Communication-Avoiding Algorithms for Linear Algebra and Beyond

Jim Demmel

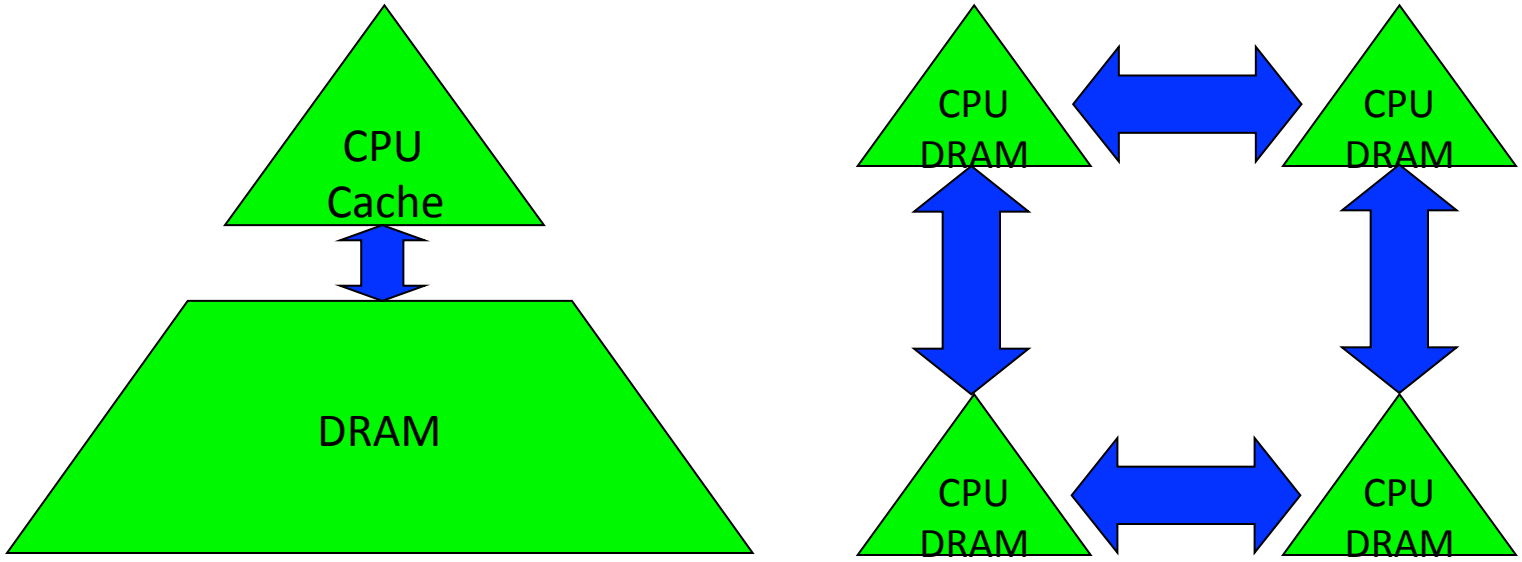
EECS & Math Departments

UC Berkeley

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

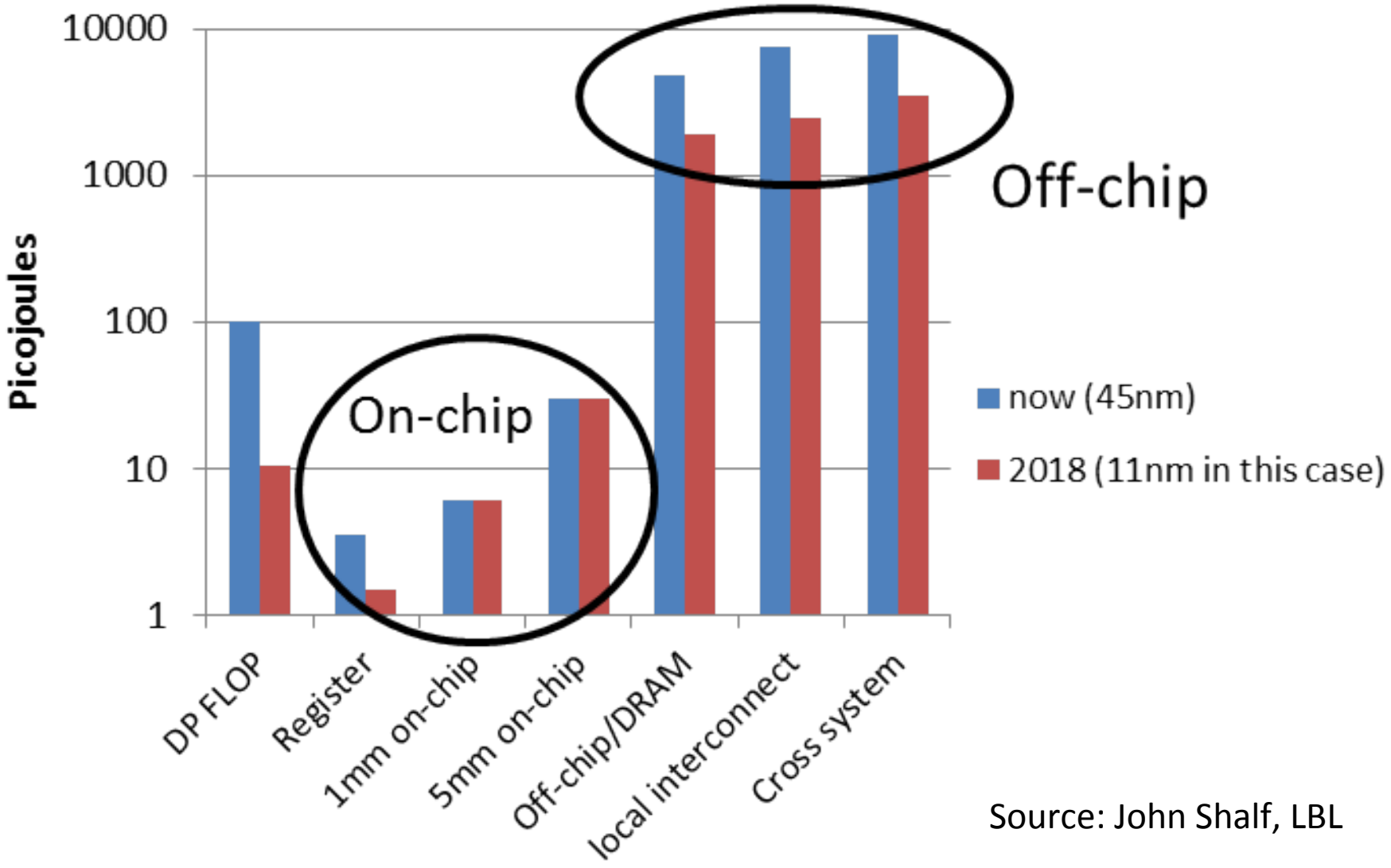
- Time_per_flop << 1/ bandwidth << latency
  - Gaps growing exponentially with time [FOSC]

