Acton, and K. Skadron.
Accelerating leukocyte tracking using cuda: A case study in leveraging manycore coprocessors.
2009.


S. Che, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, S.H. Lee, and K. Skadron.
Rodinia: A benchmark suite for heterogeneous computing.
In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pages 44–54. IEEE, 2009.


P. Harish and P. Narayanan.
Accelerating large graph algorithms on the gpu using cuda.
High Performance Computing–HiPC 2007, pages 197–208, 2007.


Wen-Mei W. Hwu.
GPU Computing Gems (Applications of Gpu Computing).
Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2011.

References III


D.B. Kirk and W.H. Wen-mei.
Programming massively parallel processors: A Hands-on approach.
Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2010.


L. Luo, M. Wong, and W. Hwu.
An effective gpu implementation of breadth-first search.
In *Proceedings of the 47th Design Automation Conference*, pages 52–55. ACM, 2010.


J. Meng and K. Skadron.
Performance modeling and automatic ghost zone optimization for iterative stencil loops on gpus.
In *Proceedings of the 23rd international conference on Supercomputing*, pages 256–265. ACM, 2009.

References IV


NVIDIA.
GPU Gems3: Programming Techniques for High-Performance Graphics and General-Purpose Computation.
Addison-Wesley, 2007.


V. Volkov and J.W. Demmel.
Benchmarking gpus to tune dense linear algebra.
In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–11. IEEE, 2008.


S. Xiao and W. Feng.
Inter-block gpu communication via fast barrier synchronization.
In *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–12. IEEE.