VI Latin American Symposium on High-Performance Computing
HPCLatAm 2013

Hierarchical N-body algorithms for the Exascale era

Lorena A. Barba

Boston University & **The George Washington University** (from August 2013)

@LorenaABarba

http://lorenabarba.com/

Acknowledgements

NSF CAREER award

NVIDIA Academic Partnership Award
CUDA Fellow

ONR Applied Computational Analysis Program

Acknowledgement

joint work with Dr. Rio Yokota
here at Nagasaki Advanced Computing Center, Japan

A history lesson

> Improved algorithms and libraries have contributed as much to increases in capability as have improvements in hardware.

p. 54 of the 2005 Report to the President of the United States, President's Information Technology Advisory Committee, PITAC.

Algorithmic speedup

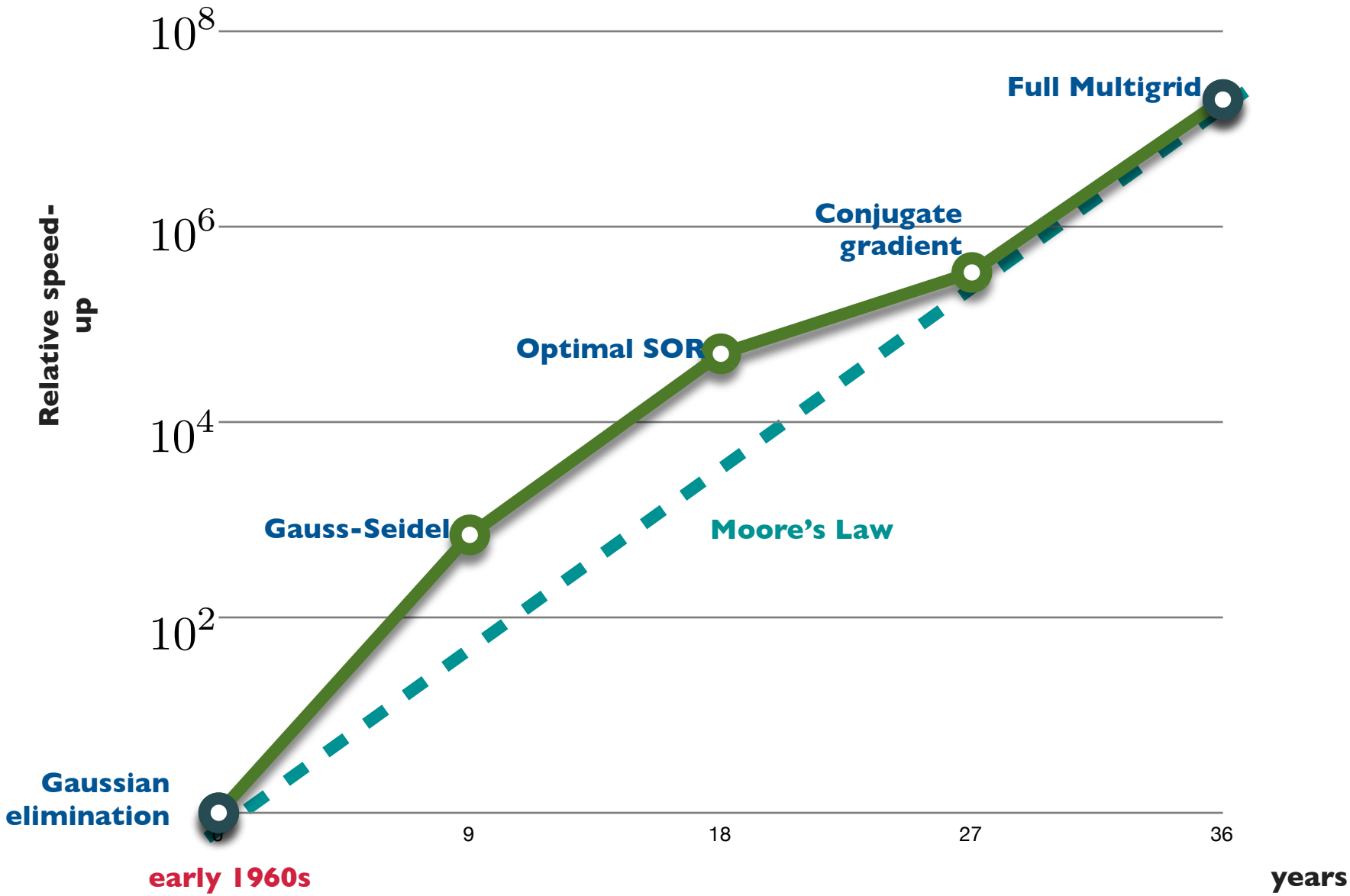
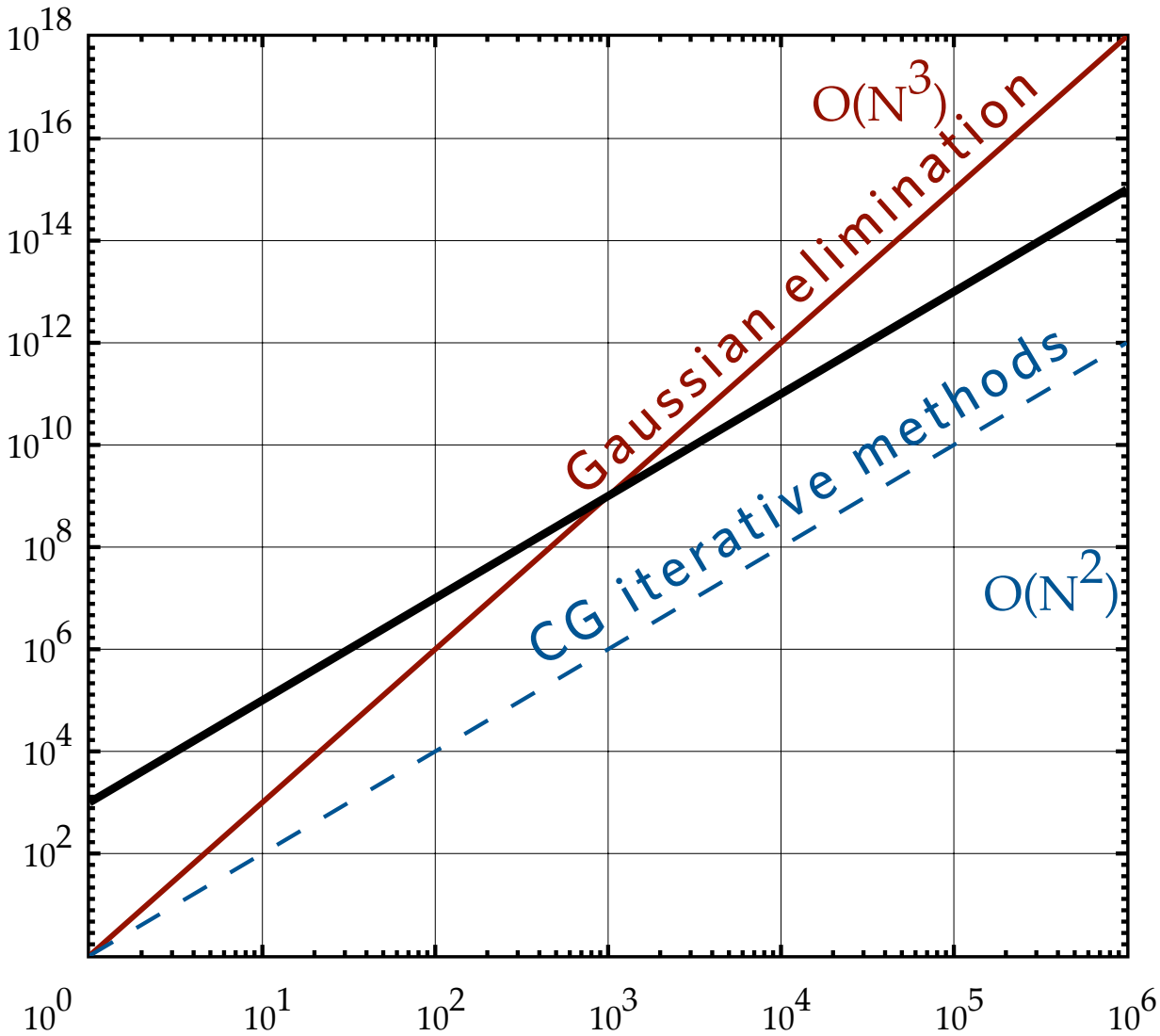


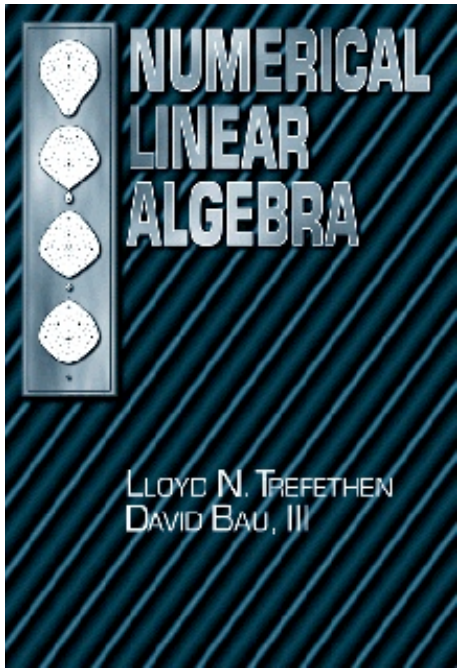
Fig. 5, p. 53 of *Computational Science: Ensuring America's Competitiveness* (2005) Report to the President of the United States, President's Information Technology Advisory Committee, PITAC.

Algorithms can often speed up science as much or more than Moore’s law.

The curious story of conjugate gradient algorithms

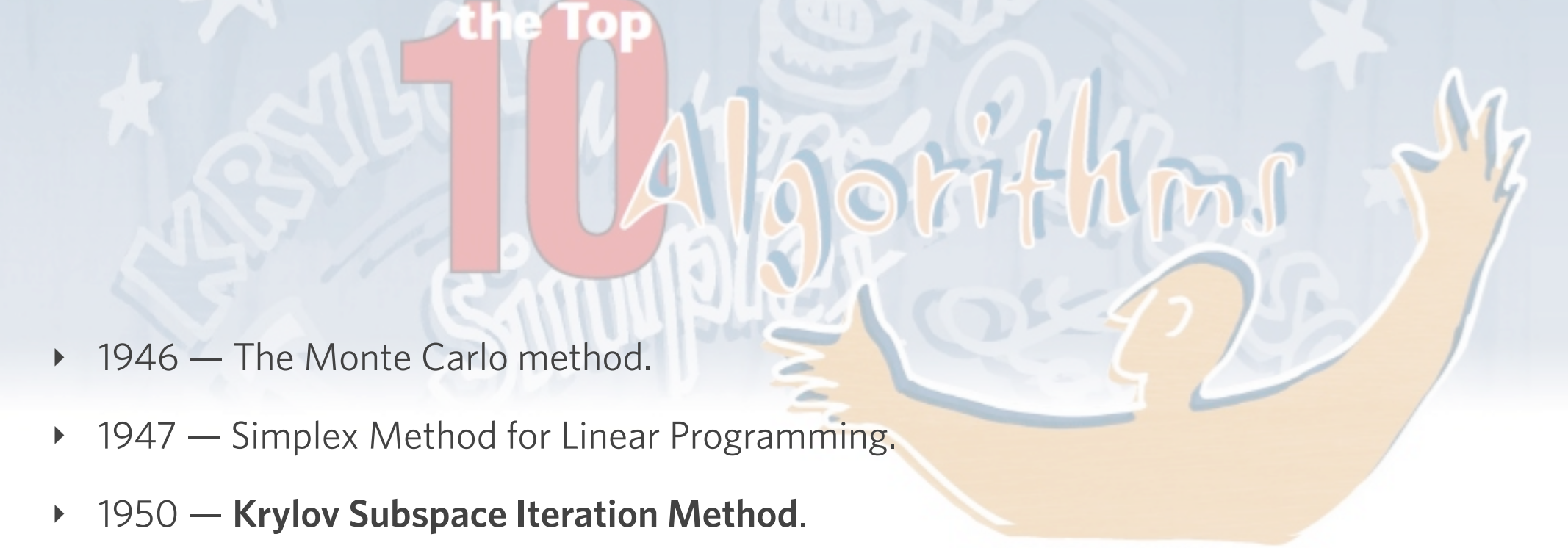


- Iterative methods:
  - *sequence of iterates converging to the solution*
- CG matrix iterations bring the $O(N^3)$ cost to $O(N^2)$
- 1950s — ***N too small*** for CG to be competitive
- 1970s — renewed attention



"... the fundamental law of computer science [is]: the faster the computer, the greater the importance of speed of algorithms"