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

- Avoid communication to save time

Why Minimize Communication? (3/3)

Minimize communication to save energy



Source: John Shalf, LBL

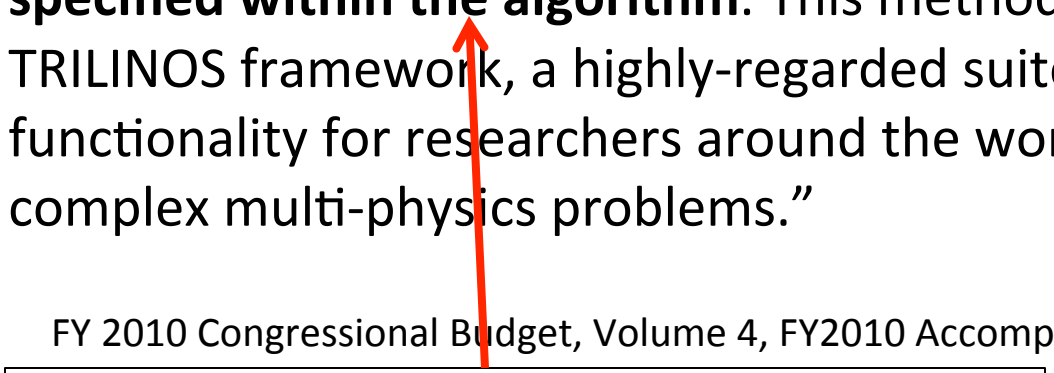
Goals

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

President Obama cites Communication-Avoiding Algorithms in the FY 2012 Department of Energy Budget Request to Congress:

"New Algorithm Improves Performance and Accuracy on Extreme-Scale Computing Systems. **On modern computer architectures, communication between processors takes longer than the performance of a floating point arithmetic operation by a given processor.** ASCR researchers have developed a new method, derived from commonly used linear algebra methods, to **minimize communications between processors and the memory hierarchy, by reformulating the communication patterns specified within the algorithm**. This method has been implemented in the TRILINOS framework, a highly-regarded suite of software, which provides functionality for researchers around the world to solve large scale, complex multi-physics problems."

FY 2010 Congressional Budget, Volume 4, FY2010 Accomplishments, Advanced Scientific Computing Research (ASCR), pages 65-67.



Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
  - CA O(n^3) 2.5D Matmul
  - CA Strassen Matmul
- Beyond linear algebra
  - Extending lower bounds to any algorithm with arrays
  - Communication-optimal N-body algorithm

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
  - CA O(n^3) 2.5D Matmul
  - CA Strassen Matmul
- Beyond linear algebra
  - Extending lower bounds to any algorithm with arrays
  - Communication-optimal N-body algorithm

Summary of CA Linear Algebra

- "Direct" Linear Algebra
  - Lower bounds on communication for linear algebra problems like Ax=b, least squares, Ax = λx, SVD, etc
  - Mostly not attained by algorithms in standard libraries
  - 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 "Iterative" Linear Algebra

Lower bound for all "n^3-like" 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 "n^3-like" 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

Summary of *dense* algorithms attaining communication lower bounds

- Do LAPACK and ScaLAPACK attain these bounds? Often not
- Assume nxn matrices on P processors
- Minimum Memory per processor = M = $O(n^2 / P)$
- Recall lower bounds:

  #words_moved = $\Omega((n^3/ P) / M^{1/2}) = \Omega(n^2 / P^{1/2})$

  #messages = $\Omega((n^3/ P) / M^{3/2}) = \Omega(P^{1/2})$
- When does ScaLAPACK attain these bounds?
  - For #words_moved: mostly, except Nonsym. Eigenproblem
  - For #messages: asymptotically worse, except Cholesky
- New algorithms attain all bounds, up to polylog(P) factors
  - Cholesky, LU, QR, Sym. and Nonsym eigenproblems, SVD
  - New numerical properties, new ways to encode answers, new data structures, not just loop transformations

Can we do Better?

