Communication-Avoiding Algorithms for Linear Algebra and Beyond

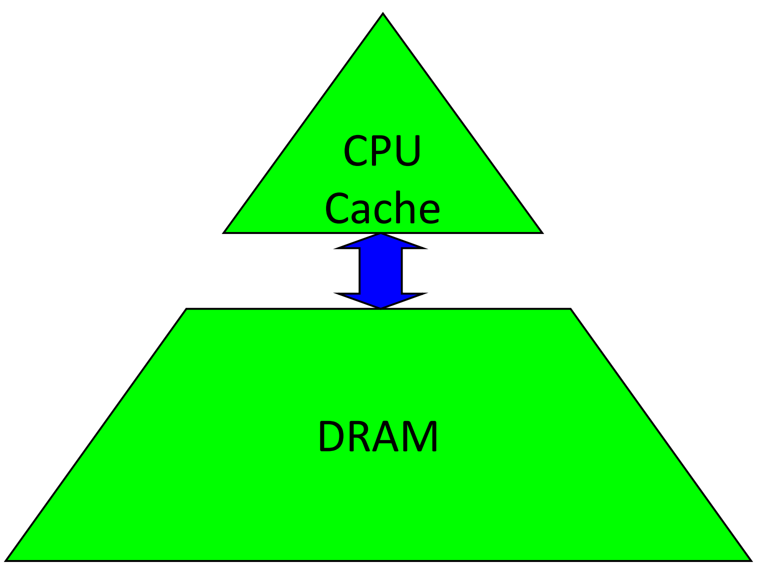
Jim Demmel, EECS & Math Depts, UC Berkeley

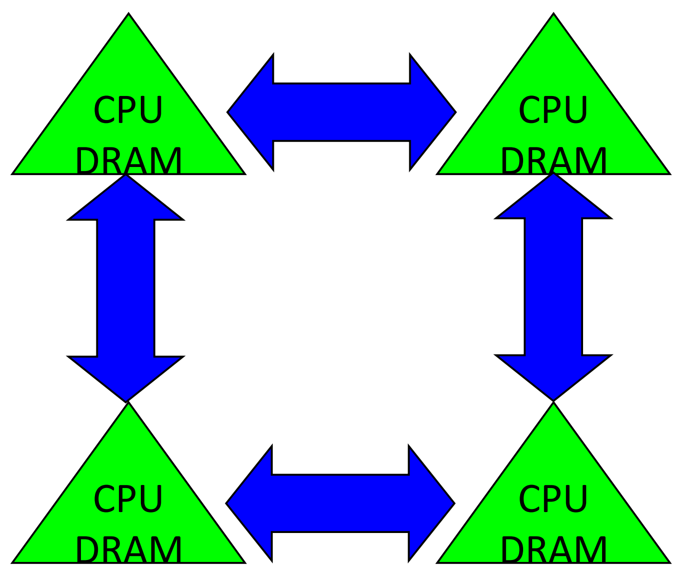
and many, many others …

Why avoid communication? (1/3)

Algorithms have two costs (measured in time or energy):

1. Arithmetic (FLOPS)
2. Communication: moving data between
   - levels of a memory hierarchy (sequential case)
   - processors over a network (parallel case).





Why avoid communication? (2/3)