Trefethen & Bau "Numerical Linear Algebra" SIAM



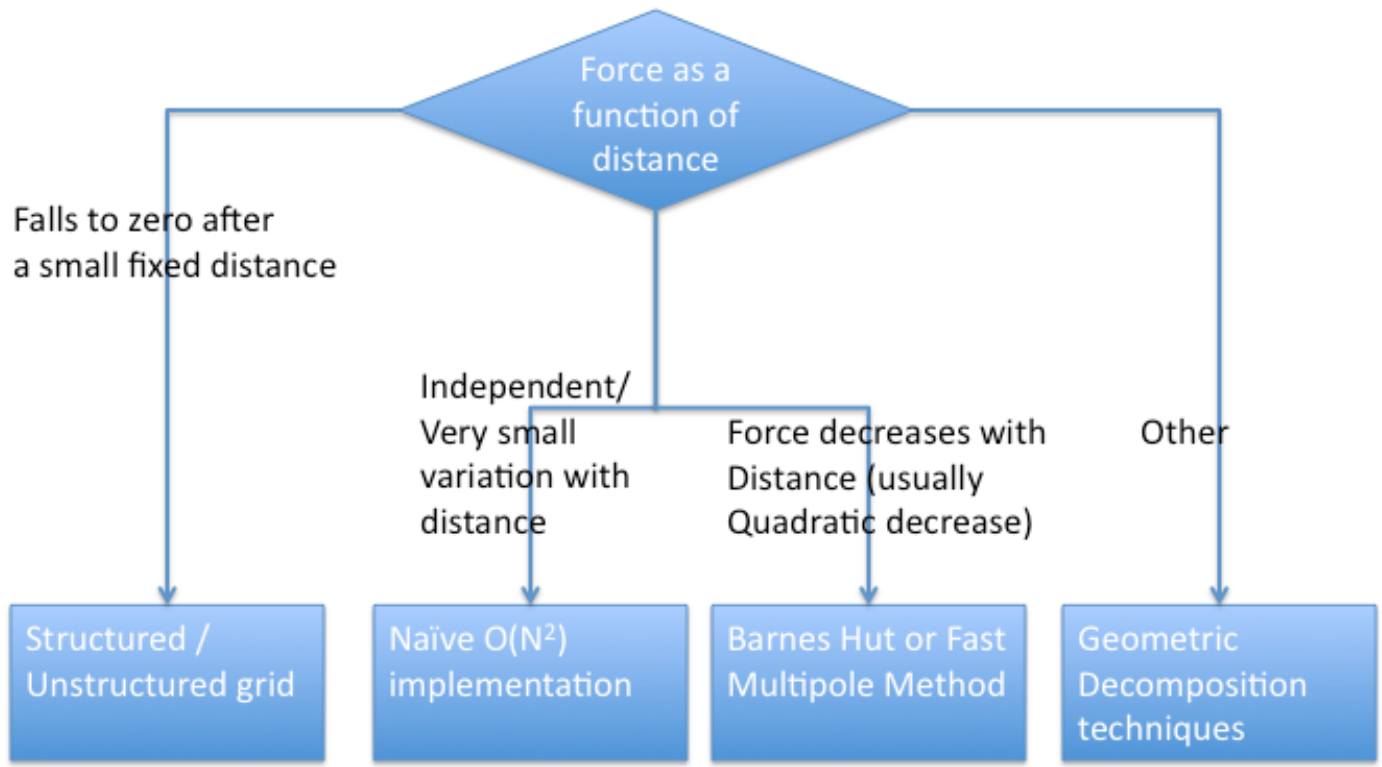
- 1946 — The Monte Carlo method.
- 1947 — Simplex Method for Linear Programming.
- 1950 — **Krylov Subspace Iteration Method**.
- 1951 — The Decompositional Approach to Matrix Computations.
- 1957 — The Fortran Compiler.
- 1959 — QR Algorithm for Computing Eigenvalues.
- 1962 — Quicksort Algorithms for Sorting.
- 1965 — Fast Fourier Transform.
- 1977 — Integer Relation Detection.
- 1987 — **Fast Multipole Method**

Dongarra& Sullivan, IEEE Comput. Sci. Eng., Vol. 2(1):22-- 23 (2000)

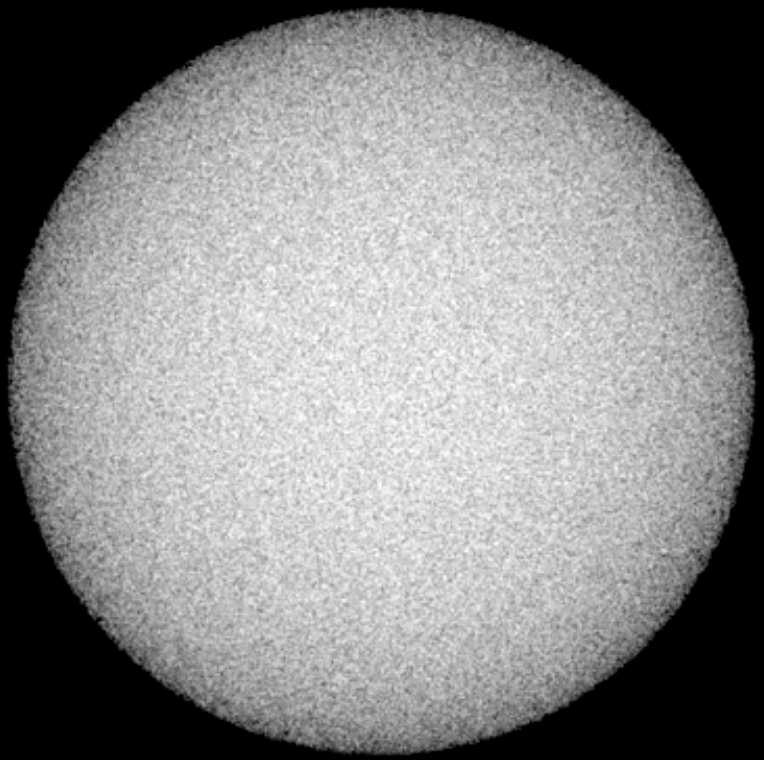
Hierarchichal N-body algorithms

N-body

- **Problem:**
  "updates to a system where each element of the system rigorously depends on the state of every other element of the system."



http://parlab.eecs.berkeley.edu/wiki/patterns/n-body_methods

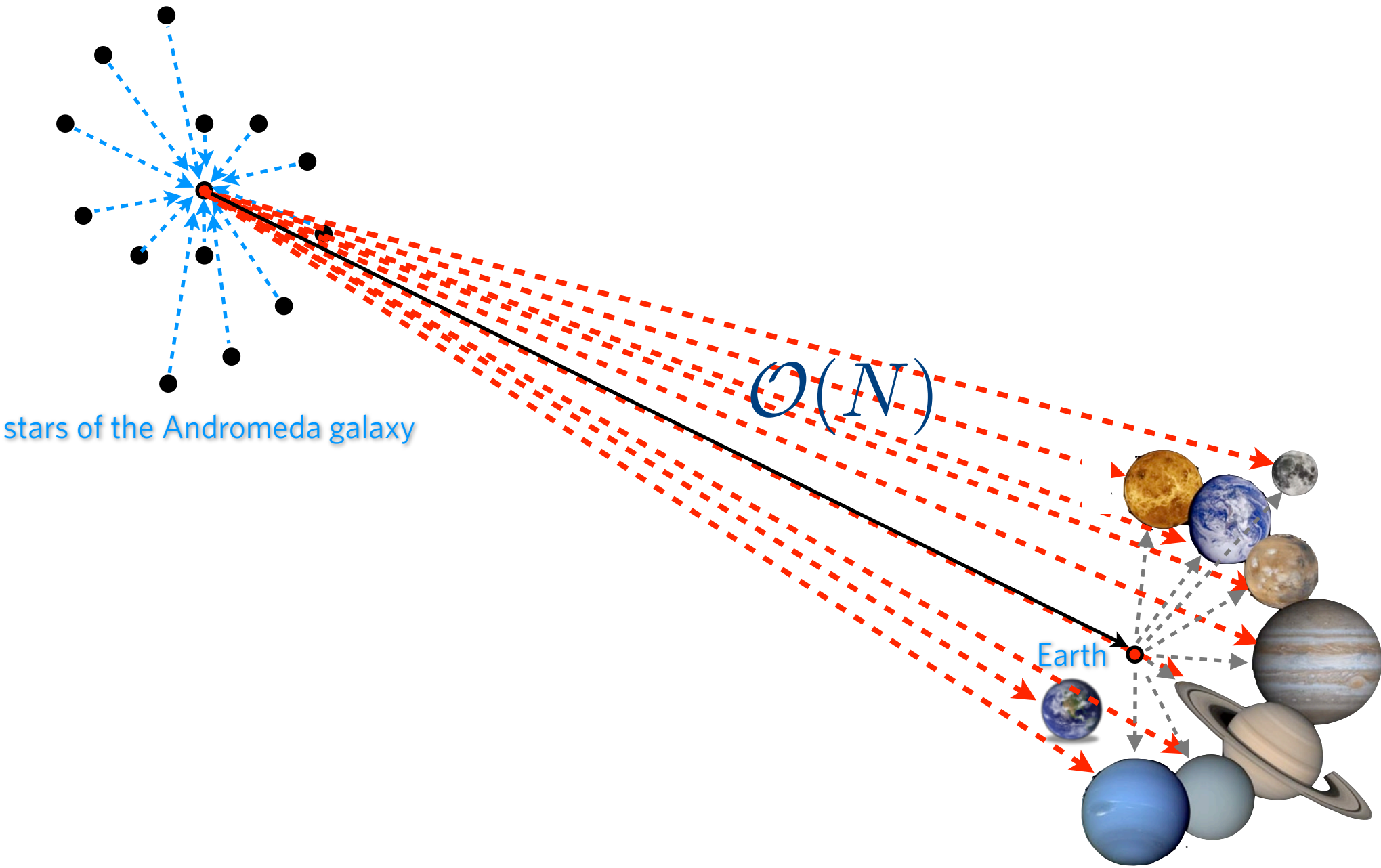


Credit: **Mark Stock**


M31 Andromeda galaxy
stars: 10^{12}


CASSIOPEIA
PEGASUS
CAMELOPARDALIS
Andromeda Galaxy
ANDROMEDA
ALGOL
PERSEUS
TRIANGULUM
PISCES
Capella
ARIES
East