Can we do better?

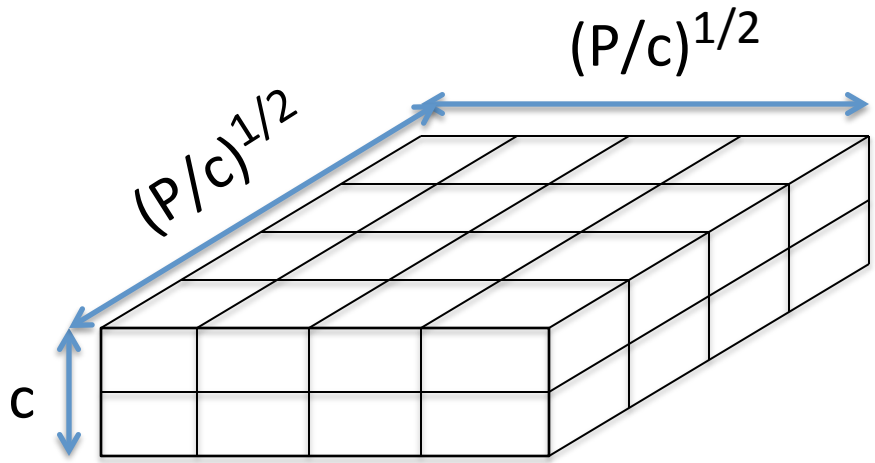
- Aren't we already optimal?
- Why assume $M = O(n^2/p)$, i.e. minimal?
  - Lower bound still true if more memory
  - Can we attain it?

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
  - CA O(n^3) 2.5D Matmul
  - CA Strassen Matmul
- Beyond linear algebra
  - Extending lower bounds to any algorithm with arrays
  - Communication-optimal N-body algorithm

2.5D Matrix Multiplication

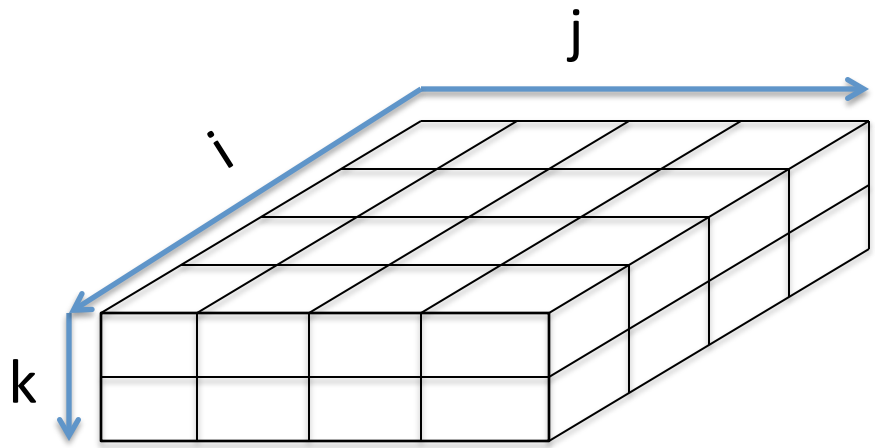
- Assume can fit cn^2/P data per processor, $c > 1$
- Processors form $(P/c)^{1/2}$ x $(P/c)^{1/2}$ x c grid



Example: P = 32, c = 2

2.5D Matrix Multiplication

- Assume can fit cn^2/P data per processor, $c > 1$
- Processors form $(P/c)^{1/2}$ x $(P/c)^{1/2}$ x c grid