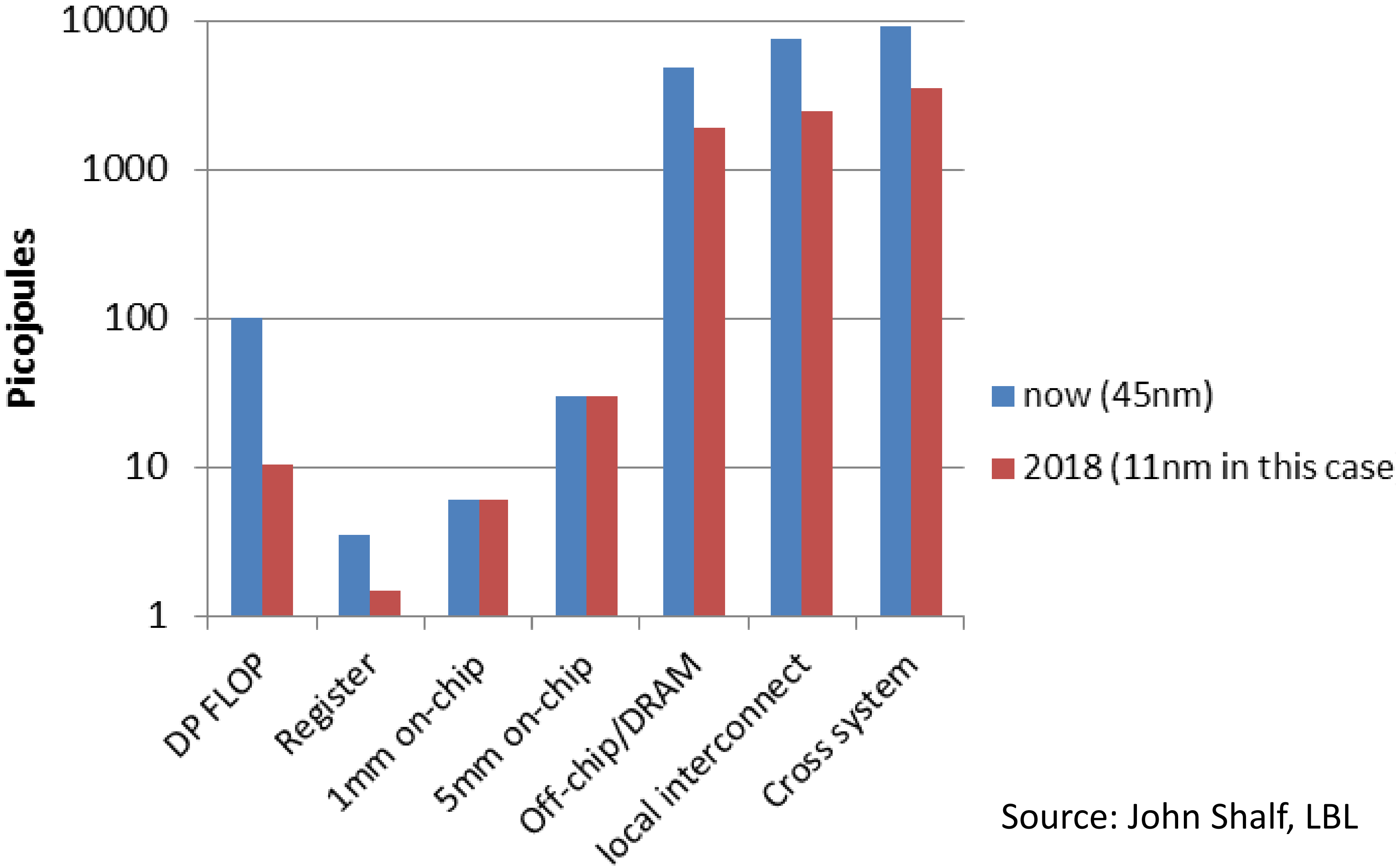
- Running time of an algorithm is sum of 3 terms:
  - # flops * time_per_flop
  - # words moved / bandwidth
  - # messages * latency

  communication
- Time_per_flop << 1/ bandwidth << latency
  - Gaps growing exponentially with time [FOSC]

Annual improvements			
Time_per_flop		Bandwidth	Latency
59%	Network	26%	15%
	DRAM	23%	5%