Fast N-body method



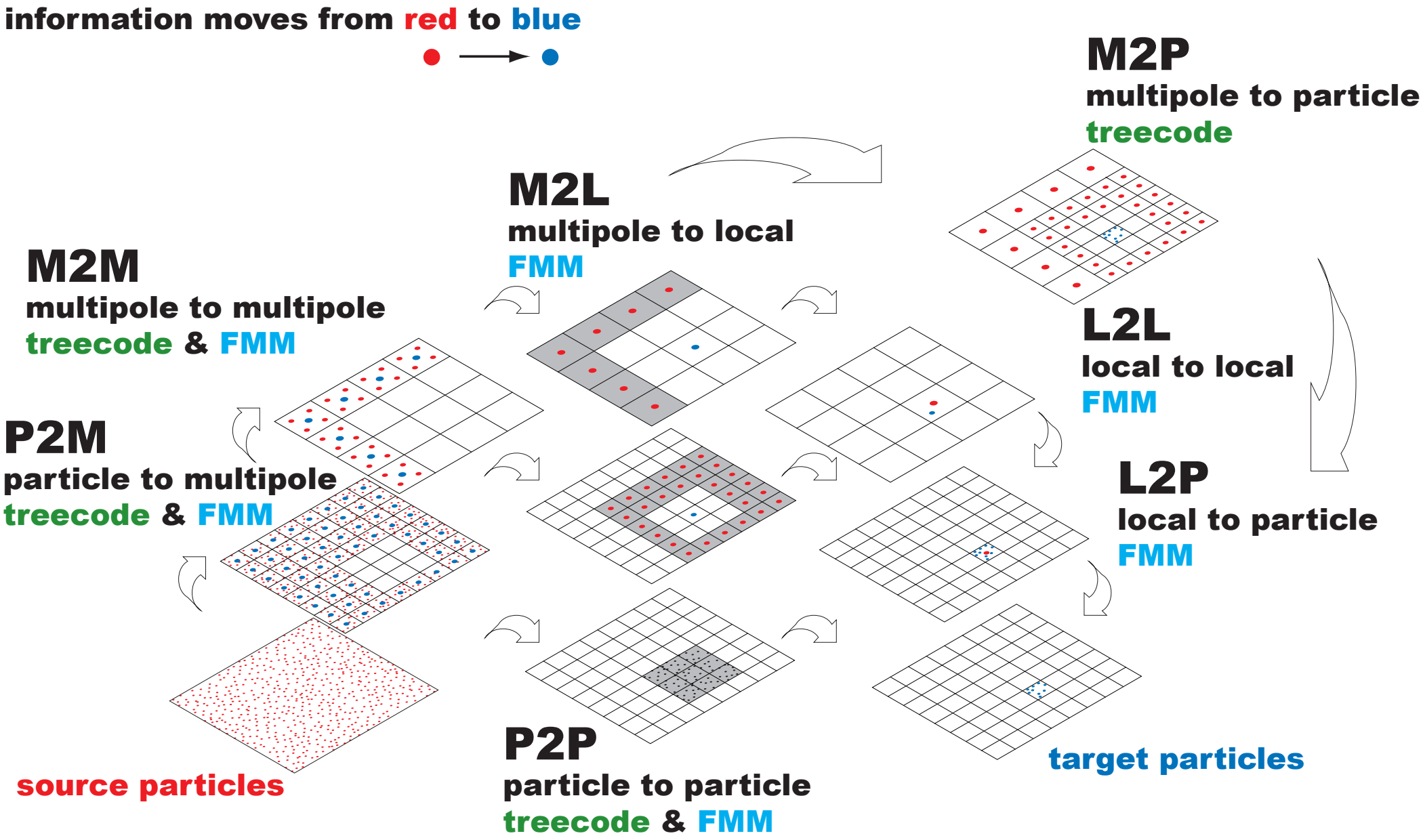
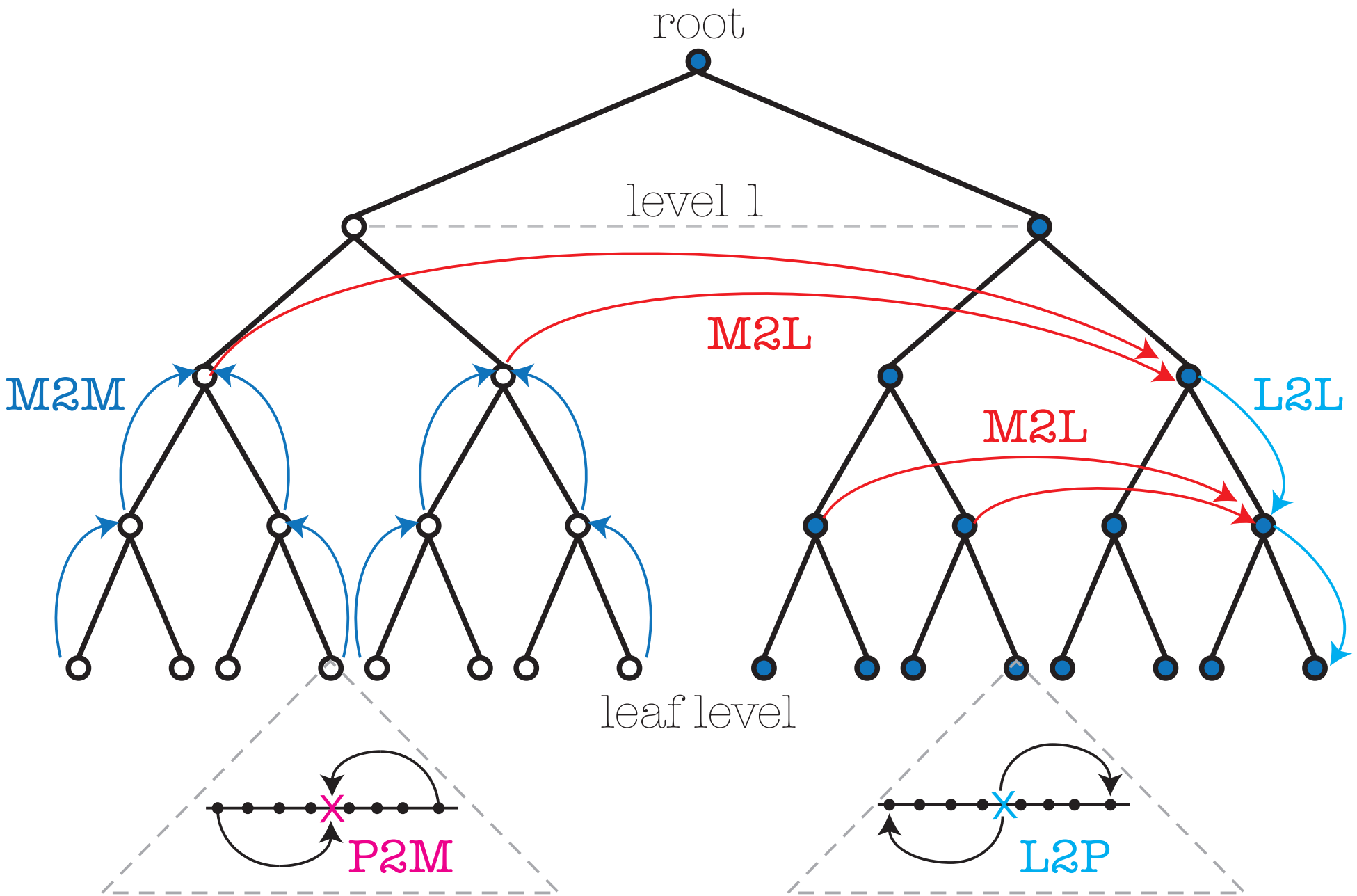


Image: "Treecode and fast multipole method for N-body simulation with CUDA", Rio Yokota, Lorena A Barba, Ch. 9 in *GPU Computing Gems Emerald Edition*, Wen-mei Hwu, ed.; Morgan Kaufmann/Elsevier (2011) pp. 113–132.

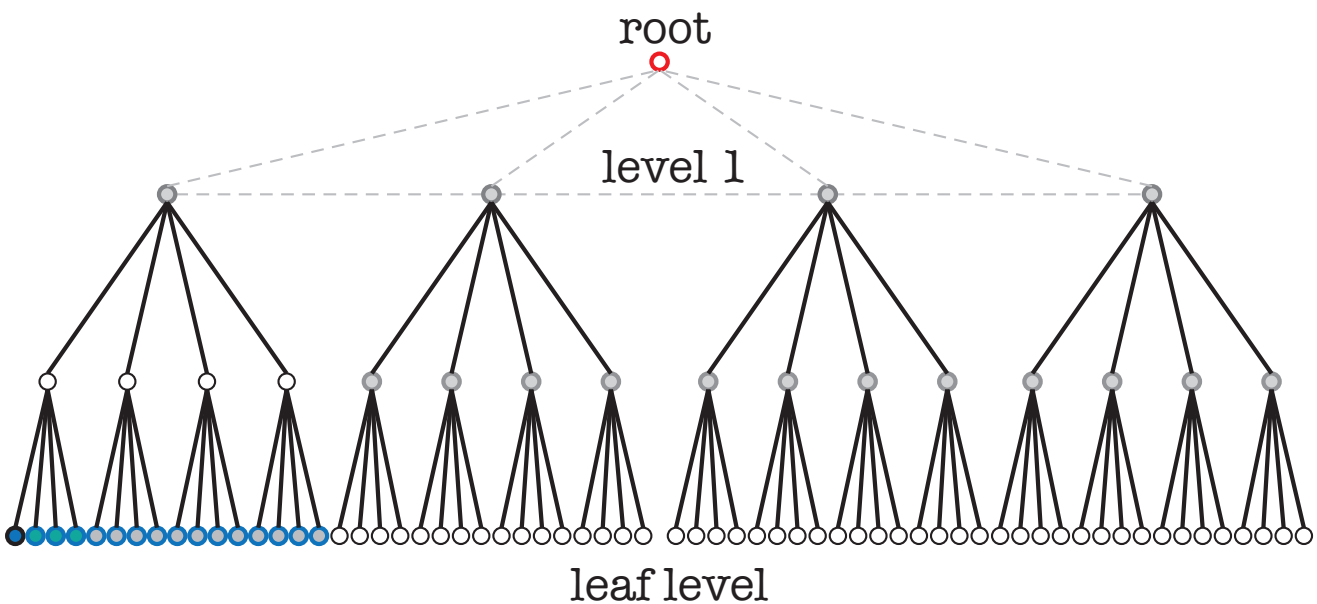


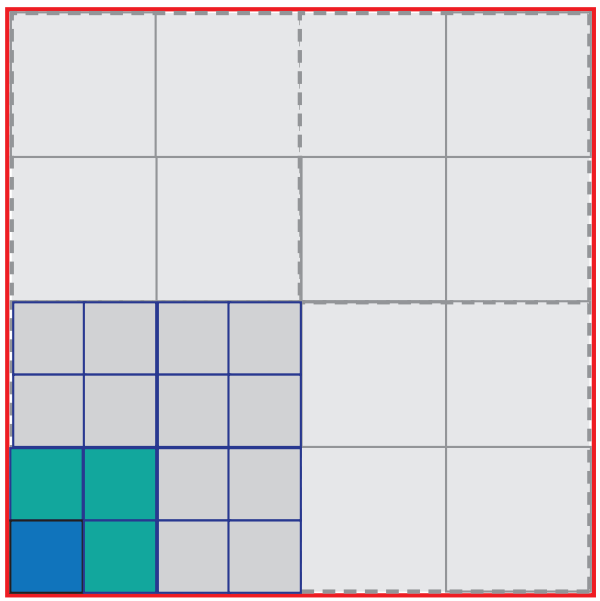
"A tuned and scalable fast multipole method as a preeminent algorithm for exascale systems", Rio Yokota, L A Barba. *Int. J. High-perf. Comput*. 26(4):337–346 (November 2012)

Treecode & Fast multipole method

- reduces operation count from **O(N²) to O(N log N) or O(N)**

$$f(y) = \sum_{i=1}^{N} c_i \mathbf{K}(y - x_i) \qquad y \in [1...N]$$





"A tuned and scalable fast multipole method as a preeminent algorithm for exascale systems", Rio Yokota, L A Barba. *Int. J. High-perf. Comput*. 26(4):337–346 (November 2012)

N-body simulation on GPU hardware:

The algorithmic and hardware speed-ups multiply.

Early application of GPUs

- 2007, Hamada & Iitaka — CUNbody
  - distributed source particles among thread blocks, requiring reduction
- 2007, Nyland et al. — GPU Gems 3
  - target particles were distributed, no reduction necessary

- 2008, Belleman et al. — ‘Kirin’ code
- 2009, Gaburov et al. — ‘Sapporo’ code

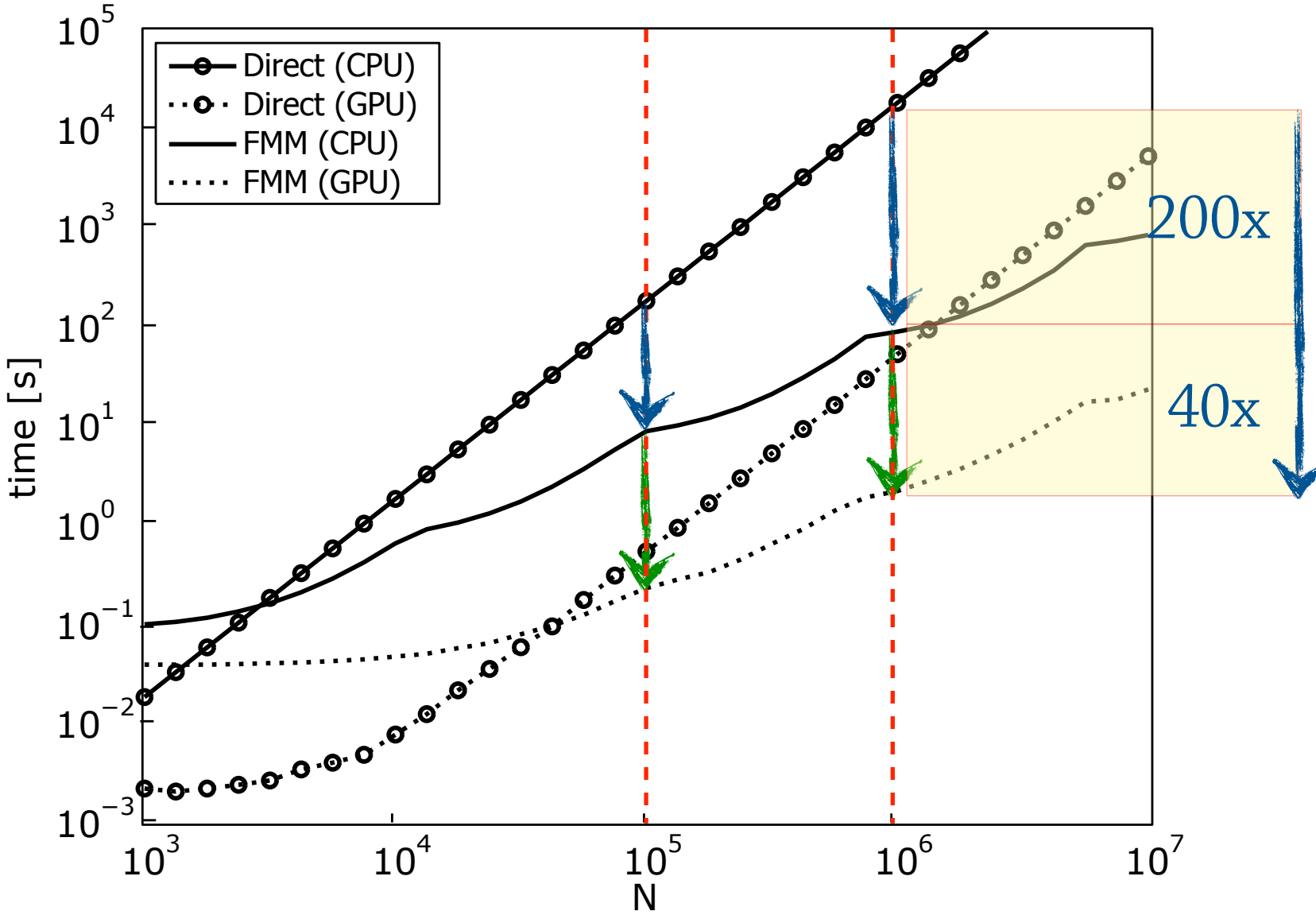
These are all
Direct Summation
--- double check