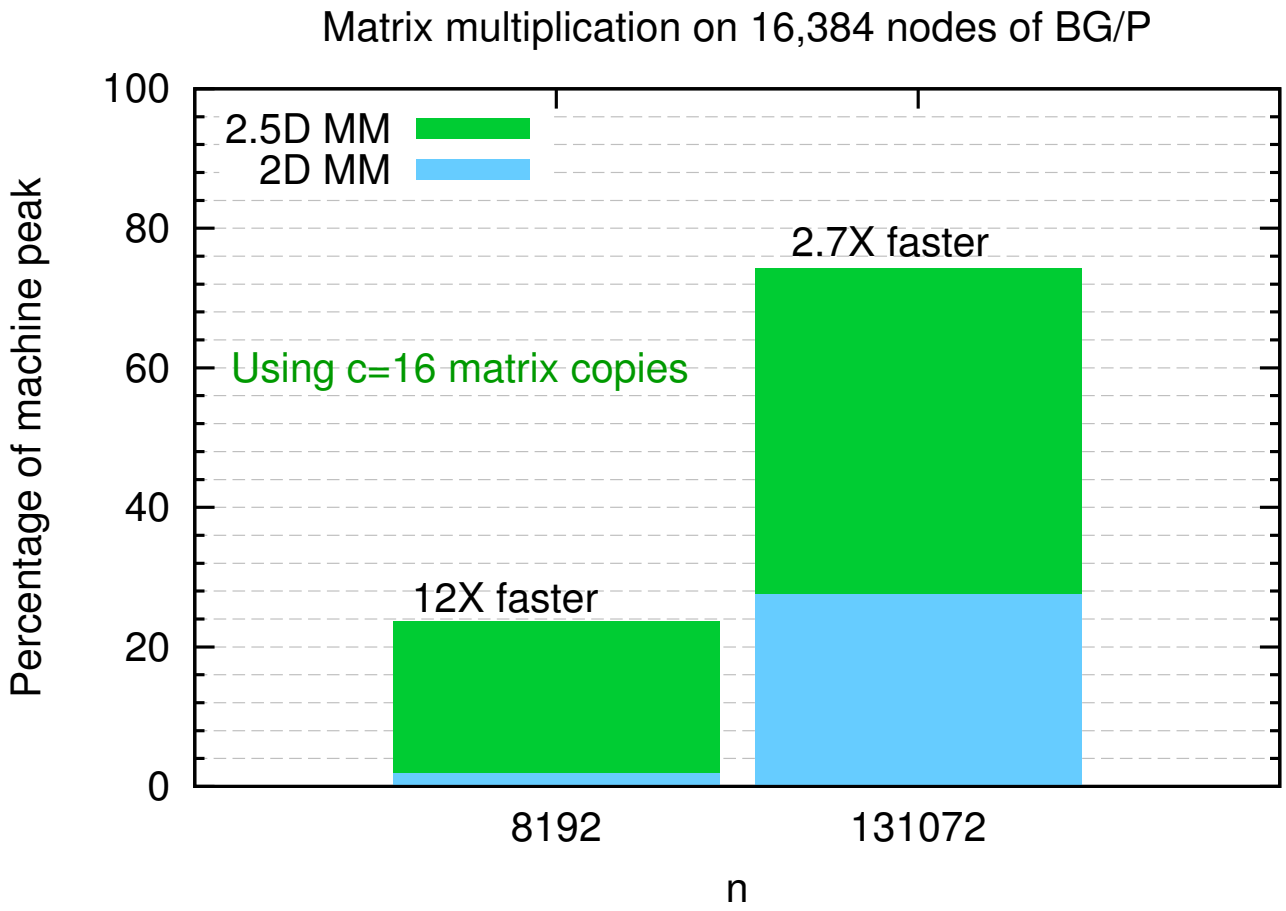


Initially P(i,j,0) owns A(i,j) and B(i,j)
each of size $n(c/P)^{1/2}$ x $n(c/P)^{1/2}$

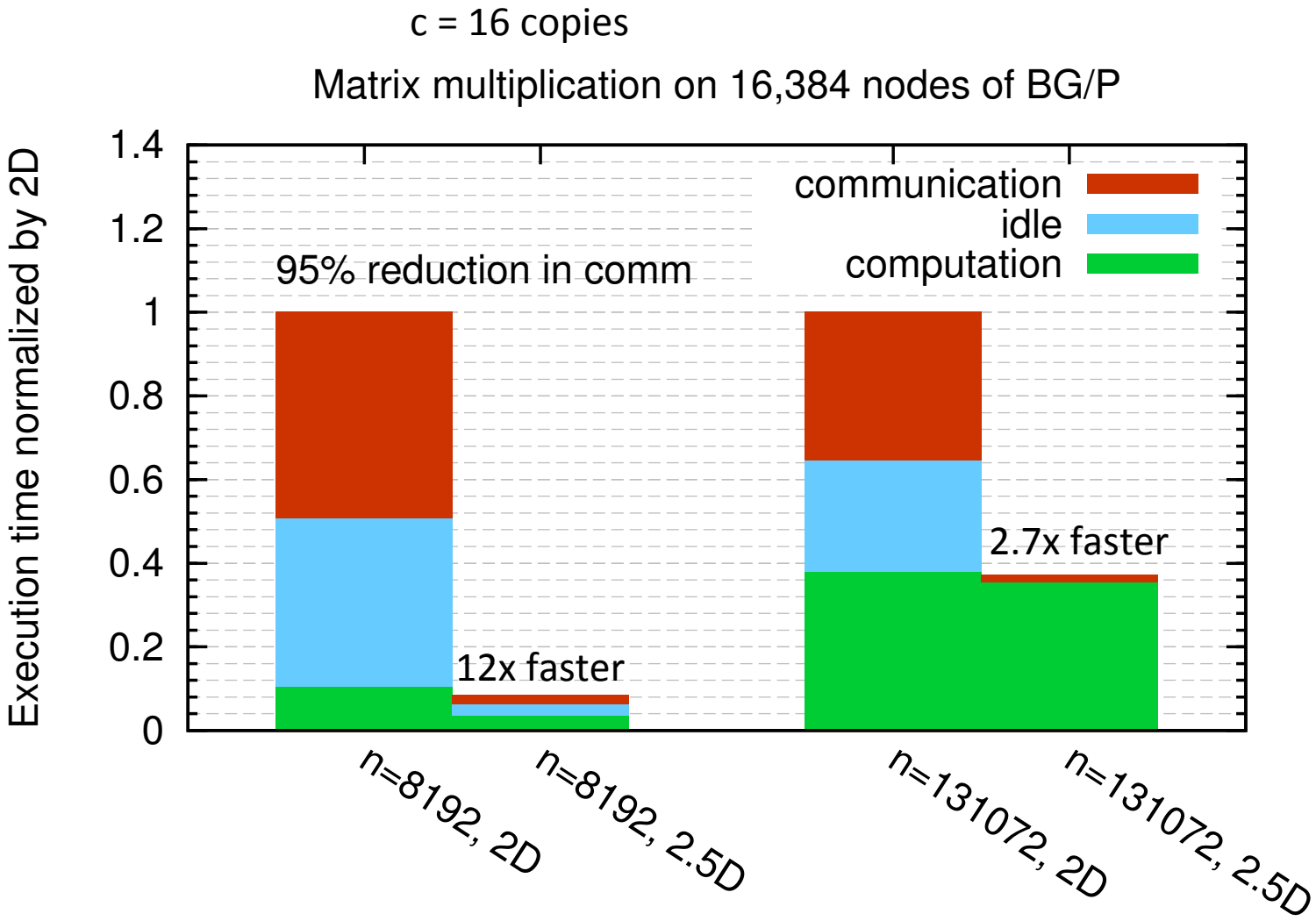
(1) P(i,j,0) broadcasts A(i,j) and B(i,j) to P(i,j,k)
(2) Processors at level k perform 1/c-th of SUMMA, i.e. 1/c-th of Σ_m A(i,m)*B(m,j)
(3) Sum-reduce partial sums Σ_m A(i,m)*B(m,j) along k-axis so P(i,j,0) owns C(i,j)