- Avoid communication to save time

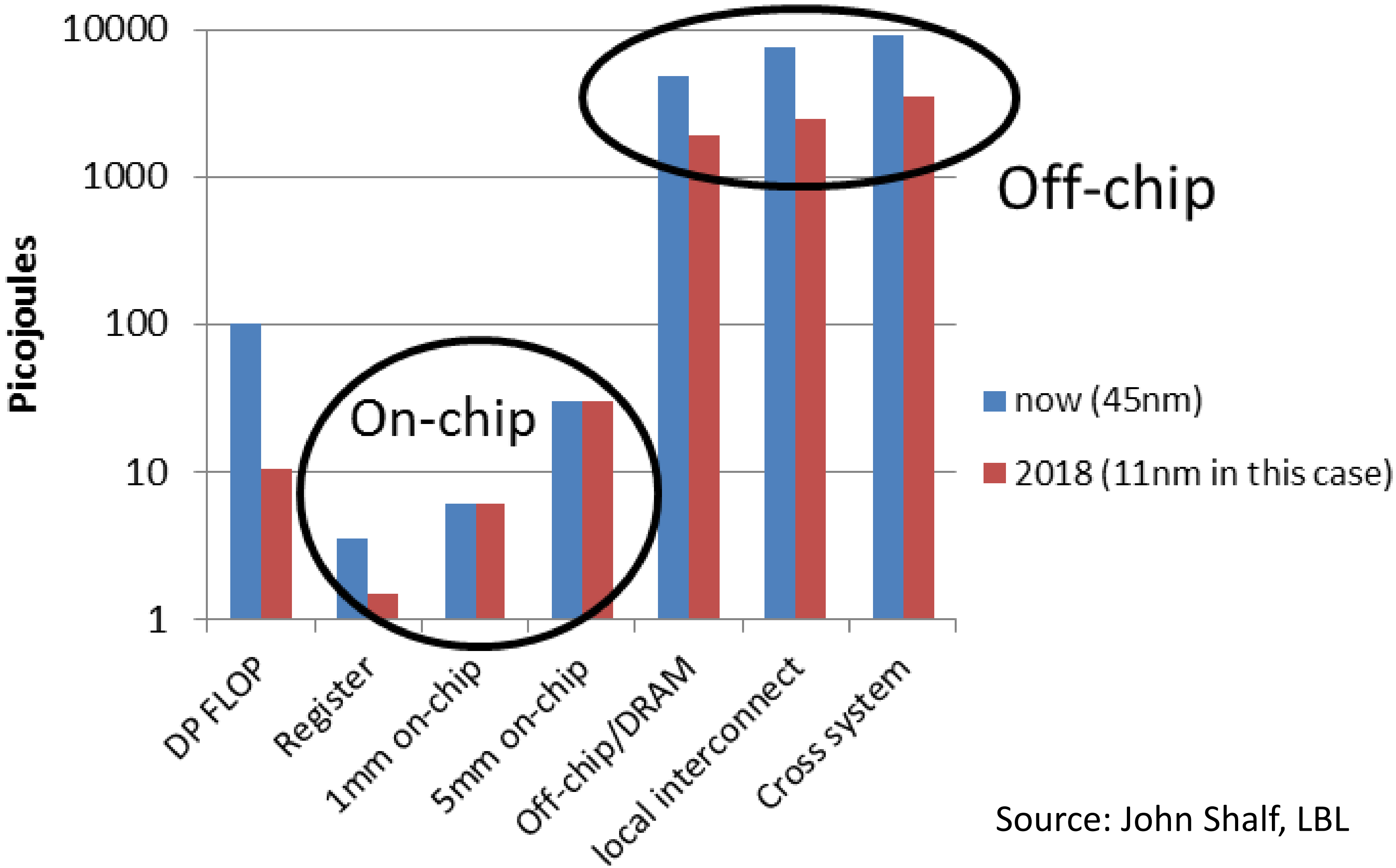
Why Minimize Communication? (3/3)



Source: John Shalf, LBL

Why Minimize Communication? (3/3)