FMM on GPU: **multiplying speed-ups**

Note:

p=10

L^2-norm error (normalized): 10^{-4}



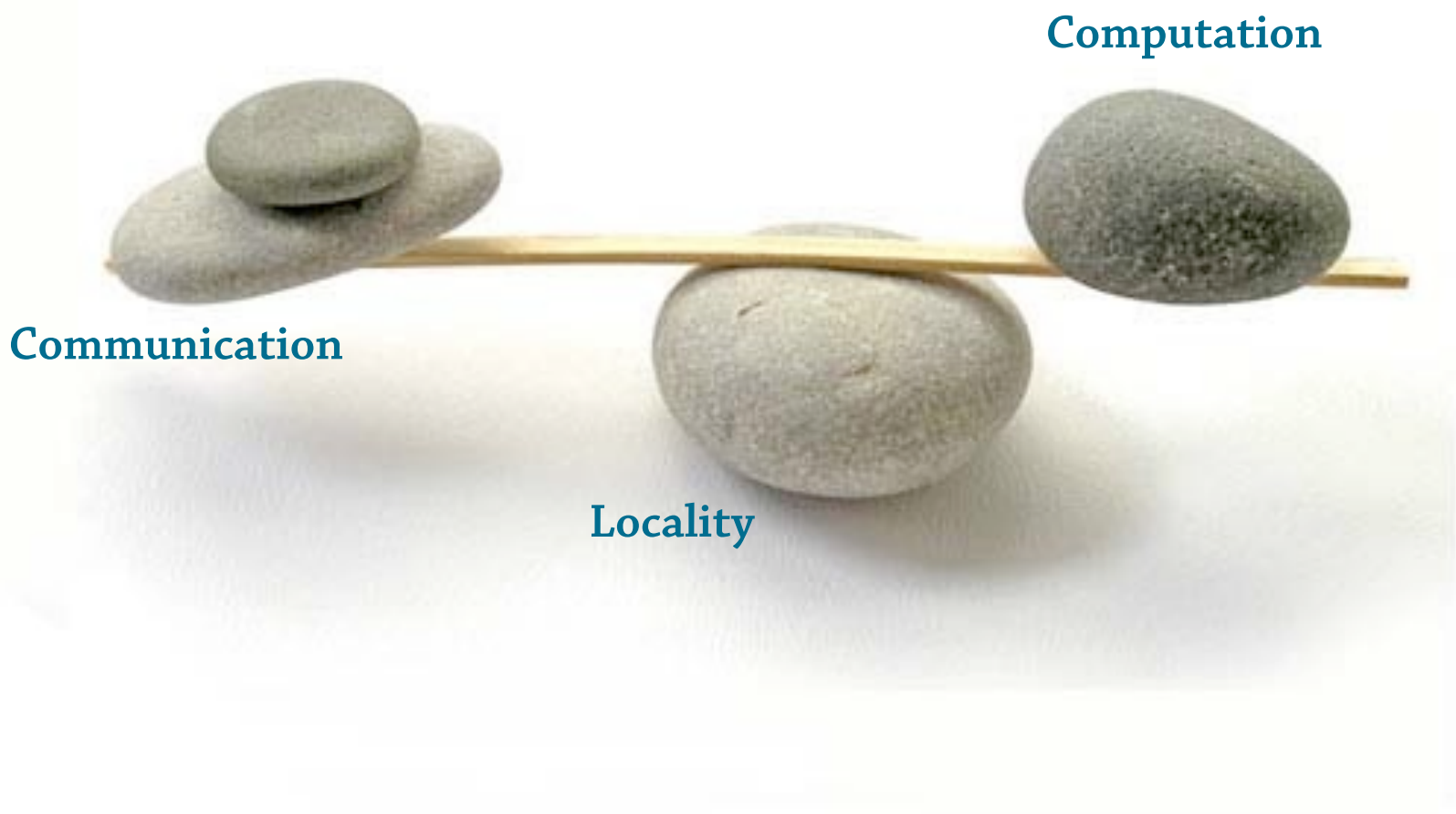
"Treecode and fast multipole method for N-body simulation with CUDA", R Yokota & L A Barba, Ch. 9 in ***GPU Computing Gems Emerald Edition,*** Elsevier/Morgan Kaufman (2011)

Advantage of N-body algorithms on GPUs

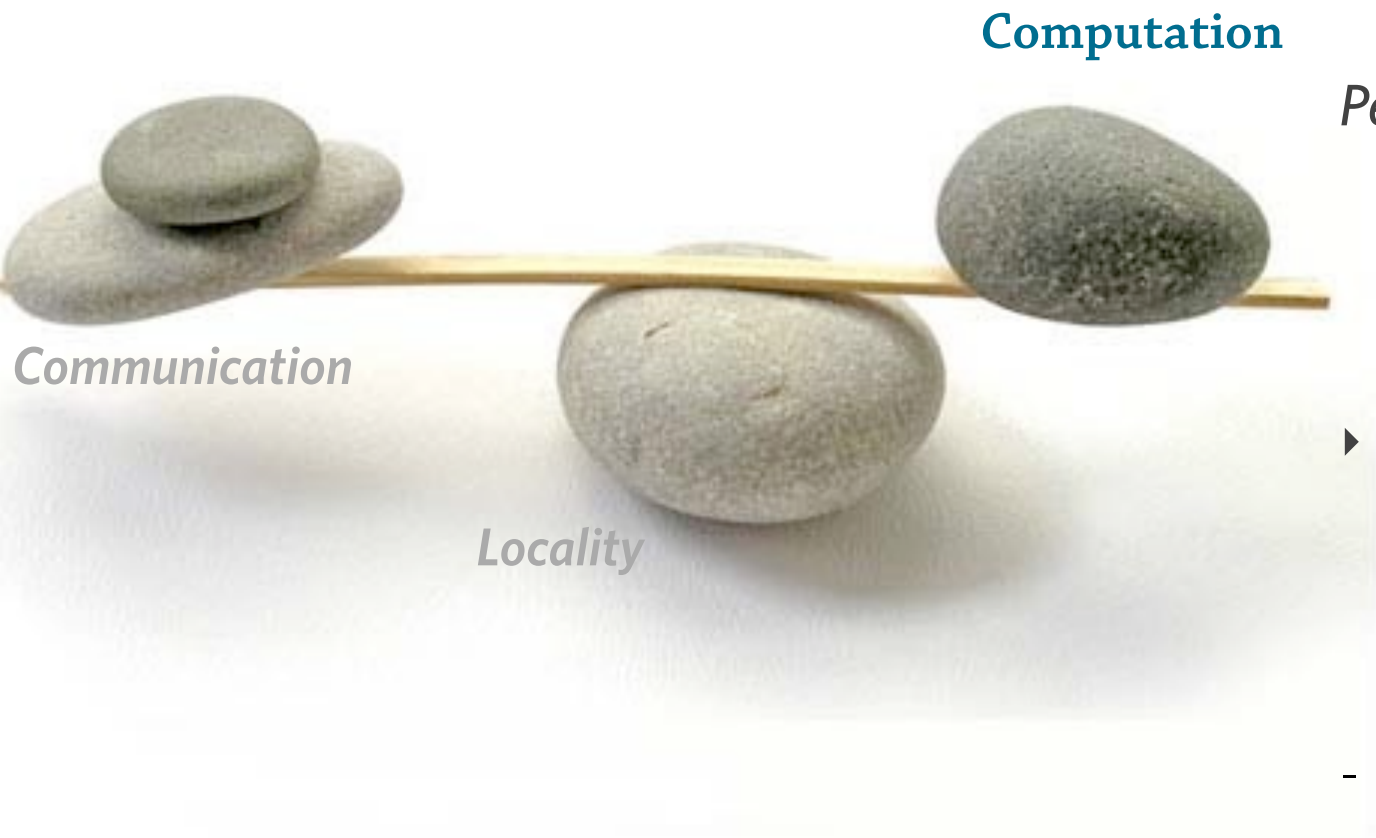
- quantify using the ***Roofline Model***

- shows hardware barriers ('ceiling') on a computational kernel

- *Components of performance:*



Performance: **Computation**



Metric:

- Gflop/s
- dp / sp

Peak achivable if:

- exploit FMA, etc.
- non-divergence (GPU)

- Intra-node parallelism:
  - explicit in algorithm
  - explicit in code

-

Source: ParLab, UC Berkeley

Performance: **Communication**

Metric:

- GB/s

Peak achivable if optimizations are explicit

- prefetching
- allocation/usage
- stride streams
- coalescing on GPU

-

Source: ParLab, UC Berkeley

Performance: **Locality**

"Computation is free"

- Maximize locality > minimize communication
- Comm lower bound

-

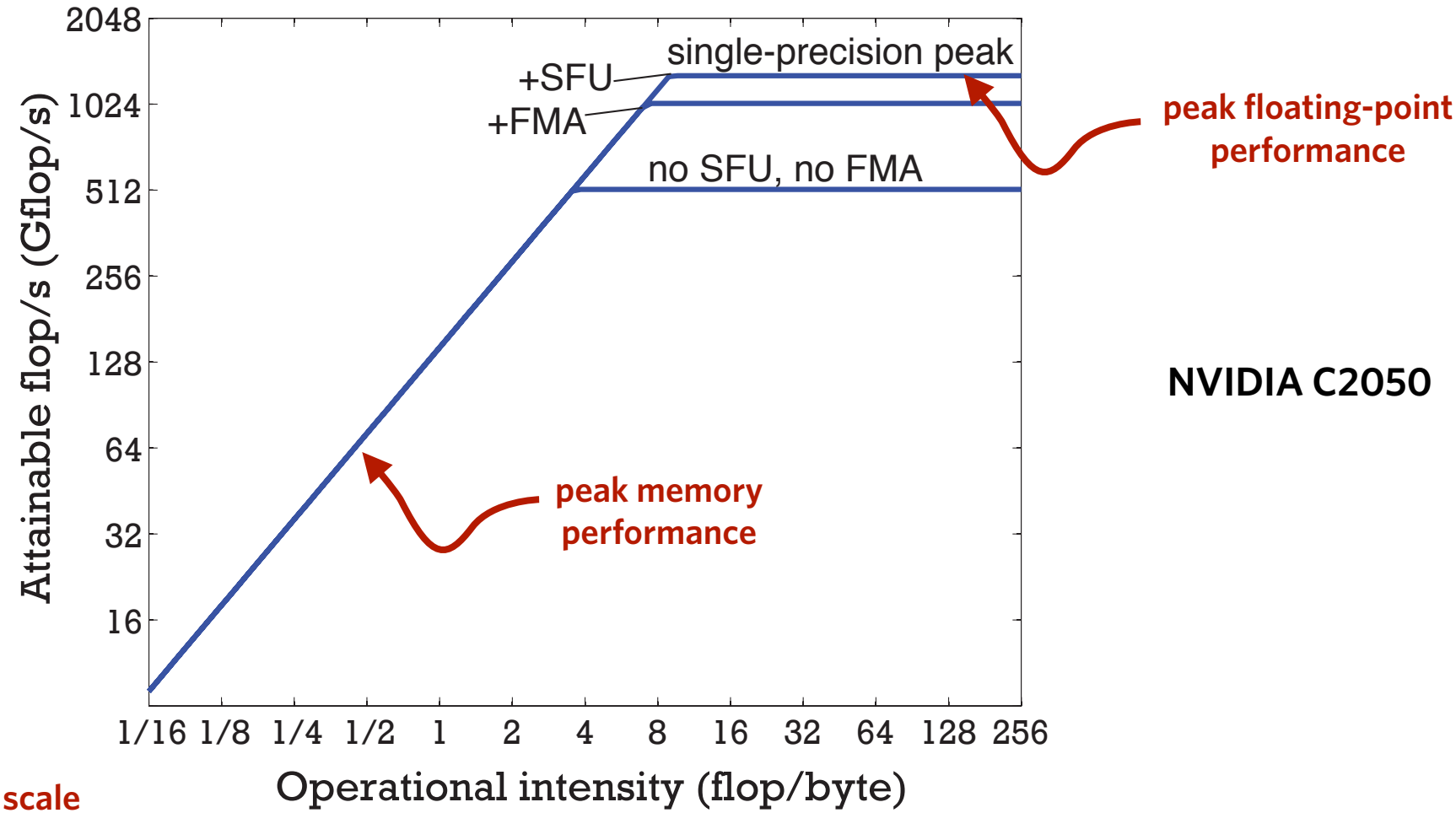
	Hardware aids	*Optimizations via software*
• minimize capacity misses	• cache size	• blocking
• minimize conflict misses	• associativities	• padding

