Scalability, Portability, and Numerical Stability in Many-core Parallel Libraries

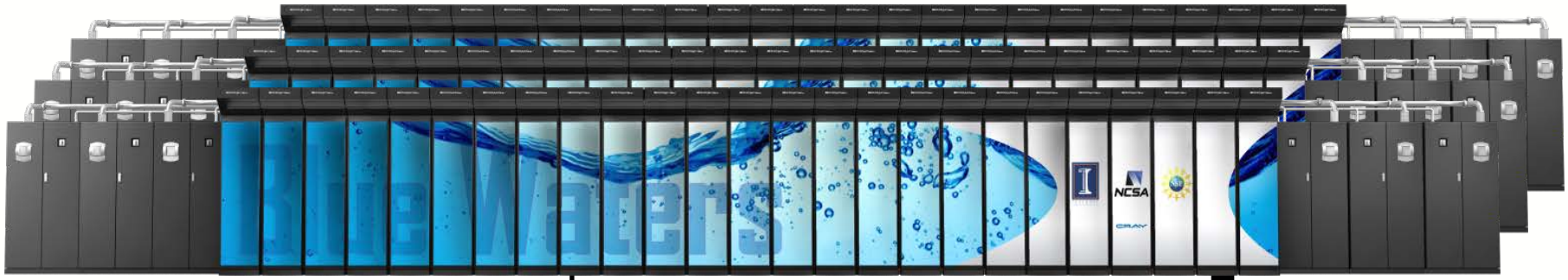
Wen-mei Hwu
University of Illinois, Urbana-Champaign

Agenda

- Many-core GPUs in Blue Waters
- Programming Interfaces
- Library Algorithms
  - Scalability, Performance, and numerical stability
- Some Important Trends
- Conclusion and Outlook

Blue Waters Computing System

Operational at Illinois since 11/2012



11.1 PF
1.5 PB DRAM

10/40/100 Gb Ethernet Switch

IB Switch

>1 TB/sec

120+ Gb/sec

100 GB/sec

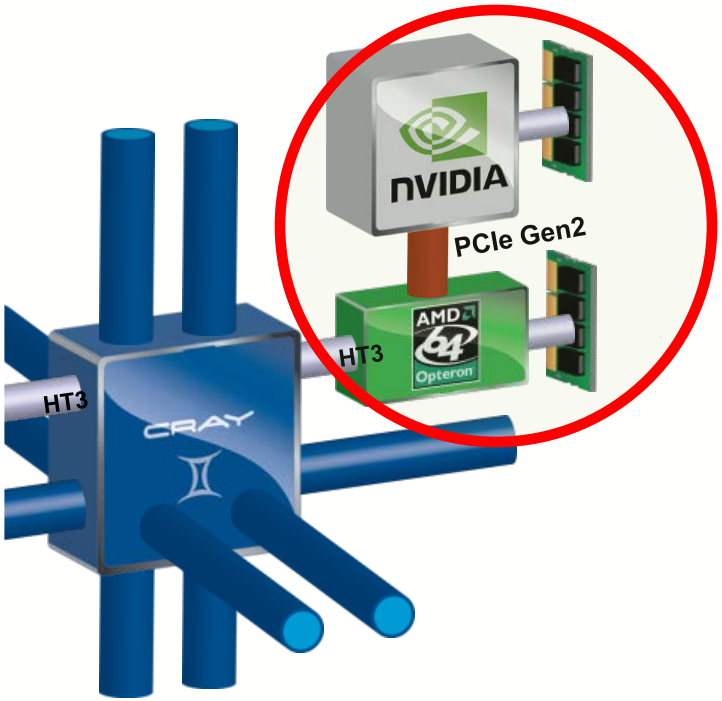
WAN

Spectra Logic: 300 PBs

Sonexion: 26 PBs

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

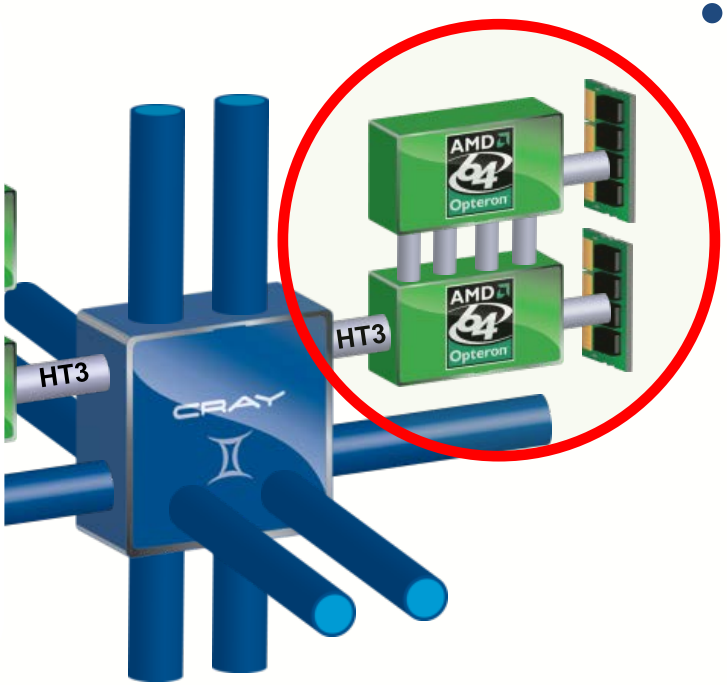
Cray XK7 Nodes



- Dual-socket Node
  - One AMD Interlagos chip
    - 8 core modules, 32 threads
    - 156.5 GFs peak performance
    - 32 GBs memory
      - 51 GB/s bandwidth
  - One NVIDIA Kepler chip
    - 1.3 TFs peak performance
    - 6 GBs GDDR5 memory
      - 250 GB/sec bandwidth
  - Gemini Interconnect
    - Same as XE6 nodes

Blue Waters contains 3,072 Cray XK7 compute nodes.

Cray XE6 Nodes



- Dual-socket Node
  - Two AMD Interlagos chips
    - 16 core modules, 64 threads
    - 313 GFs peak performance
    - 64 GBs memory
      - 102 GB/sec memory bandwidth
  - Gemini Interconnect
    - Router chip & network interface
    - Injection Bandwidth (peak)
      - 9.6 GB/sec per direction

Blue Waters contains 22,640 Cray XE6 compute nodes.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Blue Waters and Titan Computing Systems

System Attribute	NCSA Blue Waters	ORNL Titan
Vendors	Cray/AMD/NVIDIA	Cray/AMD/NVIDIA
Processors	Interlagos/Kepler	Interlagos/Kepler
Total Peak Performance (PF)	11.1	27.1
Total Peak Performance (CPU/GPU)	7.1/4	2.6/24.5
Number of CPU Chips	48,352	18,688
Number of GPU Chips	3,072	18,688
Amount of CPU Memory (TB)	1511	584
Interconnect	3D Torus	3D Torus
Amount of On-line Disk Storage (PB)	26	13.6
Sustained Disk Transfer (TB/sec)	>1	0.4-0.7
Amount of Archival Storage	300	15-30
Sustained Tape Transfer (GB/sec)	100	7

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Why did we have only 3072 GPUs in Blue Waters?