Thm: Works up to $c = P^{1/3}$, then stops

2.5D Matmul on BG/P, 16K nodes / 64K cores



2.5D Matmul on BG/P, 16K nodes / 64K cores



Distinguished Paper Award, EuroPar'11 (Solomonik, D.)

SC'11 paper by Solomonik, Bhatele, D.

Perfect Strong Scaling – in Time and Energy (1/2)

- Every time you add a processor, you should use its memory M too
- Start with minimal number of procs: $PM = 3n^2$
- Increase P by a factor of c ➔ total memory increases by a factor of c
- Notation for timing model:
  - γ_T , β_T , α_T = secs per flop, per word_moved, per message of size m
- $T(cP) = n^3/(cP) [\gamma_T + \beta_T/M^{1/2} + \alpha_T/(mM^{1/2})]$
  $= T(P)/c$
- Notation for energy model:
  - γ_E , β_E , α_E = joules for same operations
  - δ_E = joules per word of memory used per sec
  - ε_E = joules per sec for leakage, etc.
- $E(cP) = cP \{ n^3/(cP) [\gamma_E + \beta_E/M^{1/2} + \alpha_E/(mM^{1/2})] + \delta_E MT(cP) + \varepsilon_E T(cP) \}$
  $= E(P)$

Perfect Strong Scaling – in Time and Energy (2/2)

- $T(cP) = n^3/(cP) \left[\gamma_T + \beta_T/M^{1/2} + \alpha_T/(mM^{1/2}) \right] = T(P)/c$
- $E(cP) = cP \{ n^3/(cP) \left[\gamma_E + \beta_E/M^{1/2} + \alpha_E/(mM^{1/2}) \right] + \delta_E MT(cP) + \varepsilon_E T(cP) \} = E(P)$

- Perfect scaling extends to N-body, Strassen, …
- We can use these models to answer many questions, including:

- What is the minimum energy required for a computation?
- Given a maximum allowed runtime **T** , what is the minimum energy **E** needed to achieve it?
- Given a maximum energy budget **E** , what is the minimum runtime **T** that we can attain?
- The ratio **P** = **E/T** gives us the average power required to run the algorithm. Can we minimize the average power consumed?
- Given an algorithm, problem size, number of processors and target energy efficiency (GFLOPS/W), can we determine a set of architectural parameters to describe a conforming computer architecture?

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
  - CA O(n^3) 2.5D Matmul
  - CA Strassen Matmul
- Beyond linear algebra
  - Extending lower bounds to any algorithm with arrays
  - Communication-optimal N-body algorithm