Source: ParLab, UC Berkeley

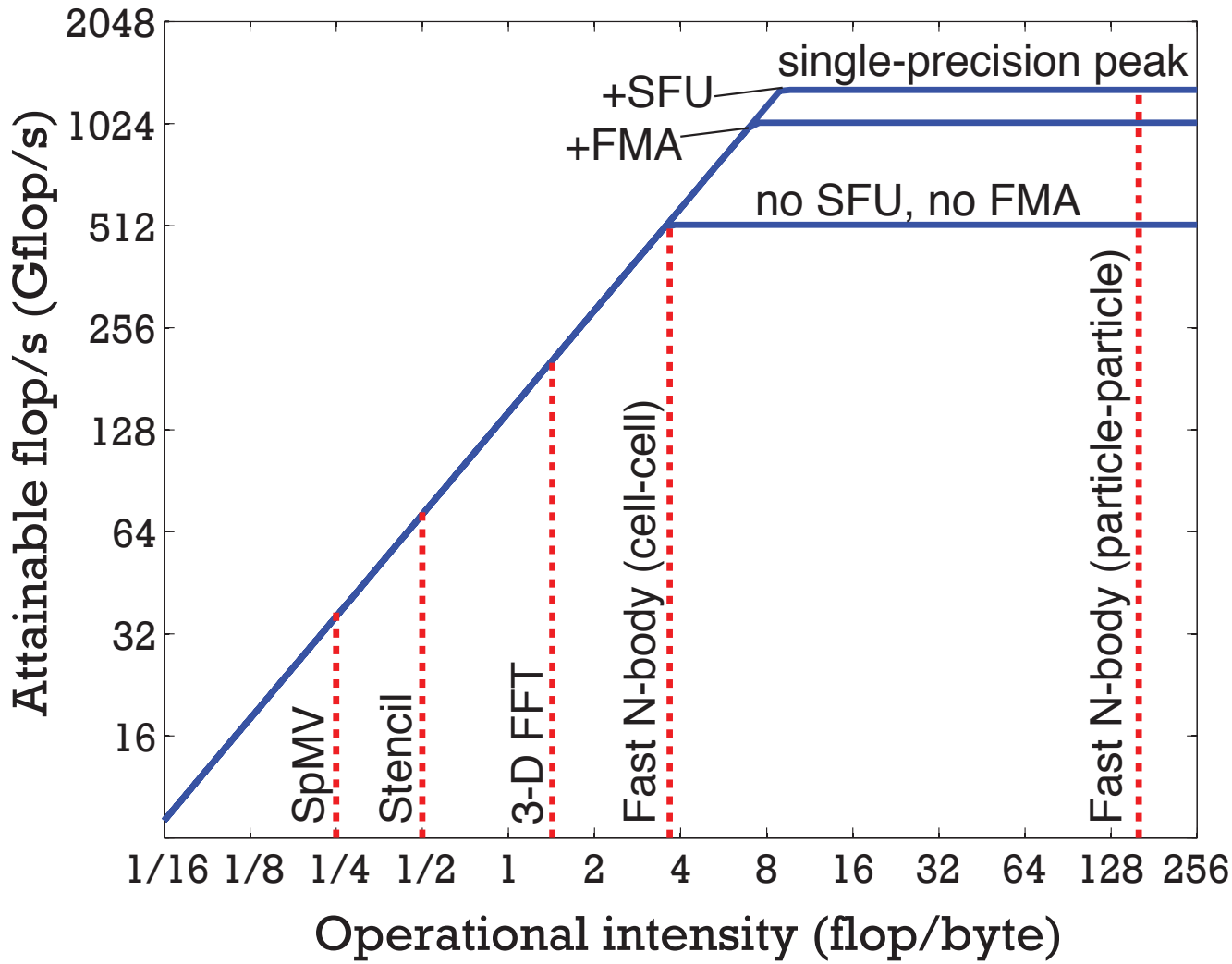
"Roofline: An Insightful Visual Performance Model for Multicore Architectures", S. Williams, A. Waterman, D. Patterson. *Communictions of the ACM*, April 2009.

Roofline model

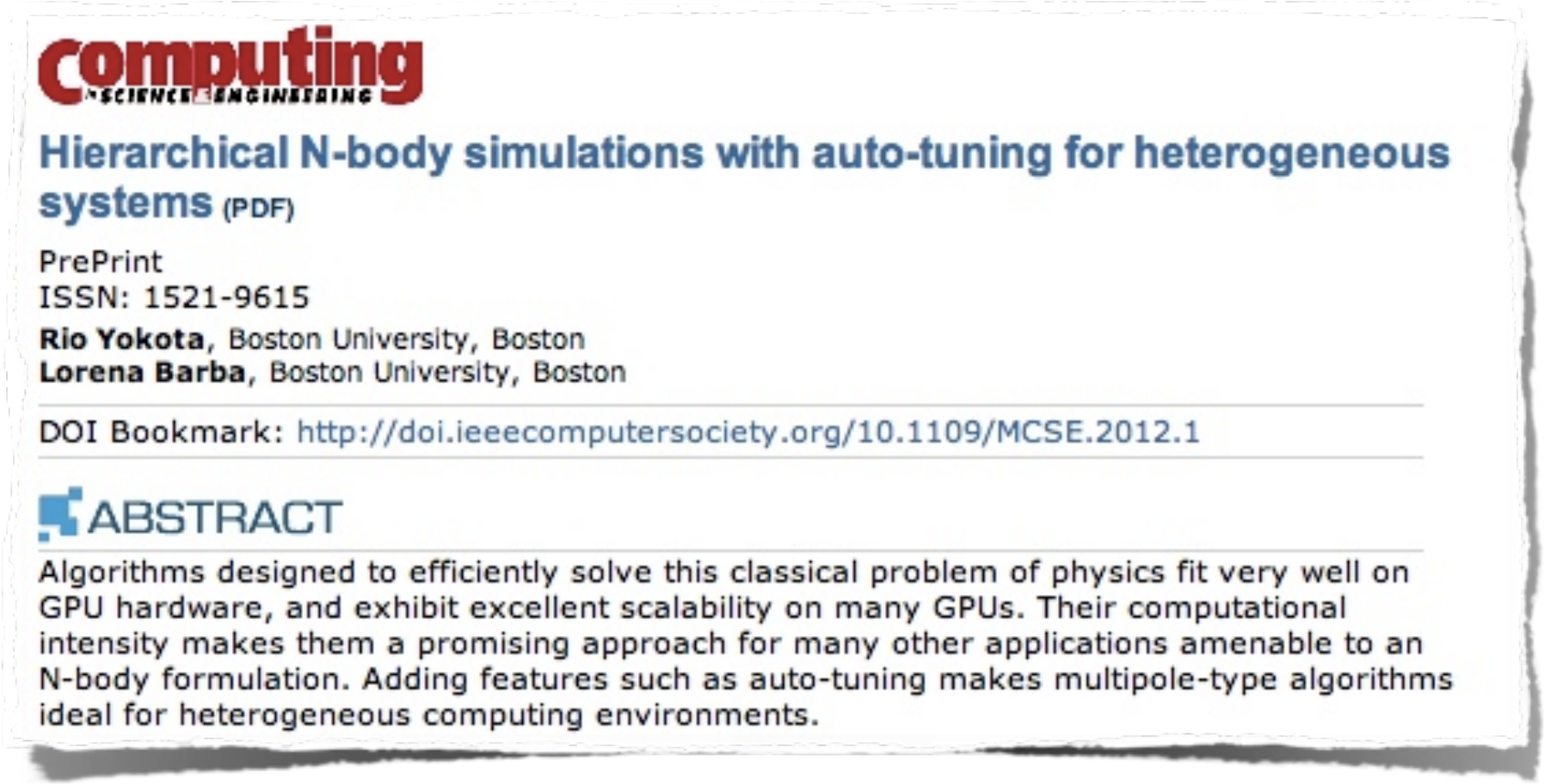
- Operational intensity = total flop / total byte = Gflop/s / GB/s



Advantage of N-body algorithms on GPUs



"Hierarchical N-body simulations with auto-tuning for heterogeneous systems", Rio Yokota, L A Barba. *IEEE Computing in Science and Engineering*, 3 January 2012



Computing in SCIENCE & ENGINEERING

Hierarchical N-body simulations with auto-tuning for heterogeneous systems (PDF)

PrePrint
ISSN: 1521-9615

Rio Yokota, Boston University, Boston
Lorena Barba, Boston University, Boston

DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/MCSE.2012.1

ABSTRACT

Algorithms designed to efficiently solve this classical problem of physics fit very well on GPU hardware, and exhibit excellent scalability on many GPUs. Their computational intensity makes them a promising approach for many other applications amenable to an N-body formulation. Adding features such as auto-tuning makes multipole-type algorithms ideal for heterogeneous computing environments.

Computing in Science and Engineering (CiSE), 3 January 2012, IEEE Computer Society, doi:10.1109/MCSE.2012.1.
Preprint arXiv:1108.5815

Scalability of FMM

in many-GPUs & many-CPU systems

Scalability of FMM

Our own progress so far:

1) 1 billion points on 512 GPUs (Degima)
2) 32 billion on 32,768 processors of Kraken
3) 69 billion on 4096 GPUs of Tsubame 2.0
   achieved **1 petaflop/s on turbulence simulation**

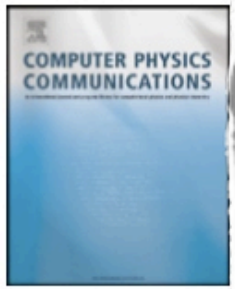
http://www.bu.edu/exafmm/

Computer Physics Communications 182 (2011) 1272–1283

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc



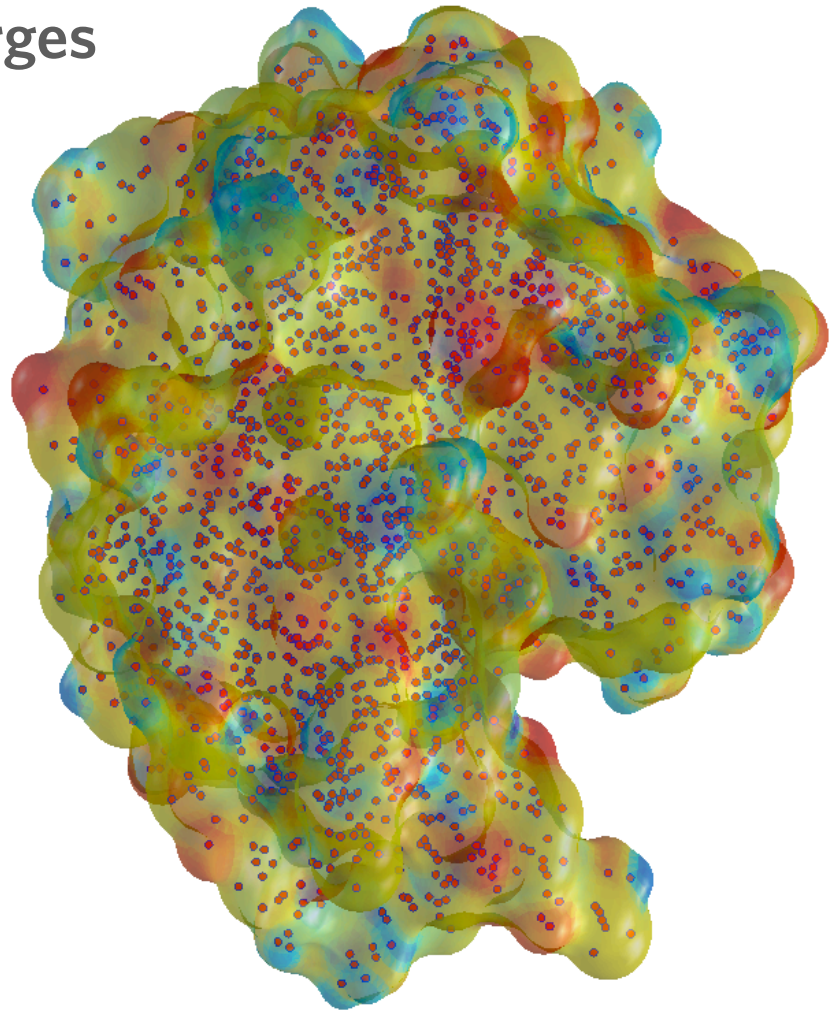
Biomolecular electrostatics using a fast multipole BEM on up to 512 GPUs and a billion unknowns