- Blue Waters will be the only Petascale machine for the NSF community for at least two years
  - Must minimize risk for petasacle application teams

- The NSF review panel was very concerned about the usability of GPUs in 2011
  - Small DRAM – up to 6GB
  - Hard to program for application teams
  - Lack of at-scale experience
  - Lack of begin-to-end production use experience

APPLICATIONS

At Scale, Begin-to-end Production Use

Science Area	Number of Teams	Codes	Struct Grids	Unstruct Grids	Dense Matrix	Sparse Matrix	N-Body	Monte Carlo	FFT	PIC	Sig I/O
Climate and Weather	3	CESM, GCRM, CM1/WRF, HOMME	X	X		X		X			X
Plasmas/ Magnetosphere	2	H3D(M),VPIC, OSIRIS, Magtail/UPIC	X				X		X		X
Stellar Atmospheres and Supernovae	5	PPM, MAESTRO, CASTRO, SEDONA, ChaNGa, MS-FLUKSS	X			X	X	X		X	X
Cosmology	2	Enzo, pGADGET	X			X	X				
Combustion/ Turbulence	2	PSDNS, DISTUF	X						X		
General Relativity	2	Cactus, Harm3D, LazEV	X			X					
Molecular Dynamics	4	AMBER, Gromacs, NAMD, LAMMPS				X	X		X		
Quantum Chemistry	2	SIAL, GAMESS, NWChem			X	X	X	X			X
Material Science	3	NEMOS, OMEN, GW, QMCPACK			X	X	X	X			
Earthquakes/ Seismology	2	AWP-ODC, HERCULES, PLSQR, SPECFEM3D	X	X			X				X
Quantum Chromo Dynamics	1	Chroma, MILC, USQCD	X		X	X					
Social Networks	1	EPISIMDEMICS									
Evolution	1	Eve									
Engineering/System of Systems	1	GRIPS,Revisit						X			
Computer Science	1			X	X	X			X		X

Current Science Team GPU Plans and Results

- ***About 1/3 of PRAC projects have active GPU efforts, including***
  - AMBER
  - LAMMPS
  - USQCD/**Chroma**/MILC
  - GAMESS
  - **NAMD**
  - **QMCPACK**
  - PLSQR/SPECFEM3D
  - PHOTONPLASMA
  - AWP-ODC
- Others are investigating use of GPUs (e.g., Cactus, PPM,, MS-FLUKSS)

Initial Production Use Results

- NAMD
  - 100 million atom benchmark with Langevin dynamics and PME once every 4 steps, from launch to finish, all I/O included
  - 768 nodes, Kepler+Interlagos is 3.9X faster over Interlagos-only
  - 768 nodes, XK7 is 1.8X XE6
- Chroma
  - Lattice QCD parameters: grid size of 48^3 x 512 running at the physical values of the quark masses
  - 768 nodes, Kepler+Interlagos is 4.9X faster over Interlagos-only
  - 768 nodes, XK7 is 2.4X XE6
- QMCPACK
  - Full run Graphite 4x4x1 (256 electrons), QMC followed by VMC
  - 700 nodes, Kepler+Interlagos is 4.9X faster over Interlagos-only
  - 700 nodes, XK7 is 2.7X XE6

PROGRAMMING INTERFACES

and Tools

Levels of GPU Programming Languages

Prototype & in development — X10, Chapel, Nesl, Delite, Par4all...

Implementation manages GPU threading and synchronization invisibly to user

Next generation — OpenACC, C++AMP, Thrust, Bolt

Simplifies data movement and kernel launch

Same GPU execution model (but less boilerplate)

Current generation — CUDA, OpenCL, DirectCompute

Performance is currently hard to predict for high-level tools.

- High-level tools are making strides in usability, generality, and performance
- Performance sometimes comparable to low-level tools
- ...sometimes much worse

Nested data-parallel code vs. CUDA running on a GPU

	Code size (lines)		Run time (ms)		
	CUDA	Nesl	CUDA	Nesl	
Convex Hull	72	25	269	283	*On par*
Barnes-Hut	414	225	40	4502	*100 × slower*

Source: Bergstrom and Reppy, ICFP 2012

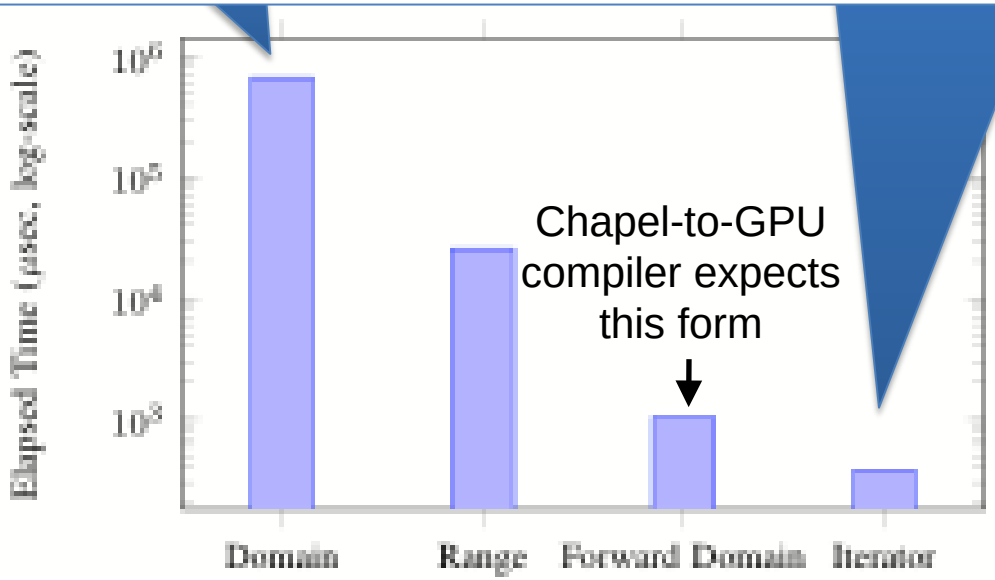
Huge overhead is often caused by the generality of the interface.

- Many performance pitfalls are fixable, but will still cause problems for novices

```
for j in [1..N] do ...;
```

```
iter myIter(min:int, max:int, step:int=1) {
  while min <= max {
    yield min;
    min += step;
}}
for j in myIter(1,N) do ...;
```

Time to execute a one-iteration loop on CPU in Chapel



Source: Dun and Taura, IPDPSW 2012

Writing efficient parallel code is complicated.

Tools can provide focused help

or broad help

Planning how to execute an algorithm

- Choose data structures

Implementing the plan

- Memory allocation
- Data movement

GMAC

- Pointer operations
- Index arithmetic

DL

Triolet, X10, Chappel, Nesl, DeLite, Par4All

- Decompose work into tasks
- Schedule tasks to threads

MxPA/MCUDA

- Kernel dimensions
- Thread ID arithmetic

Thread coarsening

- Synchronization
- Temporary data structures

LIBRARY ALGORITHMS

Scalable GPU Libraries

- Dense Linear algebra—BLAS, LU, Cholesky, Eigen solvers (CUBLAS, CULA, MAGMA)
- Sparse Matrix Vector Multiplication, Tridiagonal solvers (CUSPARSE, QUDA, ViennaCL, Parboil)
- FFTs, Convolutions (CUFFT, ViennaCL, Parboil)
- N-Body (NAMD/VMD, FMM BU, Parboil)
- Histograms (Parboil)
- Some PDE solvers (CURRENT, Parboil)
- Graphs – Breadth-First Search (Parboil)