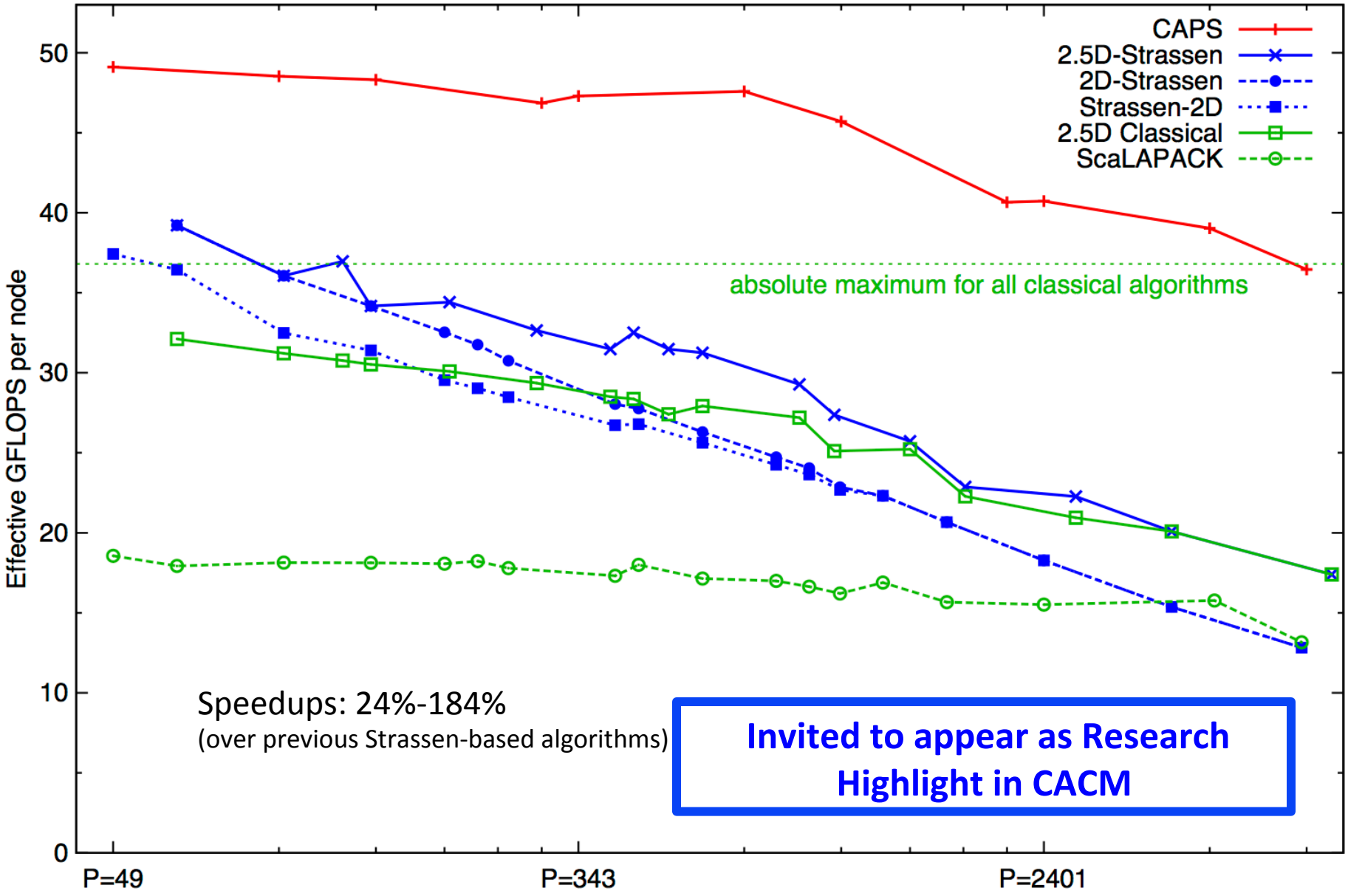
Communication Lower Bounds for Strassen-like matmul algorithms

Classical $O(n^3)$ matmul:	Strassen's $O(n^{lg7})$ matmul:	Strassen-like $O(n^{\omega})$ matmul:
#words_moved = $\Omega\,(M(n/M^{1/2})^3/P)$	#words_moved = $\Omega\,(M(n/M^{1/2})^{lg7}/P)$	#words_moved = $\Omega\,(M(n/M^{1/2})^{\omega}/P)$

- Proof: graph expansion (different from classical matmul)
  - Strassen-like: DAG must be "regular" and connected
- Extends up to $M = n^2 / p^{2/\omega}$
- Best Paper Prize (SPAA'11), Ballard, D., Holtz, Schwartz, also in JACM
- Is the lower bound attainable?

Performance Benchmarking, Strong Scaling Plot

Franklin (Cray XT4) n = 94080



Speedups: 24%-184%
(over previous Strassen-based algorithms)

Invited to appear as Research Highlight in CACM

Ongoing Work

- Lots more work on
  - Algorithms:
    - BLAS, LDL^T, QR with pivoting, other pivoting schemes, eigenproblems, …
    - All-pairs-shortest-path, …
    - Both 2D (c=1) and 2.5D (c>1)
    - But only bandwidth may decrease with c>1, not latency
      - New lower bound on bandwidth*latency
  - Platforms:
    - Multicore, cluster, GPU, cloud, heterogeneous, low-energy, …
  - Software:
    - Integration into Sca/LAPACK, PLASMA, MAGMA,…
- Integration into applications (on IBM BG/Q)
  - CTF (with ANL): symmetric tensor contractions

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
  - CA O(n^3) 2.5D Matmul
  - CA Strassen Matmul
- Beyond linear algebra
  - Extending lower bounds to any algorithm with arrays
  - Communication-optimal N-body algorithm

Recall optimal sequential Matmul

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

- "Blocked" code
  for i1 = 1:b:n, for j1 = 1:b:n, for k1 = 1:b:n
    for i2 = 0:b-1, for j2 = 0:b-1, for k2 = 0:b-1
      i=i1+i2, j = j1+j2, k = k1+k2
      C(i,j)+=A(i,k)*B(k,j)

  } b x b matmul

- Thm: Picking b = $M^{1/2}$ attains lower bound:
  #words_moved = $\Omega(n^3/M^{1/2})$
- Where does **1/2** come from?