Rio Yokota [a], Jaydeep P. Bardhan [b], Matthew G. Knepley [c], L.A. Barba [a,*], Tsuyoshi Hamada [d]

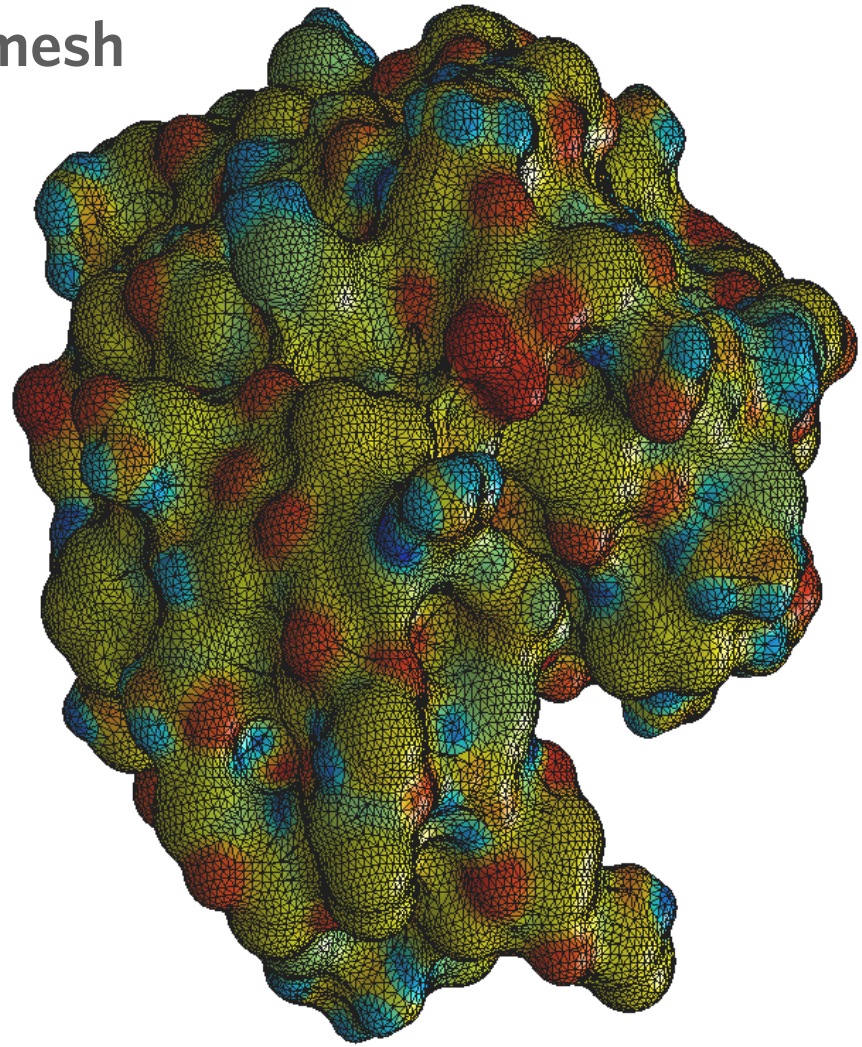
[a] *Department of Mechanical Engineering, Boston University, Boston, MA 02215, United States*
[b] *Dept. of Molecular Biophysics and Physiology, Rush University Medical Center, Chicago, IL 60612, United States*
[c] *Computation Institute, University of Chicago, Chicago, IL 60637, United States*
[d] *Nagasaki University, Advanced Computing Center (NACC), Nagasaki, Japan*

Demo: **Lysozyme molecule**

charges

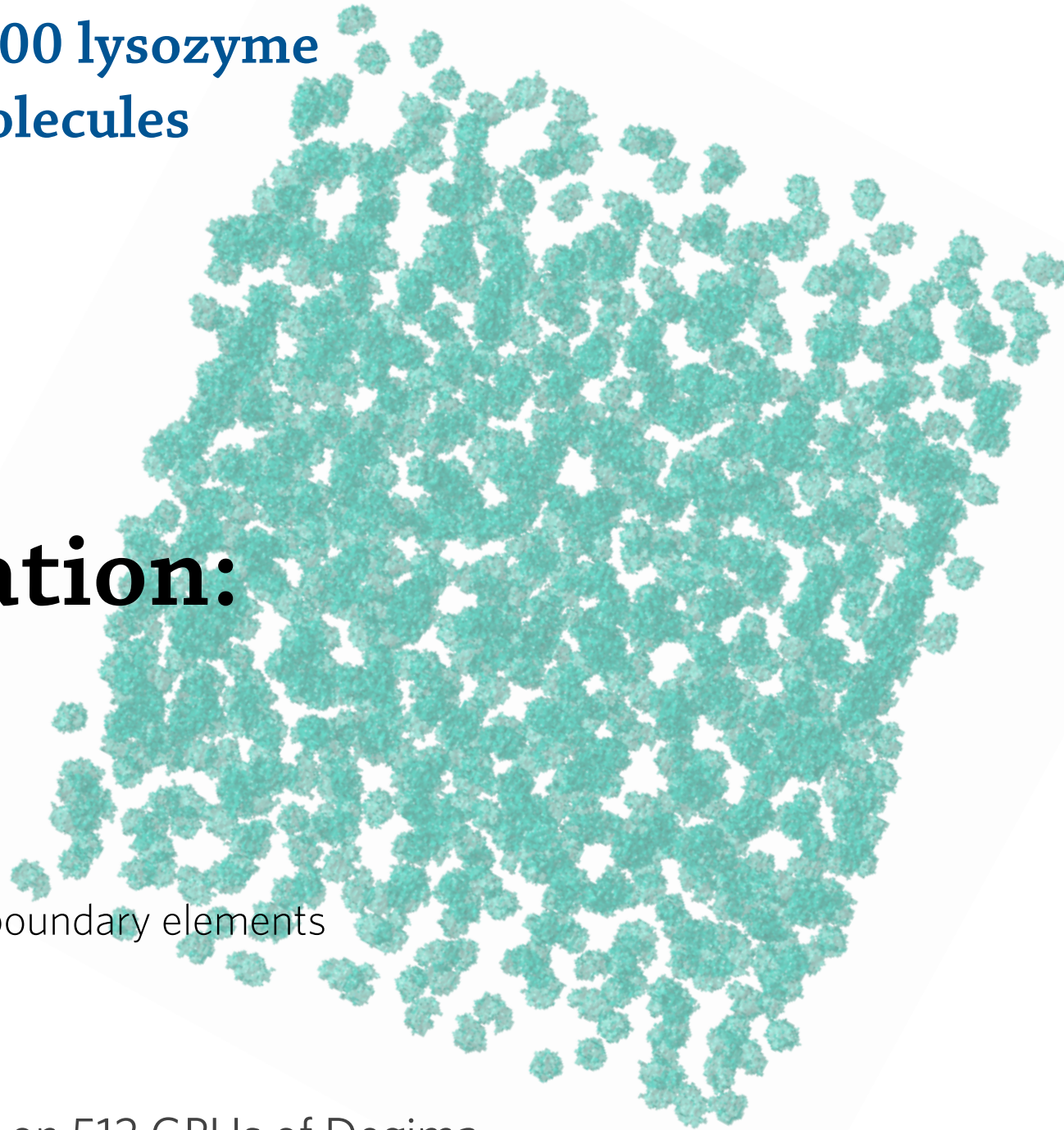


mesh



discretized with 102,486 boundary elements

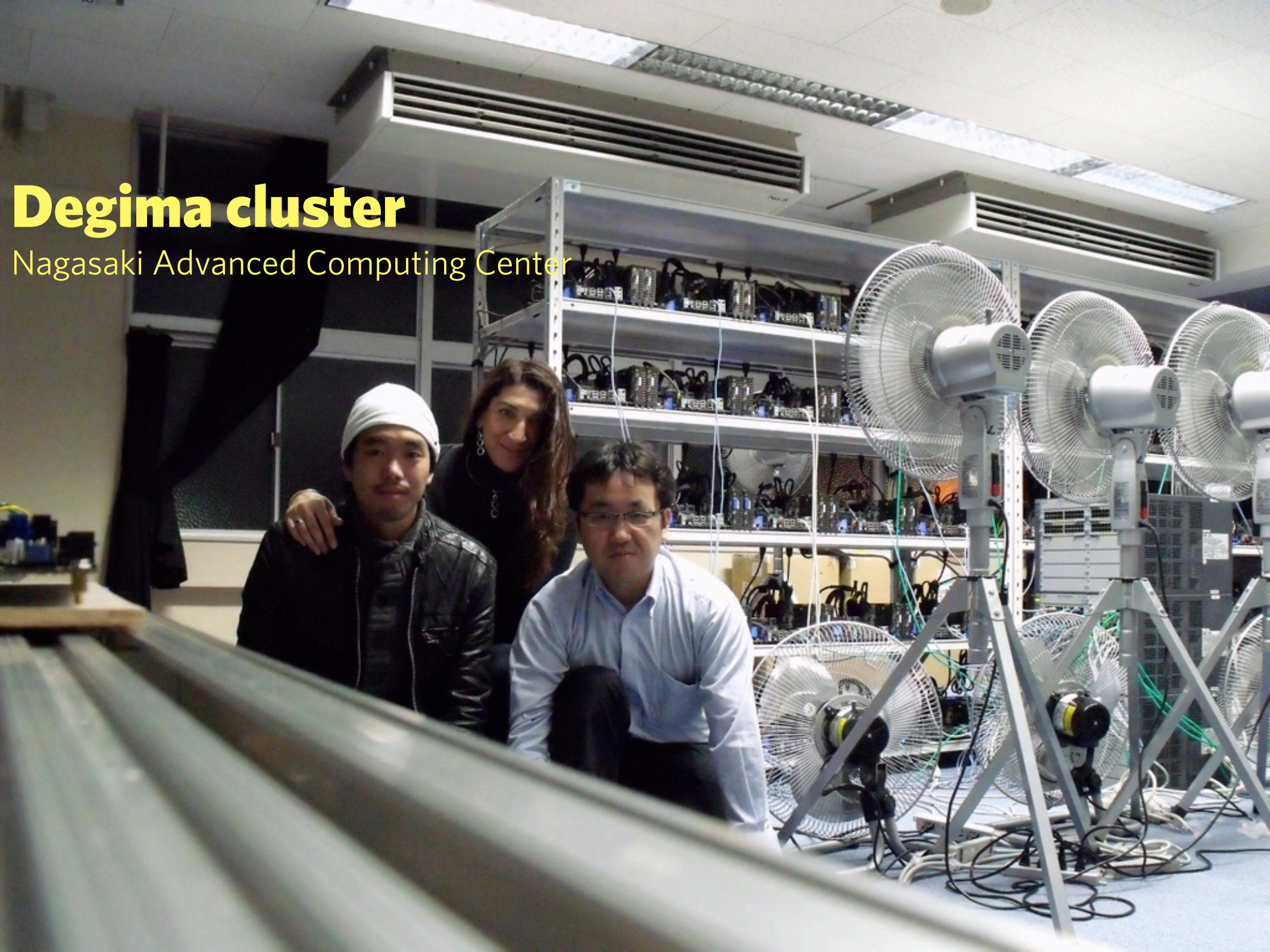
1000 lysozyme molecules



largest calculation:

- 10,648 molecules
- each discretized with 102,486 boundary elements
- more than 20 million atoms
- **1 billion unknowns**

⟶ one minute per iteration on 512 GPUs of Degima


Degima cluster
Nagasaki Advanced Computing Center

Kraken

Cray XT5 system at NICS, Tennessee:
9,408 nodes with 12 CPU cores each,
16 GB memory

peak performance is 1.17 Petaflop/s.
 # 11 in Top500 (Jun'11 & Nov'11)