Example of Library Needs

- Sparse linear algebra
  - Sparse LU, Cholesky factorization(?)
  - Sparse Eigen solvers
- Graph algorithm
  - Graph partitioning
  - Depth first search
  - …
- …

An Opportunity of a Lifetime

- Scalable and portable software lasts through many hardware generations

Scalable algorithms and libraries could be the best legacy we can leave behind from this era

Algorithm Design Challenges

Parallelism

- Parallelism to fill growing HW parallelism

Data Scalability

- Operations should grow linearly with data size

Locality

- DRAM burst and cache space utilization

Regularity

- SIMD utilization and load balance

Numerical Stability

- Pivoting for linear system solvers

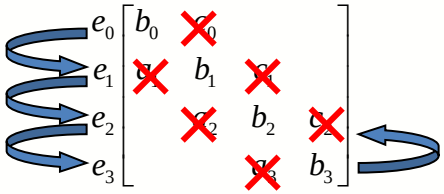
Example: Tridiagonal Solver

- Implicit finite difference methods, cubic spline interpolation, preconditioners
- An algorithm to find a solution of ***Ax = d***, where *A* is an *n*-by-*n* tridiagonal matrix and *d* is an *n*-element vector

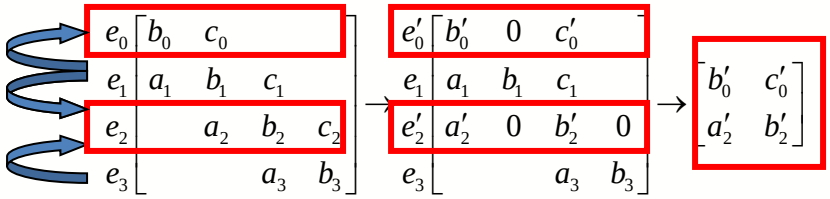
$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \quad d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

GPU Tridiagonal System Solver Case Study

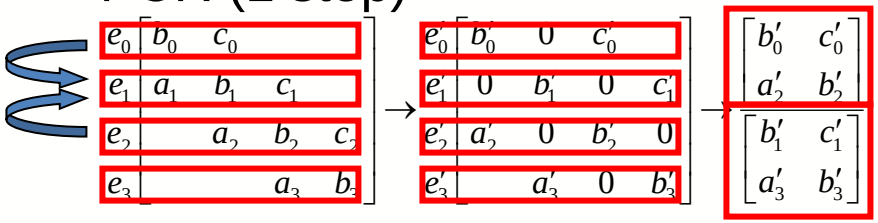
- Thomas (sequential)



- Cyclic Reduction (1 step)



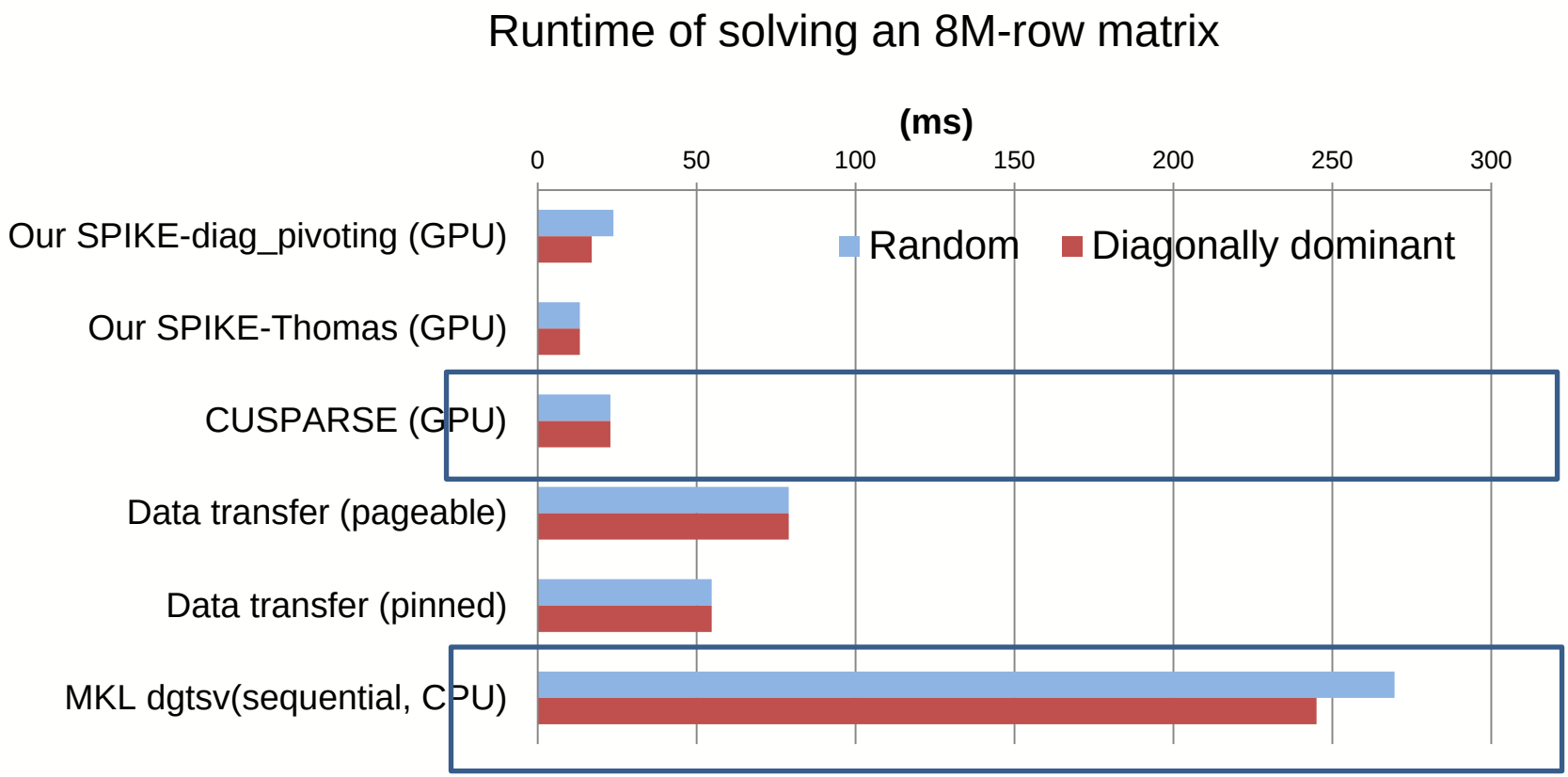
- PCR (1 step)



- Hybrid Methods
  - PCR-Thomas (Kim 2011, Davidson 2011)
  - PCR-CR (CUSPARSE 2012)
  - Etc.
- CUSPARSE is supported by NVIDIA