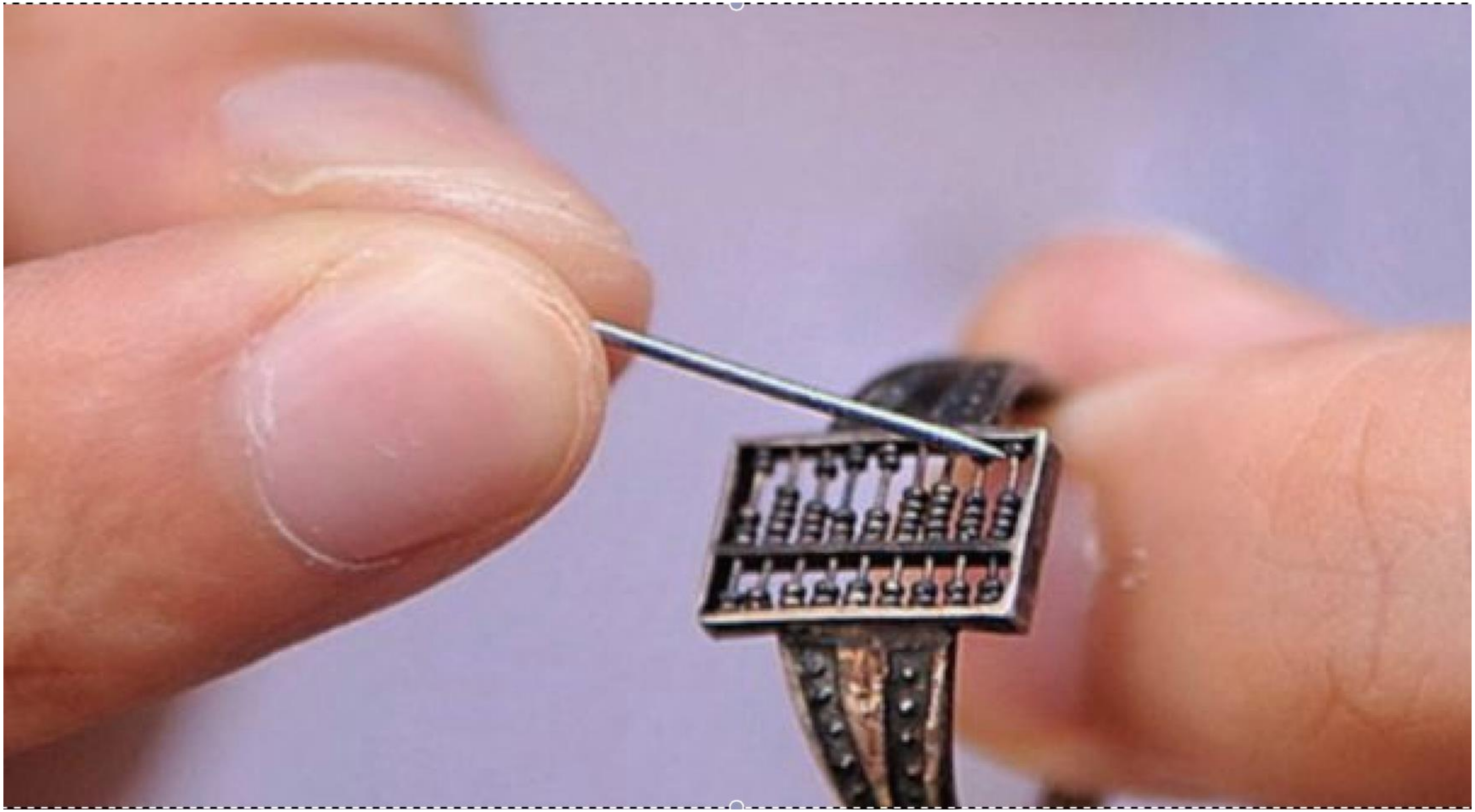
Minimize communication to save energy



Source: John Shalf, LBL

Alternative Cost Model for Algorithms?

Total distance moved by beads on an abacus



Goals

- Redesign algorithms to *avoid* communication
  - Between all memory hierarchy levels
    - L1 $\longleftrightarrow$ L2 $\longleftrightarrow$ DRAM $\longleftrightarrow$ network, etc
- Attain lower bounds if possible
  - Current algorithms often far from lower bounds
  - Large speedups and energy savings possible

Summary of CA Algorithms

- "Direct" Linear Algebra
  - Lower bounds on communication for linear algebra problems like Ax=b, least squares, $Ax = \lambda x$, SVD, etc
  - New algorithms that attain these lower bounds
    - Being added to libraries: Sca/LAPACK, PLASMA, MAGMA, …
    - Large speed-ups possible
  - Autotuning to find optimal implementation
- Ditto for programs accessing arrays (eg n-body)
- Ditto for "Iterative" Linear Algebra
- Ditto for Machine Learning

Sample Speedups

- Up to **12x** faster for 2.5D matmul on 64K core IBM BG/P

Ideas adopted by Nervana, "deep learning" startup, acquired by Intel in August 2016

- Up to **2.1x** faster for 2.5D LU on 64K core IBM BG/P
- Up to **11.8x** faster for direct N-body on 32K core IBM BG/P
- Up to **13x** faster for Tall Skinny QR on Tesla C2050 Fermi NVIDIA GPU

