Communication-Avoiding Algorithms for Linear Algebra and Beyond

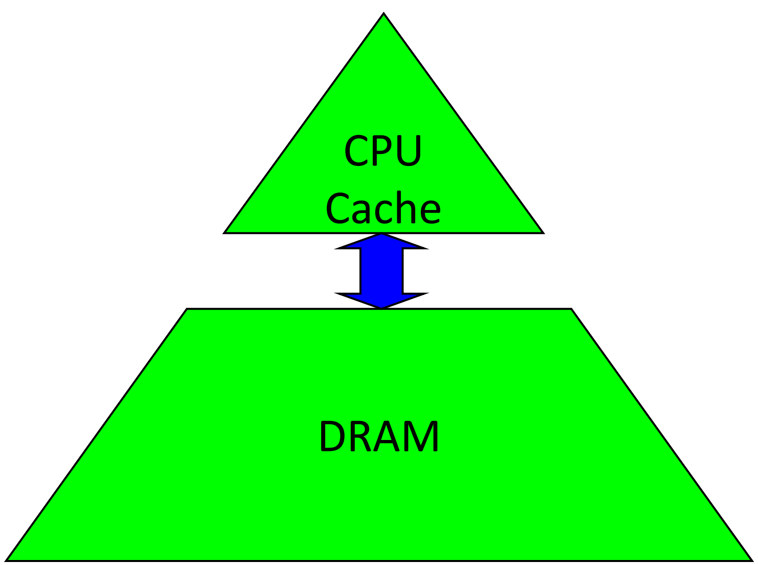
Longer version of slides available at:
www.cs.berkeley.edu/~demmel/SC16_tutorial_final

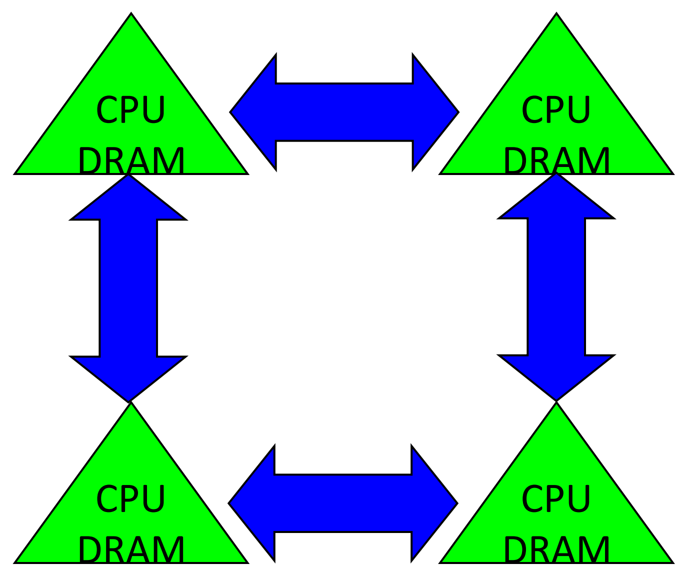
Jim Demmel, EECS & Math Depts, UC Berkeley
and many, many others …

Why avoid communication? (1/2)

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? (2/2)

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

  (# words moved / bandwidth and # messages * latency: communication)

- Time_per_flop << 1/ bandwidth << latency
  - Gaps growing exponentially with time
  - Avoid communication to save time
- Similar story for saving energy

Goals

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

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

Summary of CA Algorithms

- “Direct” Linear Algebra
  - Lower bounds on communication for linear algebra problems like Ax=b, least squares, Ax = λx, 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 => ML

Lower bound for all "direct" linear algebra

- Let M = "fast" memory size (per processor)

#words_moved (per processor) = Ω(#flops (per processor) / $M^{1/2}$)

- Parallel case: assume either load or memory balanced

- Holds for
  – Matmul

Lower bound for all "direct" linear algebra

- Let M = "fast" memory size (per processor)

#words_moved (per processor) = Ω(#flops (per processor) / $M^{1/2}$)

#messages_sent ≥ #words_moved / largest_message_size

- Parallel case: assume either load or memory balanced
- Holds for
  – Matmul, BLAS, LU, QR, eig, SVD, tensor contractions, …
  – Some whole programs (sequences of these operations, no matter how individual ops are interleaved, eg A^k)
  – Dense and sparse matrices (where #flops << n^3)
  – Sequential and parallel algorithms
  – Some graph-theoretic algorithms (eg Floyd-Warshall)