Interleaved layout after partitioning

GPU Performance Advantage



Numerical Error and Stability

Relative Backward Error

Matrix type	SPIKE-diag_pivoting	SPIKE-Thoma	CUSPARSE	MKL	Intel SPIKE	Matlab
1	1.82E-14	1.97E-14	7.14E-12	1.88E-14	1.39E-15	1.96E-14
2	1.27E-16	1.27E-16	1.69E-16	1.03E-16	1.02E-16	1.03E-16
3	1.55E-16	1.52E-16	2.57E-16	1.35E-16	1.29E-16	1.35E-16
4	1.37E-14	1.22E-14	1.39E-12	3.10E-15	1.69E-15	2.78E-15
5	1.07E-14	1.13E-14	1.82E-14	1.56E-14	4.62E-15	2.93E-14
6	1.05E-16	1.06E-16	1.57E-16	9.34E-17	9.51E-17	9.34E-17
7	2.42E-16	2.46E-16	5.13E-16	2.52E-16	2.55E-16	2.27E-16
8	2.14E-04	2.14E-04	1.50E+10	3.76E-04	2.32E-16	2.14E-04
9	2.32E-05	3.90E-04	1.93E+08	3.15E-05	9.07E-16	1.19E-05
10	4.27E-05	4.83E-05	2.74E+05	3.21E-05	4.72E-16	3.21E-05
11	7.52E-04	6.59E-02	4.54E+11	2.99E-04	2.20E-15	2.28E-04
12	5.58E-05	7.95E-05	5.55E-04	2.24E-05	5.52E-05	2.24E-05
13	5.51E-01	5.45E-01	1.12E+16	3.34E-01	3.92E-15	3.08E-01
14	2.86E+49	4.49E+49	2.92E+51	1.77E+48	3.86E+54	1.77E+48
15	2.09E+60	Nan	Nan	1.47E+59	Fail	3.69E+58
16	Inf	Nan	Nan	Inf	Fail	4.68E+171

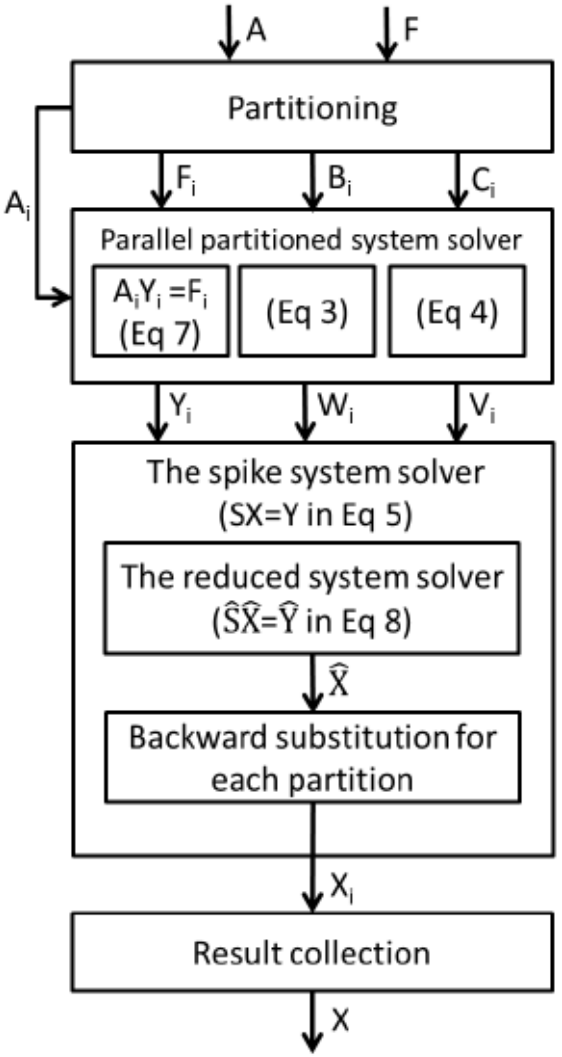
Numerically Stable Partitioning Algorithm

- SPIKE (Polizzi et al), diagonal pivoting for numerical stability

$$A=\begin{bmatrix} A_1 & B_1 & & \\ C_2 & A_2 & B_2 & \\ & C_3 & A_3 & B_3 \\ & & C_4 & A_4 \end{bmatrix}, \quad X=\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}, \quad F=\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}.$$

$$D=\begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & A_3 & \\ & & & A_4 \end{bmatrix}, \quad S=\begin{bmatrix} I & V_1 & & \\ W_2 & I & V_2 & \\ & W_3 & I & V_3 \\ & & W_4 & I \end{bmatrix}.$$

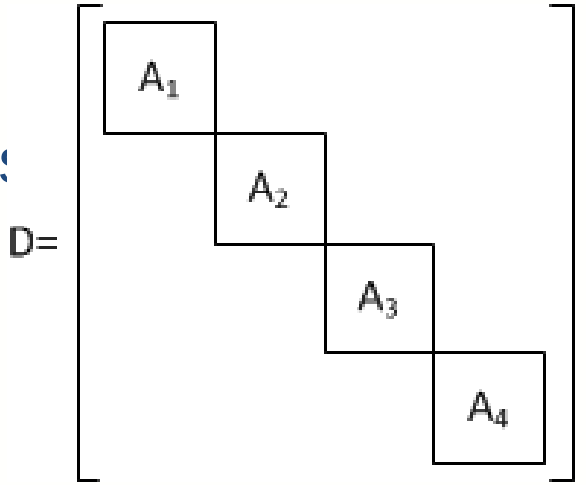
$$A_iV_i=\begin{bmatrix} 0 \\ \vdots \\ 0 \\ B_i \end{bmatrix} \quad (3) \qquad A_iW_i=\begin{bmatrix} C_i \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (4)$$



SX = Y (5) DY = F (6) AiYi = Fi (7)

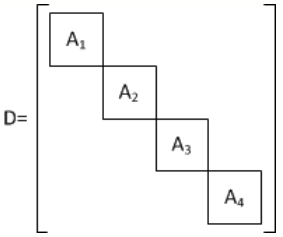
Data Layout

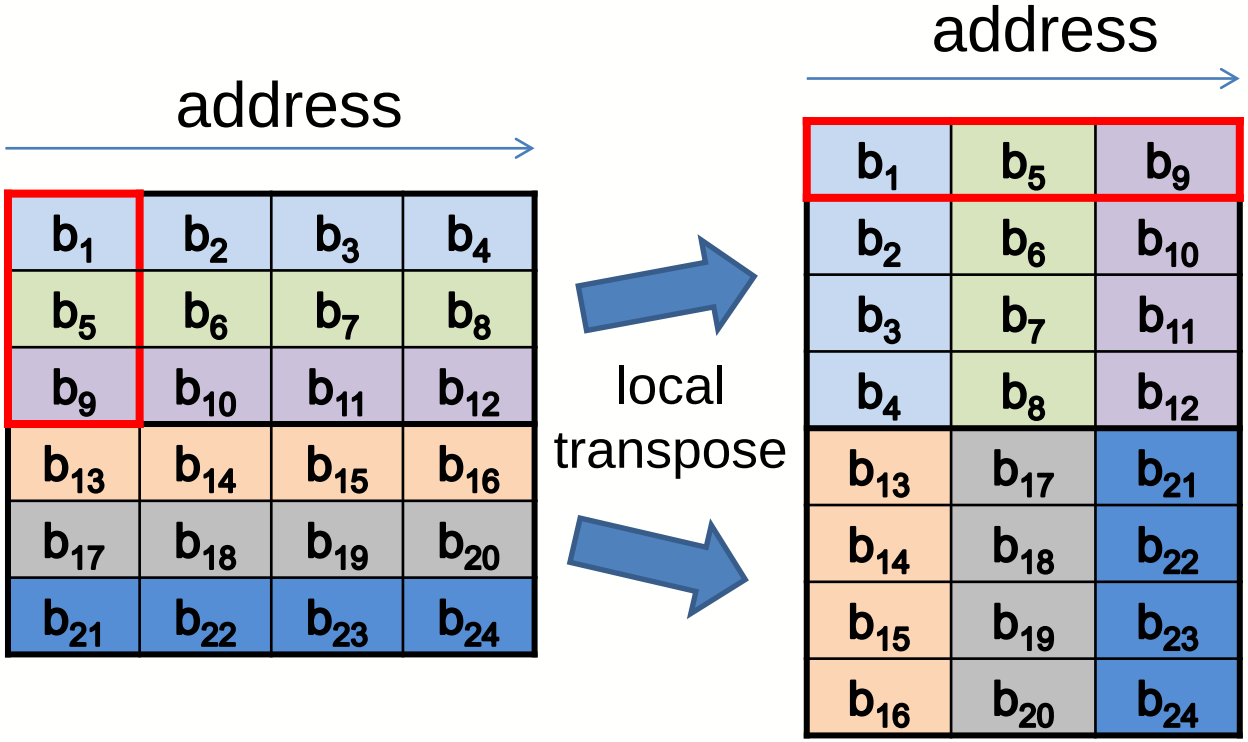
- Observation
  - GPUs require consecutive threads to access consecutive memory locations to fully utilize DRAM burst
- However, in SPIKE
  - Consecutive elements in a diagonal are stored in consecutive memory in gtsv interface
  - Each block is processed by one thread
  - Consecutive threads access locations that are of large distance away