New Thm applied to Matmul

- for i=1:n, for j=1:n, for k=1:n, C(i,j) += A(i,k)*B(k,j)
- Record array indices in matrix Δ

$$\Delta = \begin{array}{c} \begin{matrix} i & j & k \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ A \\ B \\ C \end{matrix}$$

- Solve LP for $x = [xi,xj,xk]^T$: $\max \mathbf{1}^T x$ s.t. $\Delta x \le \mathbf{1}$
  - Result: $x = [1/2, 1/2, 1/2]^T$, $\mathbf{1}^T x = 3/2 = s_{HBL}$
- Thm: #words_moved = $\Omega(n^3/M^{s_{HBL}-1}) = \Omega(n^3/M^{1/2})$

  Attained by block sizes $M^{xi},M^{xj},M^{xk} = M^{1/2},M^{1/2},M^{1/2}$

New Thm applied to Direct N-Body

- for i=1:n, for j=1:n, F(i) += force(P(i) , P(j))
- Record array indices in matrix Δ

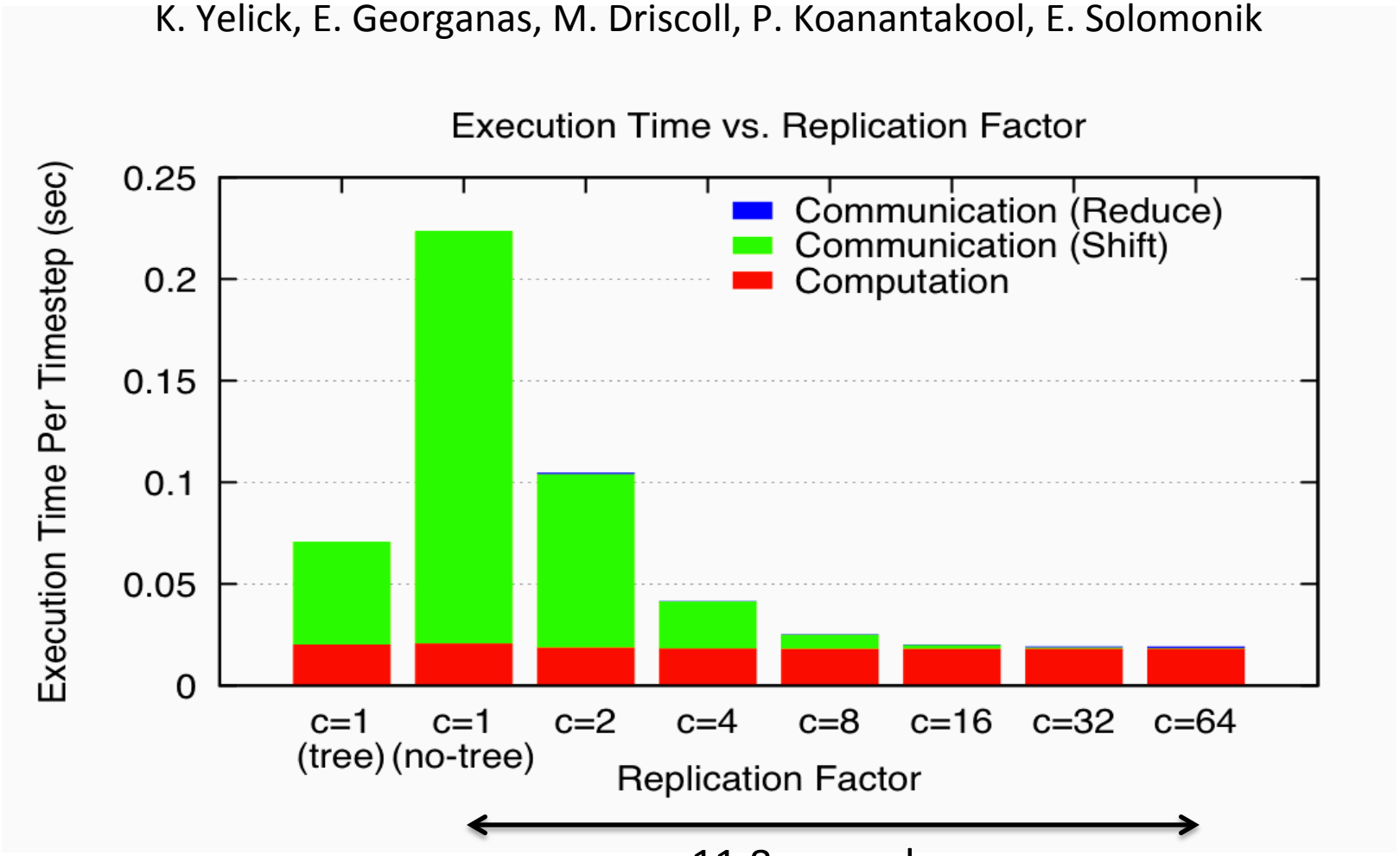
$$\Delta = \begin{array}{c} \begin{matrix} i & j \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array} \begin{matrix} F \\ P(i) \\ P(j) \end{matrix}$$