TeraGrid

Weak scaling on Kraken

p=3
$N=10^6$, per process

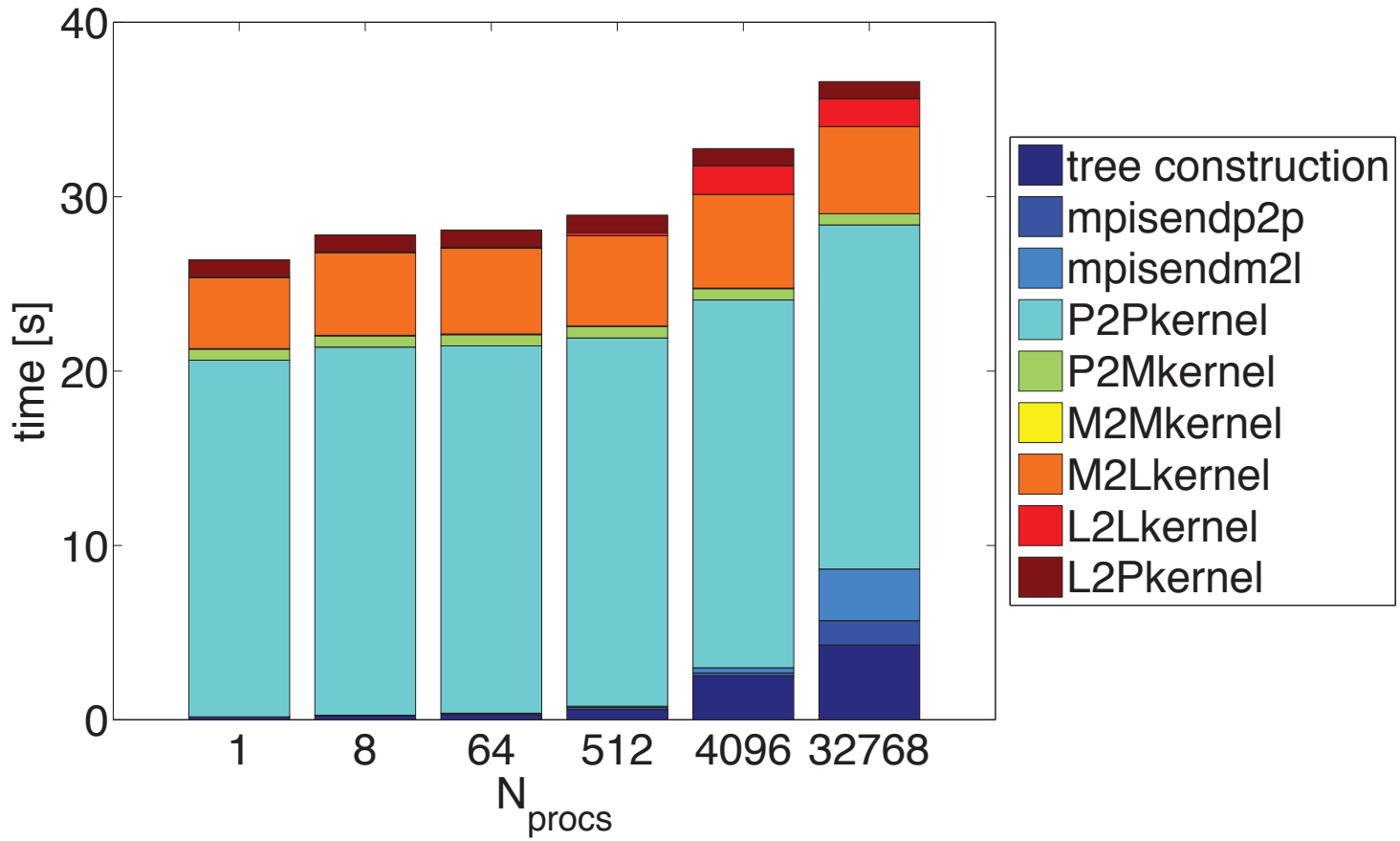
parallel efficiency
72% at 32,768
processors

largest run:

32 Billion points

time to solution:

<40 s



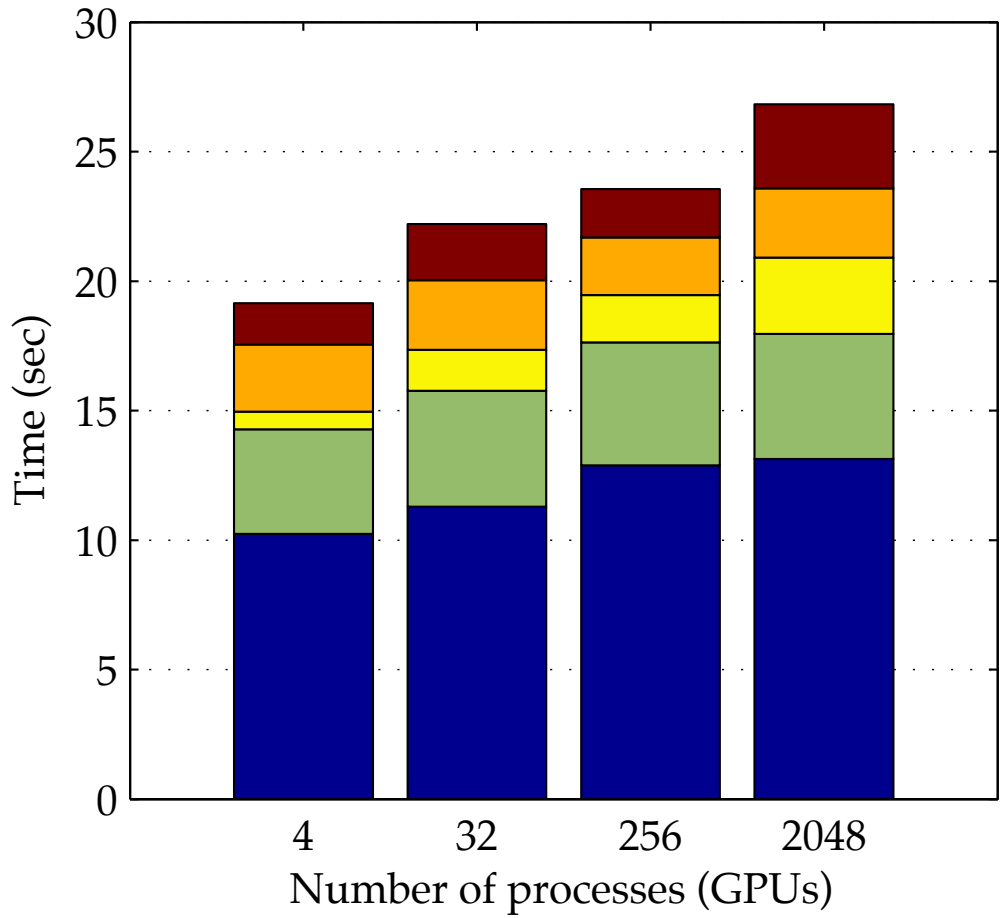
Tsubame 2.0

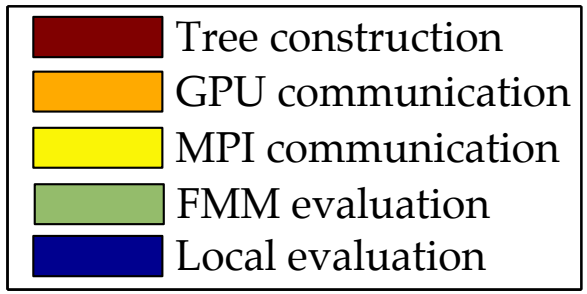
1408 nodes with 12 CPU cores each,
3 NVIDIA M2050 GPUs,
54 GB of RAM.

Total of 4224 GPUs
peak performance 2.4 Petaflop/s.
 # 5 in Top500 (Jun'11 & Nov'11)