Data Layout Transformation

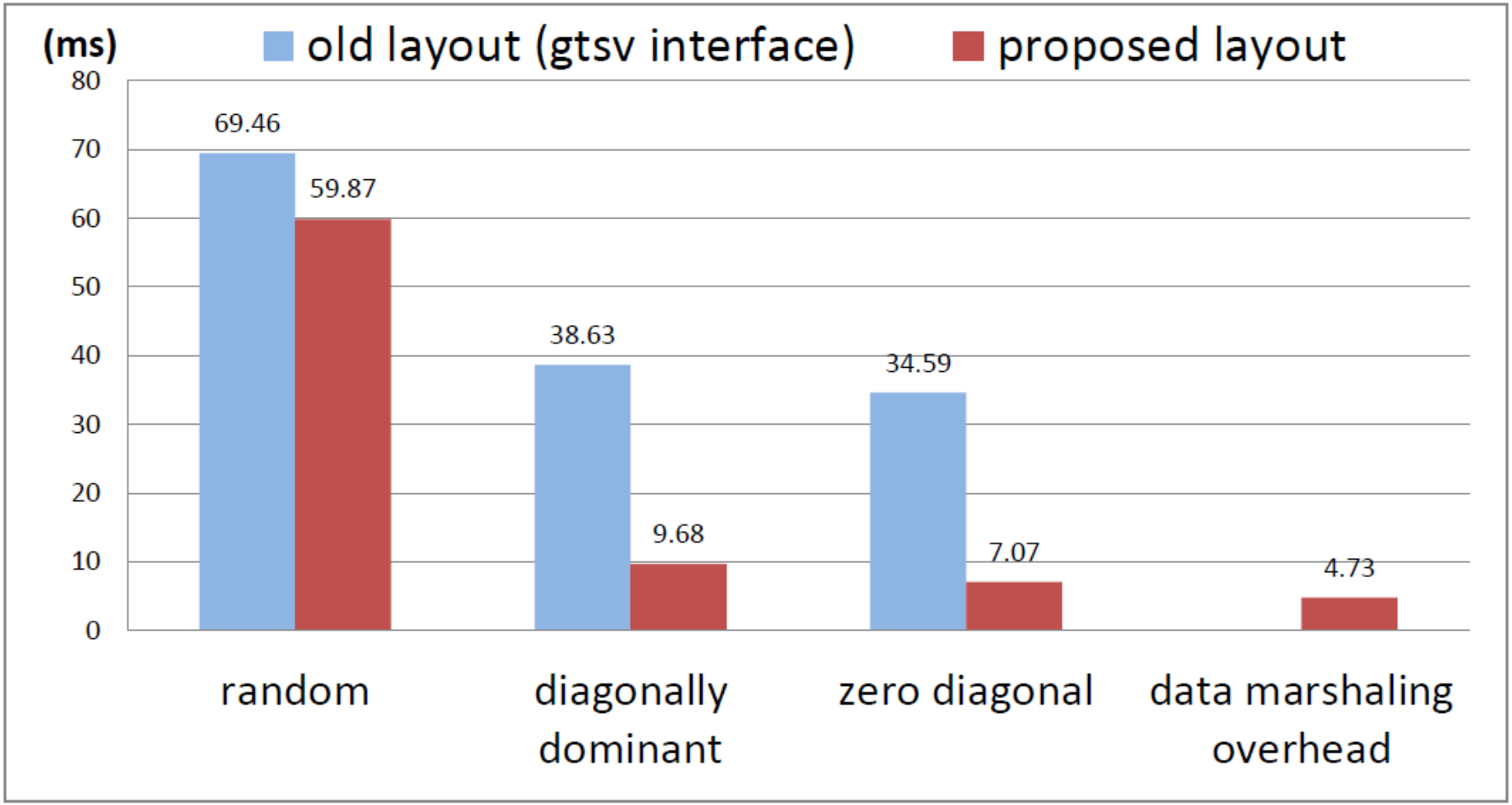
- Local transpose
  - b_i's are elements in a diagonal
  - 6 4-elements blocks (block in SPIKE)