SIAG on Supercomputing Best Paper Prize, 2016
Released in LAPACK 3.7, Dec 2016

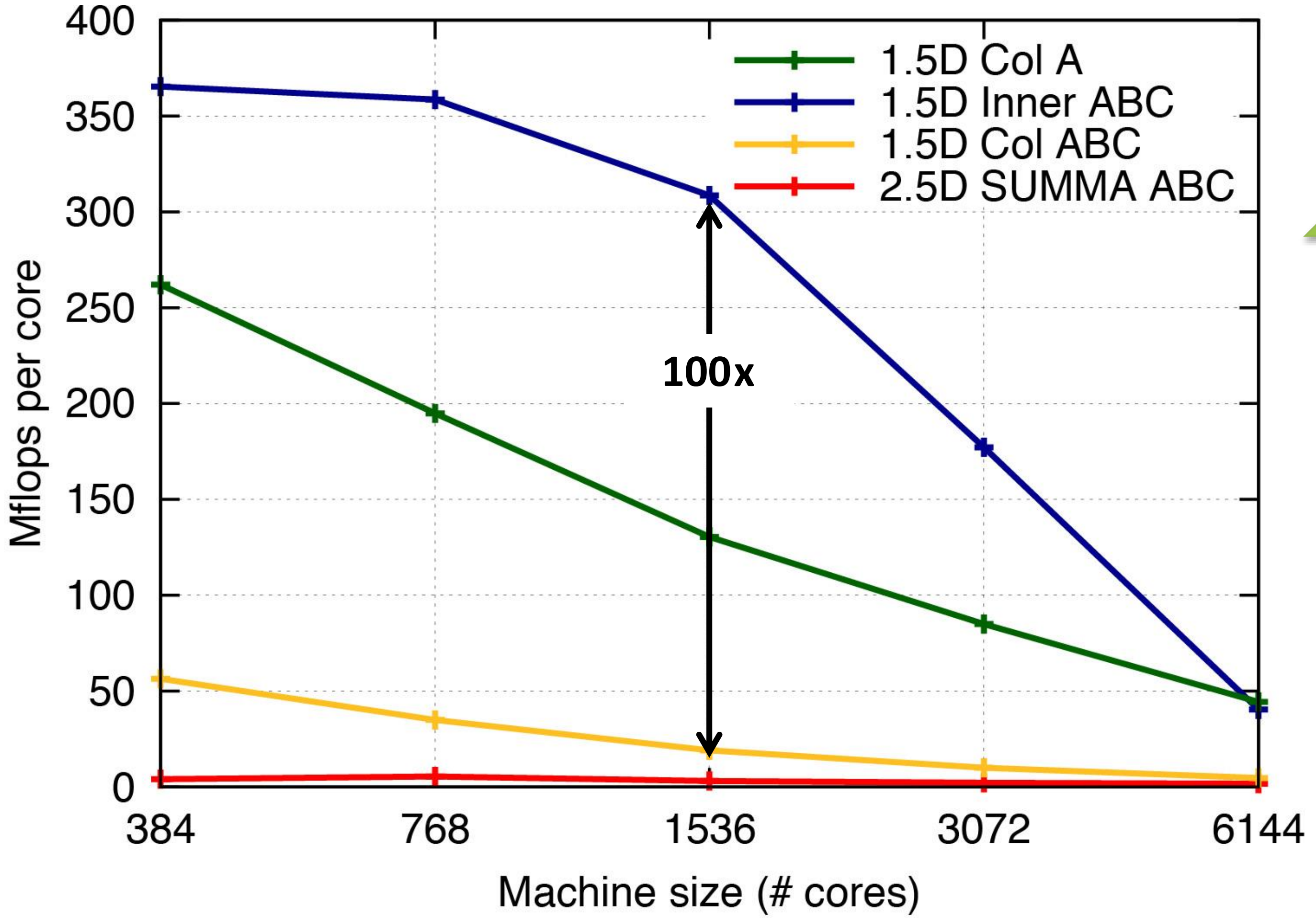
- Up to **4.2x** faster for MiniGMG benchmark bottom solver, using CA-BiCGStab (**2.5x** for overall solve) on 32K core Cray XE6
  - **2.5x** / **1.5x** for combustion simulation code
- Up to **5.1x** faster for coordinate descent LASSO on 3K core Cray XC30
- Up to **100x** faster for Sparse-Dense MM (in ML) on 10K core Cray XC30
- Fastest 100-epoch ImageNet/AlexNet training (**11 min**) on 1024 cores

CA Iterative Methods

- Linear Algebra – solving $Ax=b$, $Ax=\lambda x$
  - Classical algorithm: Repeat $x_i = A^*x_{i-1}$, compute "optimal" solution y_i in $V_i = \text{span}(x_1 , \ldots , x_i)$
  - CA version: form different basis of V_k with one communication, rearrange algebra to compute y_k
  - Can reduce communication by factor k
- Depends on linearity – what if it isn't linear?
- Machine learning – Coordinate Descent for LASSO,…
  - Can still rearrange algebra to take k steps at a time, but only latency goes down by k, BW and flops go up
  - Up to **5.1x** faster on 3K core Cray XC30
  - Aditya Devarakonda, Kimon Fountoulakis, Michael Mahoney, D.