Weak scaling on Tsubame

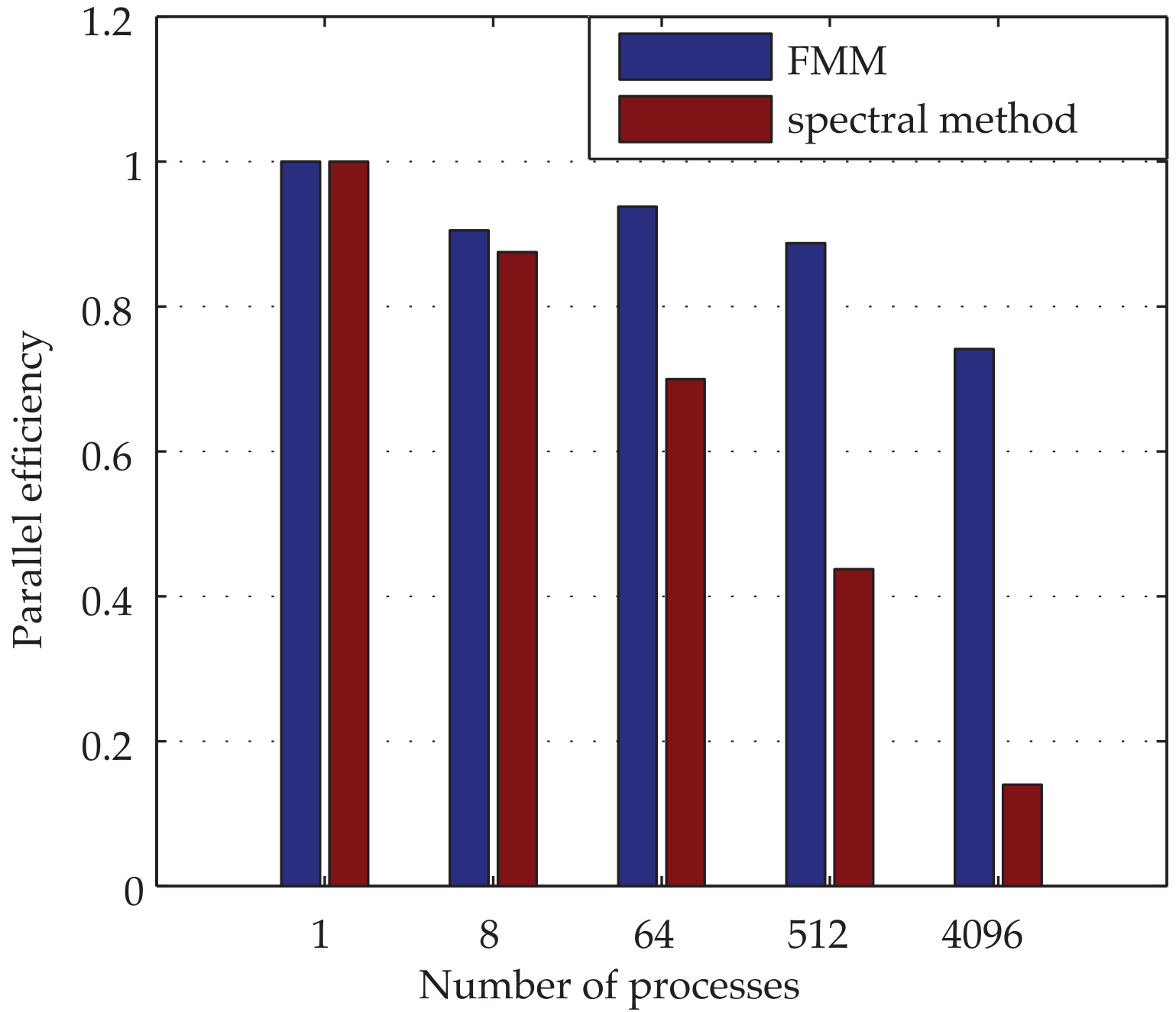
- 4 million points per process





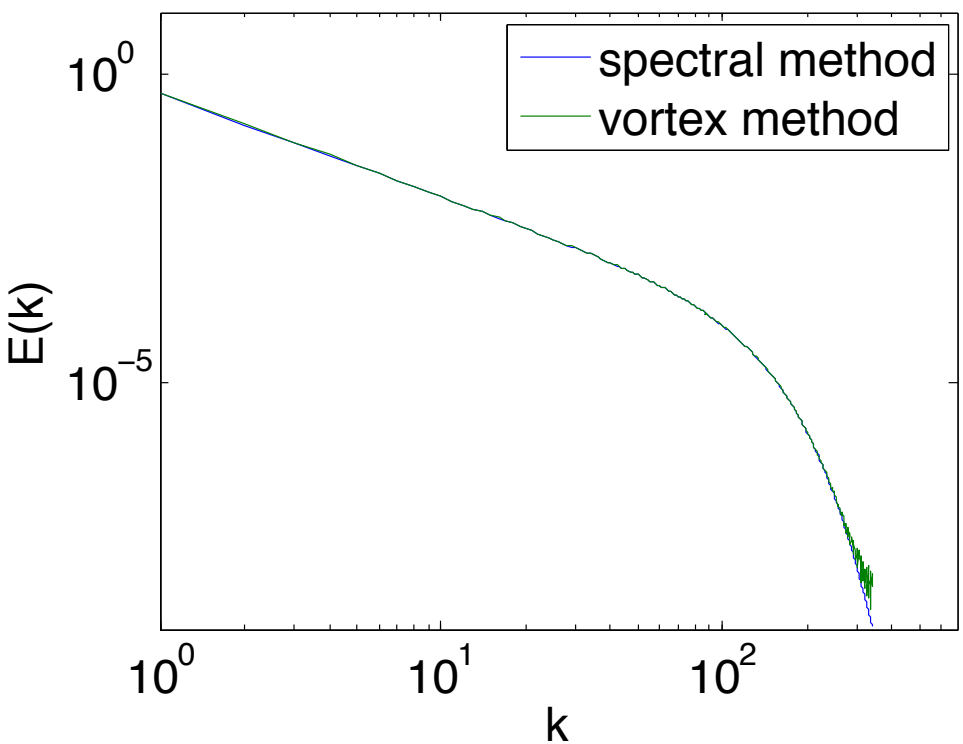
~9 billion points,
30s on 2048 GPUs

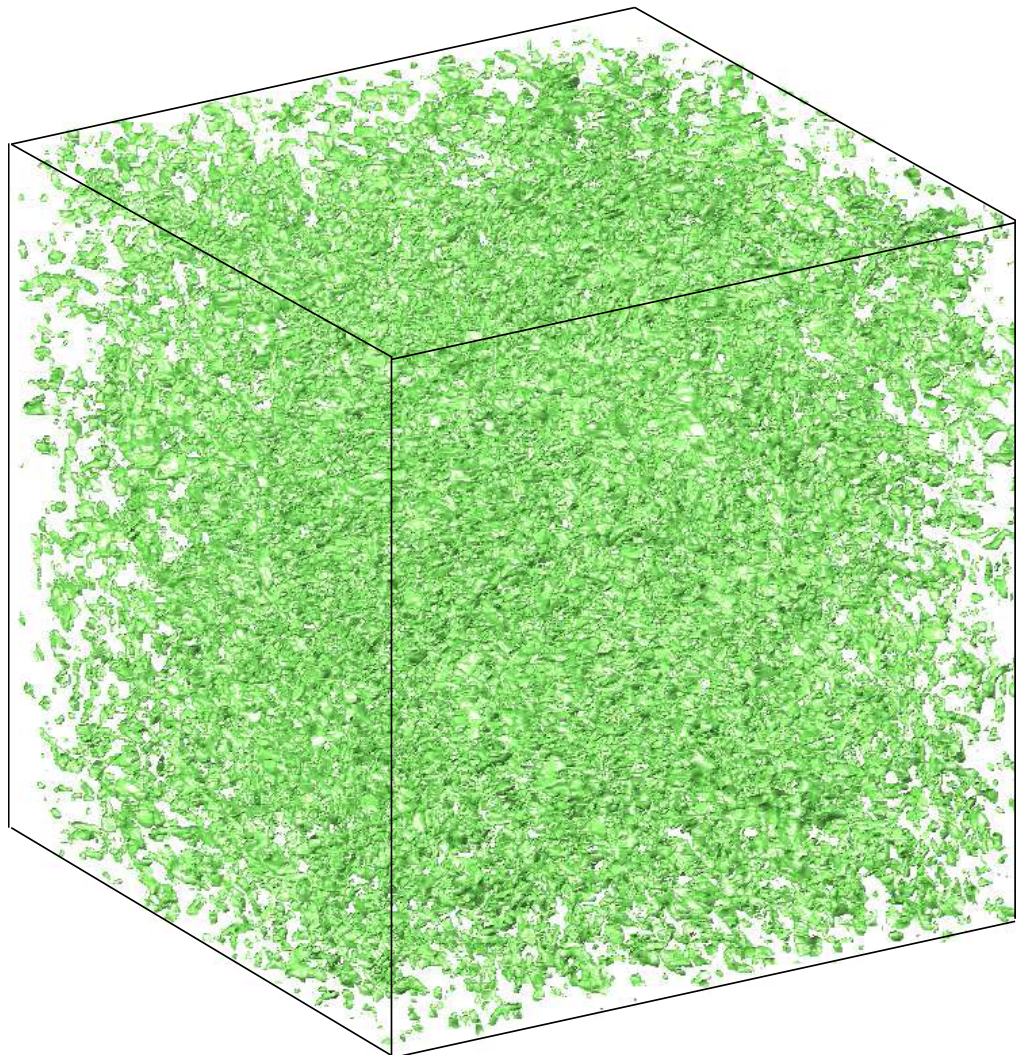
FMM vs. FFT, weak scaling



Petascale turbulence simulation

- using vortex method
- $4{,}096^3$ grid, **69 billion** points
- **1 Pflop/s**
- Energy spectrum well-matched





14 Gordon Bell awards for N-body

Gordon Bell awards for N-body

- Performance 1992 — Warren & Salmon, 5 Gflop/s
  - Price/performance 1997 — Warren et al., 18 Gflop/s / $1 M
  - Price/performance 2009 — Hamada et al., 124 Mflop/s / $1
- Performance 2010 — Rahimian et al., 0.7 Pflop/s on Jaguar

34x more than Moore's law

6200x cheaper

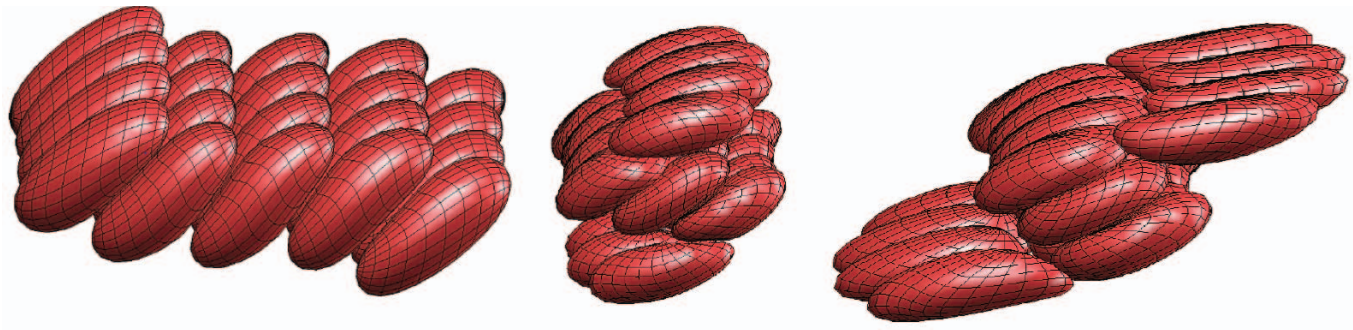
Petascale direct numerical simulation of blood flow on 200K cores and heterogeneous architectures

Abtin Rahimian*, Ilya Lashuk*, Shravan K. Veerapaneni†, Aparna Chandramowlishwaran*
Dhairya Malhotra*, Logan Moon*, Rahul Sampath‡, Aashay Shringarpure*,
Jeffrey Vetter‡, Richard Vuduc*, Denis Zorin†, and George Biros*

Petascale direct numerical simulation of blood flow on 200K cores and heterogeneous architectures

Abtin Rahimian*, Ilya Lashuk*, Shravan K. Veerapaneni†, Aparna Chandramowlishwaran*
Dhairya Malhotra*, Logan Moon*, Rahul Sampath‡, Aashay Shringarpure*,
Jeffrey Vetter‡, Richard Vuduc*, Denis Zorin†, and George Biros*

- largest simulation — ***90 billion unknowns***
- scale — 256 GPUs of Lincoln cluster / 196,608 cores of Jaguar
- numerical engine: FMM (kernel-independent version, 'kifmm')



How will FMM fare in exascale?

Pressure from hardware

- Overall byte-to-flop ratios —machine balance— are declining
- Machine balance is converging

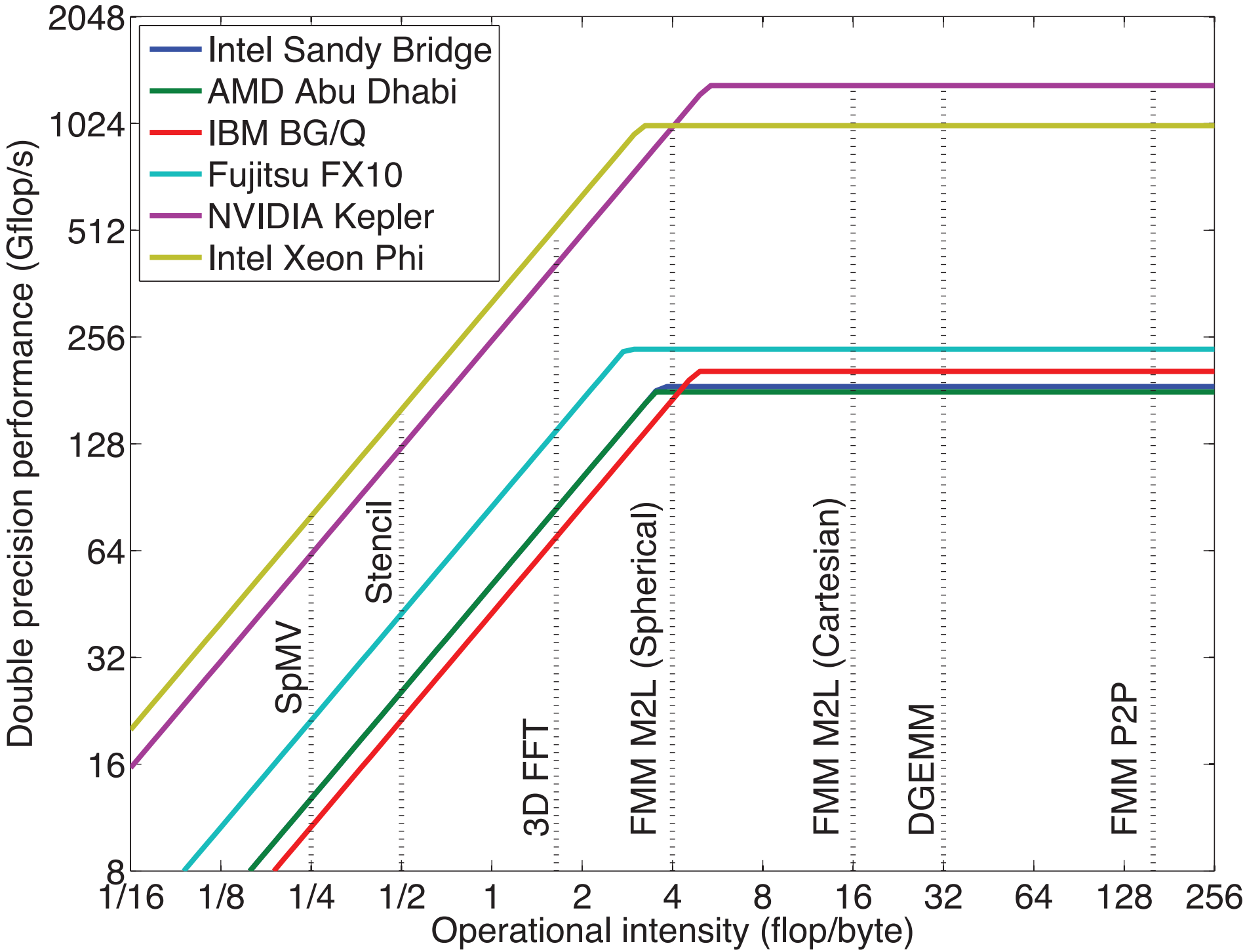
Vendor	Microarchitecture	Model	Byte/s	Flop/s	Byte/flop
Intel	Sandy Bridge	Xeon E5-2690	51.2	243.2	0.211
AMD	Bulldozer	Opteron 6284 SE	51.2	217.6	0.235
AMD	Southern Islands	Radeon HD7970 (GHz Ed.)	288	1010	0.285
NVIDIA	Fermi GF110	Tesla M2090	177	665	0.266
IBM	PowerPC	PowerPC A2 (BG/Q)	42.6	204.8	0.208
Fujitsu	SPARC64	SPARC64 IXfx (FX10)	85	236.5	0.359

Features of FMM for exascale

- The tree data structure extracts data locality from nonuniform and multi-scale resolutions;
- benign synchronization requirements;
- some kernels are purely local
- hierarchichal communication pattern

➡ The FMM is **not** a locality-sensitive application

In the sense of: Bergman et al. (2008) “Exascale Computing Study”, DARPA IPTO


Intel Sandy Bridge
AMD Abu Dhabi
IBM BG/Q
Fujitsu FX10
NVIDIA Kepler
Intel Xeon Phi
Double precision performance (Gflop/s)
2048
1024
512
256
128
64
32
16
8
1/16
1/8
1/4
1/2
1
2
4
8
16
32
64
128
256
Operational intensity (flop/byte)
SpMV
Stencil
3D FFT
FMM M2L (Spherical)
FMM M2L (Cartesian)
DGEMM
FMM P2P

From *SIAM News*, Volume 46, Number 6, July/August 2013

CSE 2013

How Will the Fast Multipole Method Fare in the Exascale Era?

By Lorena A. Barba and Rio Yokota