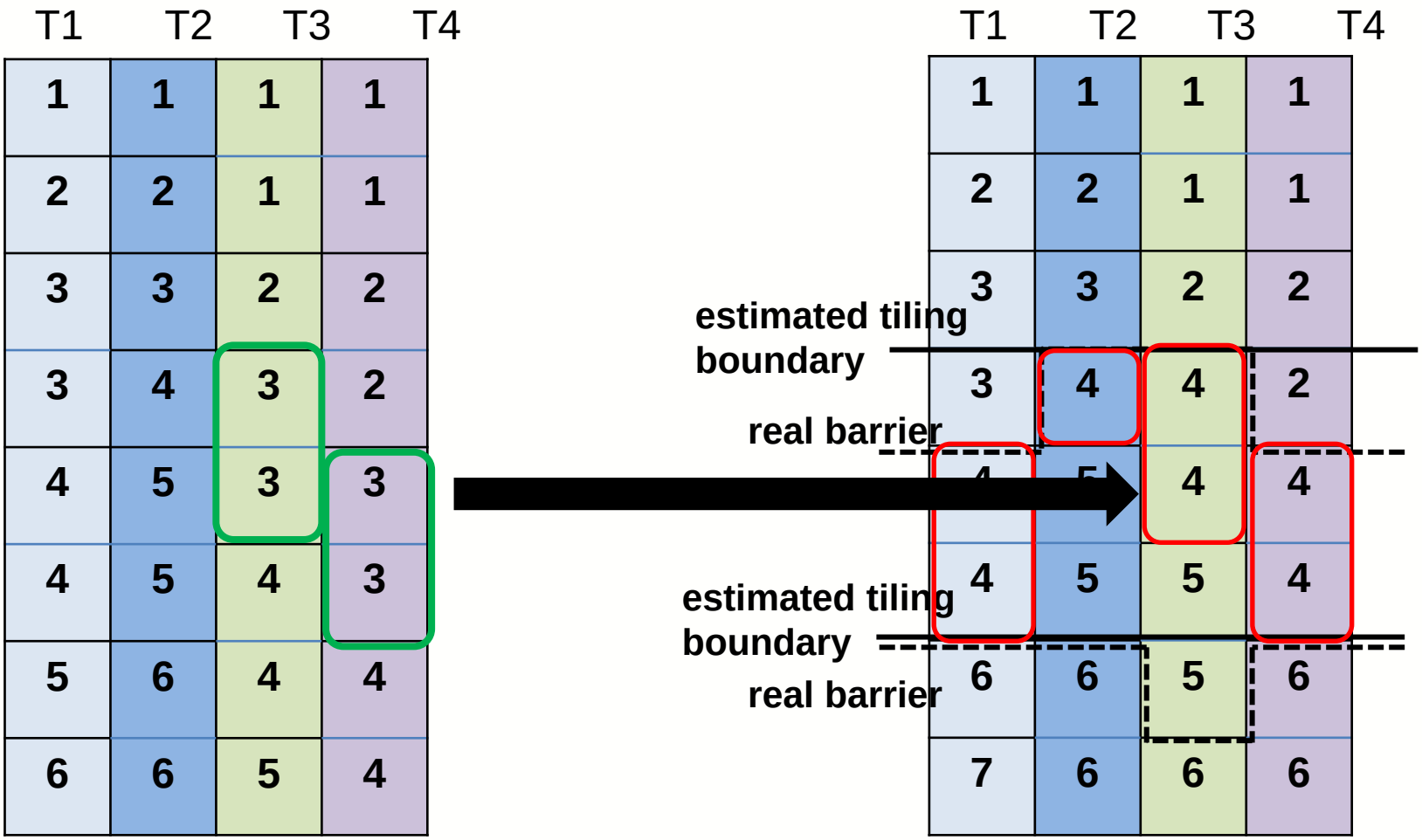
Sung, et al, InPar 2012

Fast Transposition on GPU



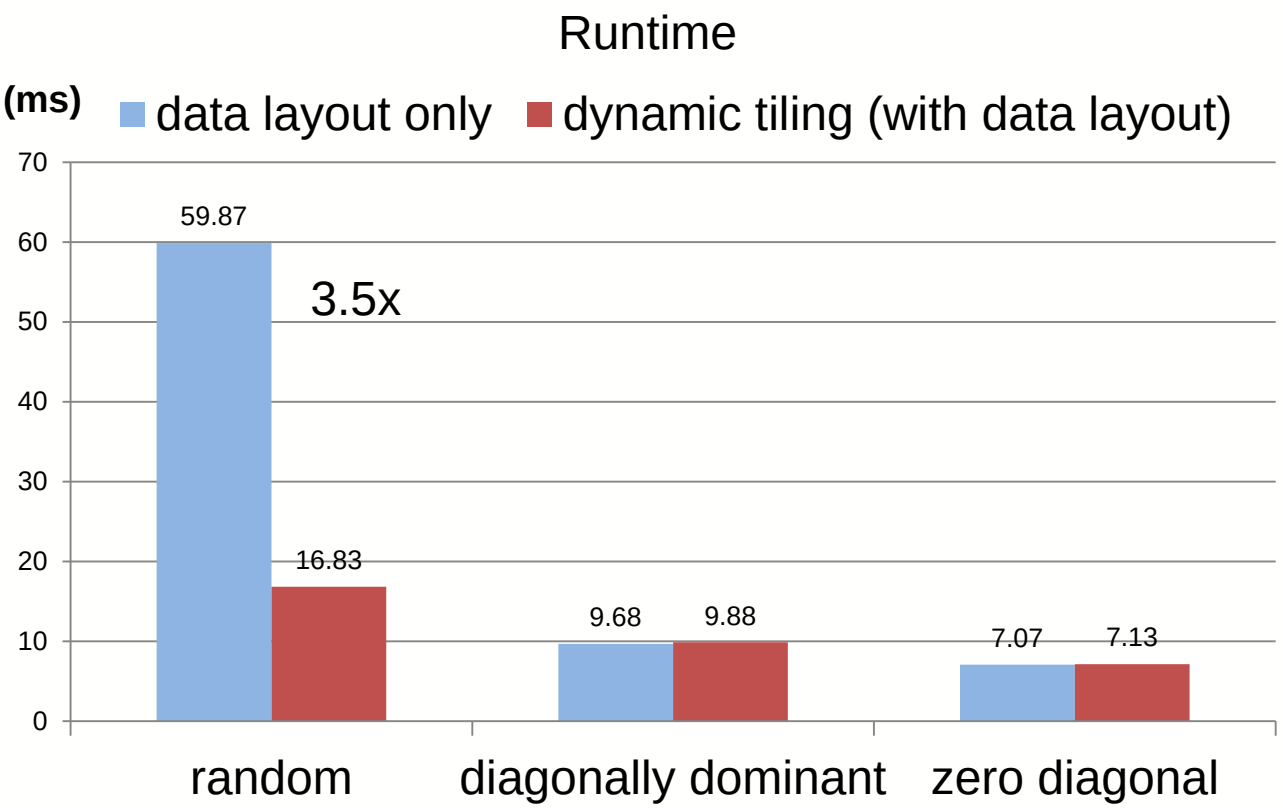
Chang, et al, SC 2012

Dynamic Tiling



Chang, et al, SC 2012

Dynamic Tiling



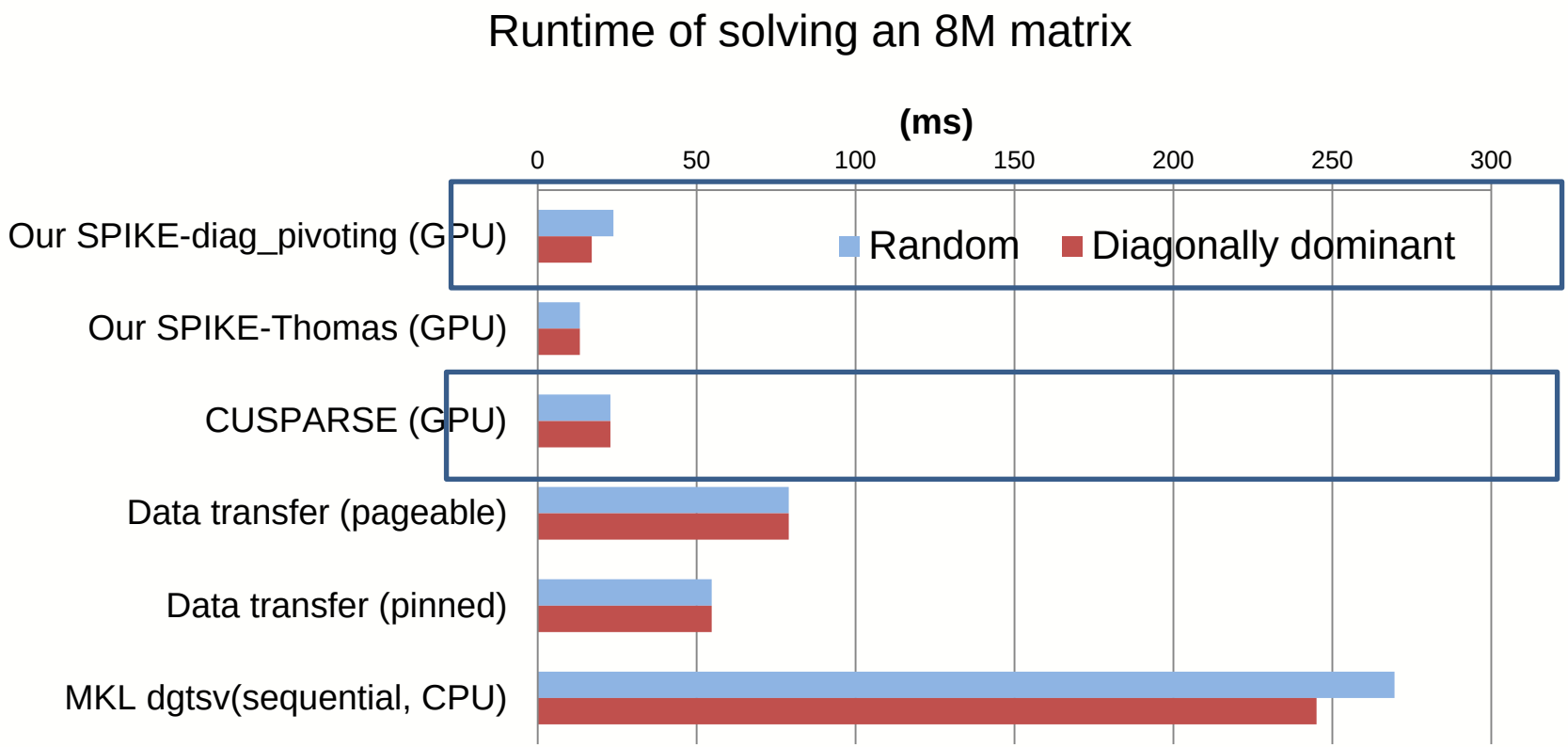
Numerical Error and Stability

Relative Backward Error

Matrix type	SPIKE-diag_pivoting	SPIKE-Thomas	CUSPARSE	MKL	Intel SPIKE	Matlab
1	1.82E-14	1.97E-14	7.14E-12	1.88E-14	1.39E-15	1.96E-14
2	1.27E-16	1.27E-16	1.69E-16	1.03E-16	1.02E-16	1.03E-16
3	1.55E-16	1.52E-16	2.57E-16	1.35E-16	1.29E-16	1.35E-16
4	1.37E-14	1.22E-14	1.39E-12	3.10E-15	1.69E-15	2.78E-15
5	1.07E-14	1.13E-14	1.82E-14	1.56E-14	4.62E-15	2.93E-14
6	1.05E-16	1.06E-16	1.57E-16	9.34E-17	9.51E-17	9.34E-17