100x Speedup on Sparse-Dense Matmul

- Bottleneck in some machine learning algorithms
- $A^{66k \times 172k}$, $B^{172k \times 66k}$, 0.0038% nnz, Cray XC30
- Penporn Koanantakool, Kathy Yelick



Some Other Activities

- Reproducible floating point summation & BLAS
  - New instructions to be added to next IEEE 754 Floating Point Standard
  - New routines to be added to next BLAS standard
- Precimonious
  - “Parsimonious with Precision”
  - Tool for automatically reducing floating point precision
- HipMer
  - First scalable parallel genome assembler
  - Human genome sequenced in 8.4 min on 15K cores

Awards

1. 2017 Householder Prize (Solomonik)
2. 2017 Householder Prize Finalist (Carson)
3. 2017 ACM-IEEE CS George Michael Memorial HPC Fellowship (You)
4. 2017 Member of the National Academy of Engineering (Yelick)
5. 2017 Member of the American Academy of Arts and Sciences (Yelick)
6. 2016 SIAG on Supercomputing Best Paper Prize (D., Grigori, Hoemmen)
7. 2015 Fellow of the Amer. Asso. Advancement of Science (D.)
8. 2015 IPDPS Best Paper Award (You, Czechowski, Song, Vuduc, D.)
9. 2014 David J. Sakrison Memorial Prize (Solomonik)
10. 2014 ACM Paris Kanellakis Theory and Practice Award (D.)
11. 2014 CACM Research Highlight (Ballard, Holtz, Schwartz, D.)
12. 2013 ACM Doctoral Dissertation Honorable Mention (Ballard)
13. 2013 ACM-IEEE CS George Michael Memorial HPC Fellowship (Solomonik)
14. 2013 IPDPS Charles Babbage Award (D.)
15. 2013 IPDPS Best Paper Award (Algorithms Track) (Becker, Ballard, D., et al)
16. 2012 SIAM Linear Algebra Prize (Ballard, Holtz, Schwartz, D.)
17. 2011 Distinguished Paper Prize EuroPar’11 (Solomonik, D.)
18. 2011 Member of the National Academy of Sciences (D.)

Collaborators and Supporters

- **James Demmel**, **Kathy Yelick**, Aditya Devarakonda, David Dinh, Michael Driscoll, Penporn Koanantakool, Alex Rusciano
- Peter Ahrens, Michael Anderson, Grey Ballard, Austin Benson, Erin Carson, Maryam Dehnavi, David Eliahu, Andrew Gearhart, Evangelos Georganas, Mark Hoemmen, Shoaib Kamil, Nicholas Knight, Ben Lipshitz, Marghoob Mohiyuddin, Hong Diep Nguyen, Jason Riedy, Oded Schwartz, Edgar Solomonik, Omer Spillinger
- Abhinav Bhatele, Aydin Buluc, Michael Christ, Ioana Dumitriu, Armando Fox, David Gleich, Ming Gu, Jeff Hammond, Mike Heroux, Olga Holtz, Kurt Keutzer, Julien Langou, Xiaoye Li, Devin Matthews, Tom Scanlon, Michelle Strout, Sam Williams, Hua Xiang
- Jack Dongarra, Mark Gates, Jakub Kurzak, Dulceneia Becker, Ichitaro Yamazaki, …
- Sivan Toledo, Alex Druinsky, Inon Peled
- Greg Henry, Peter Tang
- Laura Grigori**,** Sebastien Cayrols, Simplice Donfack, Mathias Jacquelin, Amal Khabou, Sophie Moufawad, Mikolaj Szydlarski
- Members of ASPIRE, BEBOP, ParLab, CACHE, EASI, FASTMath, MAGMA, PLASMA
- Thanks to DOE, NSF, UC Discovery, INRIA, Intel, Microsoft, Mathworks, National Instruments, NEC, Nokia, NVIDIA, Samsung, Oracle
- bebop.cs.berkeley.edu

Summary

Time to redesign all
linear algebra, n-body, and machine learning
algorithms and software
(and compilers)

Don't Communic…