Lower bound for all "direct" linear algebra

- Let M = "fast" memory size (per processor)

#words_moved (per processor) = Ω(#flops (per processor) / $M^{1/2}$)

#messages_sent (per processor) = Ω(#flops (per processor) / $M^{3/2}$)

- Parallel case: assume either load or memory balanced
- Holds for
  - Matmul, BLAS, LU, QR, eig, SVD, tensor contractions, …
  - Some whole programs (sequences of these operations, no matter how individual ops are interleaved, eg A^k)

SIAM SIAG/Linear Algebra Prize, 2012

Ballard, D., Holtz, Schwartz

Approach to generalizing lower bounds

- Matmul

  for i=1:n, for j=1:n, for k=1:n, C(i,j)+=A(i,k)*B(k,j)

=> for (i,j,k) in S = subset of Z^3, access locations indexed by (i,j), (i,k), (k,j)

- Direct N-body

  for i=1:n, for j=1:n, F(i) += func(P(i), P(j))

=> for (i,j) in S = subset of Z^2, access locations indexed by (i), (j)

- More general case

  for i1=1:n, for i2 = i1:m, … for ik = i3:i4

  C(i1+2*i3-i7) = func(A(i2+3*i4,i1,i2,i1+i2,…),B(pnt(3*i4)),…)

  D(something else) = func(something else), …

=> for (i1,i2,…,ik) in S = subset of Z^k

  Access locations indexed by "projections", eg

  ϕ_C (i1,i2,…,ik) = (i1+2*i3-i7)

  ϕ_A (i1,i2,…,ik) = (i2+3*i4,i1,i2,i1+i2,…), …

- Goal: Communication lower bounds and optimal algorithms for *any* program that looks like this

General Communication Lower Bound

- Thm: Given a program with array refs given by projections ϕ_j, then there is an $e \geq 1$ such that

  #words_moved = Ω (#iterations/M^{e-1})

  where e is the the value of a linear program:

  minimize $e = \Sigma_j \, e_j$ subject to

  $\text{rank}(H) \leq \Sigma_j \, e_j * \text{rank}(\phi_j(H))$ for all subgroups $H < Z^k$

  – Proof depends on recent result in pure mathematics by Christ/Tao/Carbery/Bennett
- Thm: This lower bound is attainable, via loop tiling
  – Assumptions: dependencies permit, and iteration space big enough

Avoiding Communication in Iterative Linear Algebra

- k-steps of iterative solver for sparse Ax=b or Ax=λx
  - Does k SpMVs with A and starting vector
  - Many such "Krylov Subspace Methods"
- Goal: minimize communication
  - Assume matrix "well-partitioned"
  - Serial implementation
    - Conventional: O(k) moves of data from slow to fast memory
    - **New**: **O(1) moves of data – optimal**
  - Parallel implementation on p processors
    - Conventional: O(k log p) messages (k SpMV calls, dot prods)
    - **New: O(log p) messages - optimal**
- Lots of speed up possible (modeled and measured)
  - Price: some redundant computation
- Recent extensions to Machine Learning (SGD)

Other on-going work

- Extending “2.5D algorithms”
  - Replicate data to avoid more communication with dist. mem.
- Write-avoiding algorithms for Nonvolatile memories
  - With NVM, writes can be much more expensive than reads
  - Can sometimes do asymptotically fewer writes than reads
- Reproducibility
  - Roundoff makes floating point nonassociative, so different summation orders give different results
  - Have new algorithms that are reproducible, but still cost just one reduction operation, one pass over data
  - IEEE 754 Standard considering adding new instruction
  - BLAS Standard considering adding ReproBLAS

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

For more details

- Bebop.cs.berkeley.edu
  - 155 page survey in Acta Numerica (2014)
- CS267 – Berkeley's Parallel Computing Course
  - Live broadcast in Spring 2017
    - www.cs.berkeley.edu/~demmel
    - All slides, video available
  - Prerecorded version broadcast since Spring 2013
    - www.xsede.org
    - Free supercomputer accounts to do homework
    - Free autograding of homework

Summary

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

Don’t Communic…