7	2.42E-16	2.46E-16	5.13E-16	2.52E-16	2.55E-16	2.27E-16
8	2.14E-04	2.14E-04	1.50E+10	3.76E-04	2.32E-16	2.14E-04
9	2.32E-05	3.90E-04	1.93E+08	3.15E-05	9.07E-16	1.19E-05
10	4.27E-05	4.83E-05	2.74E+05	3.21E-05	4.72E-16	3.21E-05
11	7.52E-04	6.59E-02	4.54E+11	2.99E-04	2.20E-15	2.28E-04
12	5.58E-05	7.95E-05	5.55E-04	2.24E-05	5.52E-05	2.24E-05
13	5.51E-01	5.45E-01	1.12E+16	3.34E-01	3.92E-15	3.08E-01
14	2.86E+49	4.49E+49	2.92E+51	1.77E+48	3.86E+54	1.77E+48
15	2.09E+60	Nan	Nan	1.47E+59	Fail	3.69E+58
16	Inf	Nan	Nan	Inf	Fail	4.68E+171

Chang, et al, SC 2012

GPU Performance Advantage



Chang, et al, SC 2012

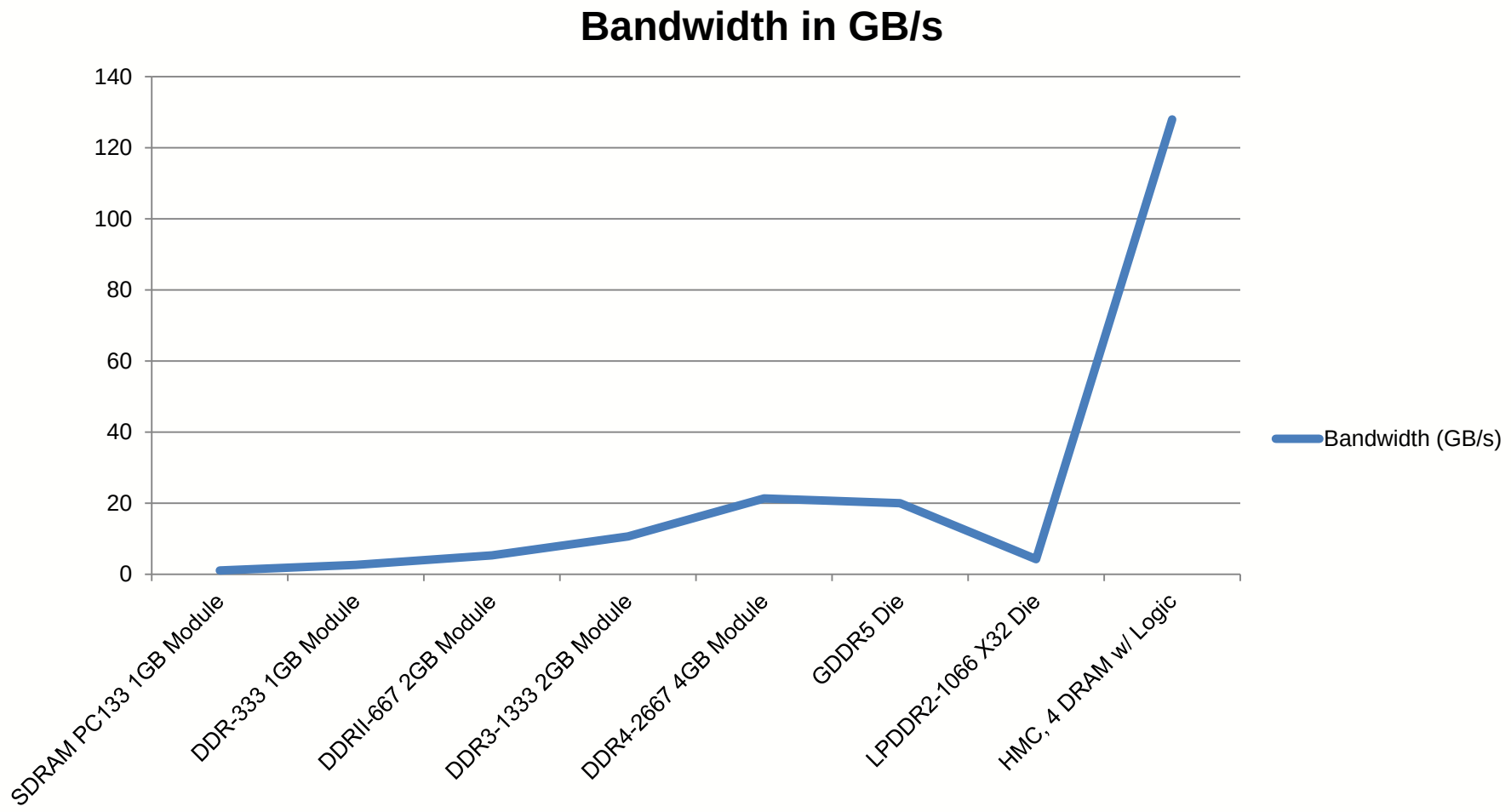
Eight Techniques for Scalable Kernels (so far)

	Memory Bandwidth	Update Contention	Load Balance	Regularity	Efficiency
Scatter to Gather		X			
Privatization		X			
Tiling	X				X
Coarsening	X	X			X
Data Layout	X	X			X
Input Binning	X				X
Regularization			X	X	X
Compaction	X		X	X	X

Stratton, et al, IEEE Computer, 8/2012

SOME IMPORTANT TRENDS

DRAM trends in bandwidth

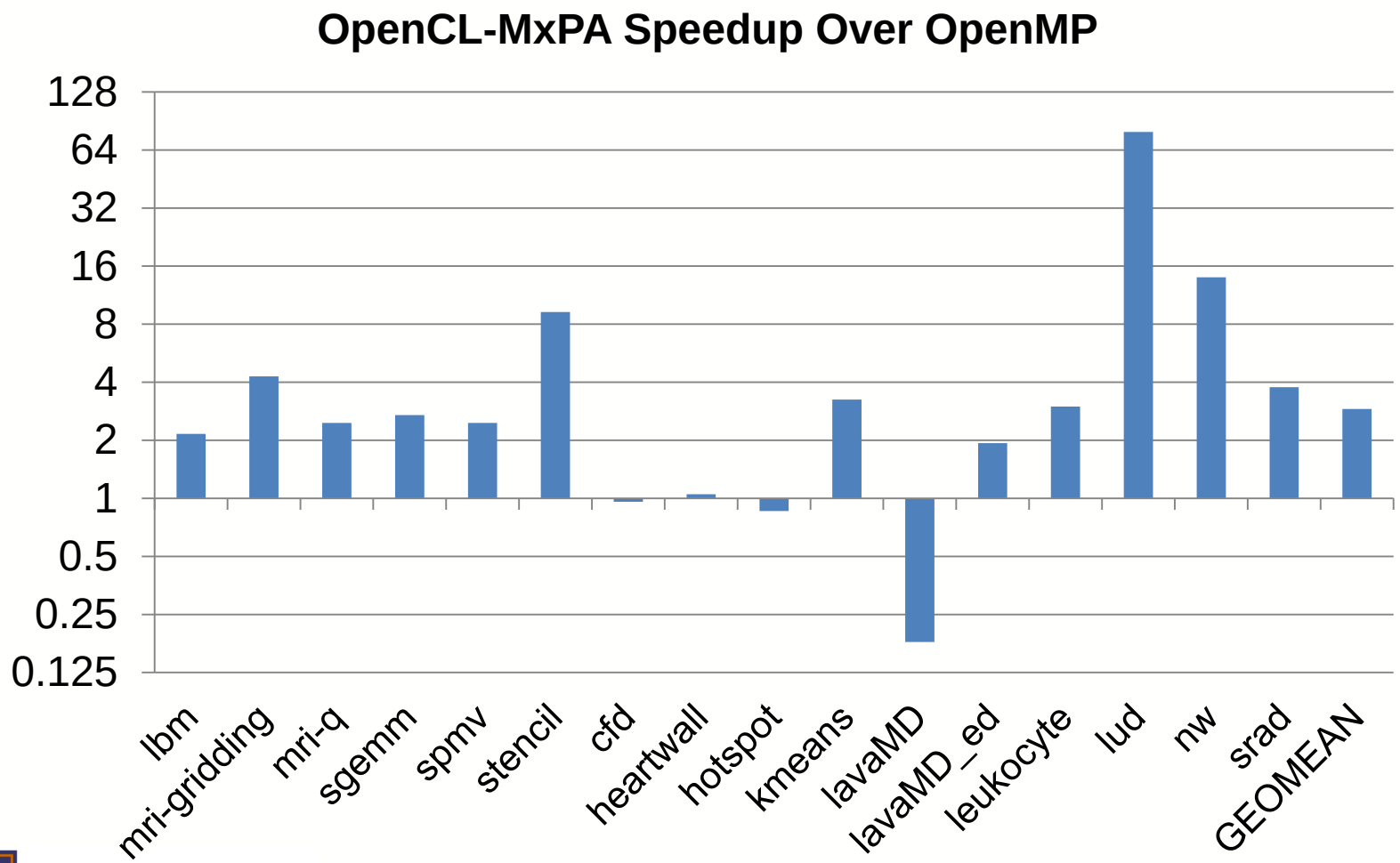


Source: J. Thomas Pawlowski, “Hybrid Memory Cube (HMC)”, Hot Chips 23

CPU/GPU Performance Portability

- Developers today almost never “port” their code to OpenCL. They “fork” a version of their code for OpenCL, and maintain their original CPU code as well
  - Perception that OpenCL code will not work well for CPUs

But there is hope!



Stratton, Ph.D. Thesis

Exascale Energy Pressure

- Pressure for higher energy efficiency will likely make processors more difficult to program
  - More specialized processor data path (width, connectivity, etc.)
  - Wider SIMD
  - More system-level data movement control
  - Smaller on-chip storage per thread
  - …

Research Opportunities

- Most important scalability goals are goals are still achieved by hand today
  - Lack of effective compiler tools and APIs
  - Too many code versions and too much programming effort
- Compiler transformations have been built on dependence analysis that require array accesses based on affine expression
  - Most real data structures do not have affine access expressions
  - May need strongly typed programming interfaces to alleviate this
- Current directive based runtime systems require too much programmer effort
- New programming interfaces, compilers, APIs and runtime reduce programming cost for scalability and portability

Conclusion and Outlook

- We have enjoyed some victories
  - Good set of applications and kernels
  - Good low-level interface in major languages
  - Good initial results, educated developers
- We will face more battles
  - Energy efficiency vs. easy of programming
  - Potential fragmentation of programming interfaces
  - Widen the set of applications, algorithms and kernels
  - Robust programming interfaces and tools

Acknowledgements

- D. August (Princeton), S. Baghsorkhi (Illinois), N. Bell (NVIDIA), D. Callahan (Microsoft), J. Cohen (NVIDIA), B. Dally (Stanford), J. Demmel (Berkeley), P. Dubey (Intel), M. Frank (Intel), M. Garland (NVIDIA), Isaac Gelado (BSC), M. Gschwind (IBM), R. Hank (Google), J. Hennessy (Stanford), P. Hanrahan (Stanford), M. Houston (AMD), T. Huang (Illinois), D. Kaeli (NEU), K. Keutzer (Berkeley), I. Gelado (UPC), B. Gropp (Illinois), D. Kirk (NVIDIA), D. Kuck (Intel), S. Mahlke (Michigan), T. Mattson (Intel), N. Navarro (UPC), J. Owens (Davis), D. Padua (Illinois), S. Patel (Illinois), Y. Patt (Texas), D. Patterson (Berkeley), C. Rodrigues (Illinois), S. Ryoo (ZeroSoft), K. Schulten (Illinois), B. Smith (Microsoft), M. Snir (Illinois), I. Sung (Illinois), P. Stenstrom (Chalmers), J. Stone (Illinois), S. Stone (Harvard) J. Stratton (Illinois), H. Takizawa (Tohoku), M. Valero (UPC)

- And many others!

There is always hope.

– Aragorn in the eve of the Battle of Pelennor Minas Tirith

THANK YOU!