- Solve LP for $x = [xi,xj]^T$: max $\mathbf{1}^T x$ s.t. $\Delta x \le \mathbf{1}$
  - Result: $x = [1,1]$, $\mathbf{1}^T x = 2 = s_{HBL}$
- Thm: #words_moved = $\Omega(n^2/M^{s_{HBL}-1}) = \Omega(n^2/M^1)$

  Attained by block sizes $M^{xi},M^{xj} = M^1,M^1$

N-Body Speedups on IBM-BG/P (Intrepid) 8K cores, 32K particles

K. Yelick, E. Georganas, M. Driscoll, P. Koanantakool, E. Solomonik



New Thm applied to Random Code

- for i1=1:n, for i2=1:n, … , for i6=1:n
  A1(i1,i3,i6) += func1(A2(i1,i2,i4),A3(i2,i3,i5),A4(i3,i4,i6))
  A5(i2,i6) += func2(A6(i1,i4,i5),A3(i3,i4,i6))
- Record array indices in matrix Δ

$\Delta =$

i1	i2	i3	i4	i5	i6	
1	0	1	0	0	1	A1
1	1	0	1	0	0	A2
0	1	1	0	1	0	A3
0	0	1	1	0	1	A3,A4
0	0	1	1	0	1	A5
1	0	0	1	1	0	A6

- Solve LP for $x = [x1,\ldots,x7]^T$: max $\mathbf{1}^T x$ s.t. $\Delta x \le \mathbf{1}$
  - Result: $x = [2/7,3/7,1/7,2/7,3/7,4/7]$, $\mathbf{1}^T x = 15/7 = s_{HBL}$
- Thm: #words_moved = $\Omega(n^6/M^{s_{HBL}-1}) = \Omega(n^6/M^{8/7})$
  Attained by block sizes $M^{2/7},M^{3/7},M^{1/7},M^{2/7},M^{3/7},M^{4/7}$

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)

- 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 group homomorphisms, eg

  ϕ_C (i1,i2,…,ik) = (i1+2*i3-i7)

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

- Can we bound #loop_iterations (= |S|)

  given bounds on #points in its images, i.e. bounds on $|\phi_C(S)|$, $|\phi_A(S)|$, … ?

General Communication Bound

- Given S subset of Z^k, group homomorphisms ϕ_1, ϕ_2, …, bound $|S|$ in terms of $|\phi_1(S)|$, $|\phi_2(S)|$, … , $|\phi_m(S)|$
- Def: Hölder-Brascamp-Lieb LP (HBL-LP) for $s_1,\ldots,s_m$:

  for all subgroups $H < Z^k$, $\quad \text{rank}(H) \le \Sigma_j\, s_j * \text{rank}(\phi_j(H))$
- Thm (Christ/Tao/Carbery/Bennett): Given $s_1,\ldots,s_m$

$$|S| \le \Pi_j\, |\phi_j(S)|^{s_j}$$

- Thm: Given a program with array refs given by ϕ_j, choose s_j to minimize $s_{HBL} = \Sigma_j\, s_j$ subject to HBL-LP. Then

$$\#\text{words\_moved} = \Omega\,(\#\text{iterations}/M^{s_{HBL}-1})$$

Is this bound attainable (1/2)?

- But first: Can we write it down?
- Thm: (bad news) HBL-LP reduces to Hilbert's 10^{th} problem over Q (conjectured to be undecidable)
- Thm: (good news) Another LP with same solution is decidable (but expensive, so far)
- Thm: (better news) Easy to write down LP explicitly in many cases of interest (eg all ϕ_j = {subset of indices})
- Thm: (good news) Easy to approximate, i.e. get upper or lower bounds on s_{HBL}

Is this bound attainable (2/2)?

- Depends on loop dependencies
- Best case: none, or reductions (matmul)
- Thm: When all ϕ_j = {subset of indices}, dual of HBL-LP gives optimal tile sizes:

  HBL-LP: minimize $1^T * s$ s.t. $s^T * \Delta \geq 1^T$

  Dual-HBL-LP: maximize $1^T * x$ s.t. $\Delta * x \leq 1$

  Then for sequential algorithm, tile i_j by M^{x_j}
- Ex: Matmul: $s = [\ 1/2\ ,\ 1/2\ ,\ 1/2\ ]^T = x$
- Extends to unimodular transforms of indices

Ongoing Work

- Accelerate decision procedure for lower bounds
  - Ex: At most 3 arrays, or 4 loop nests
- Have yet to find a case where we cannot attain lower bound – can we prove this?
- Extend "perfect scaling" results for time and energy by using extra memory
  - "n.5D algorithms"
- Incorporate into compilers

For more details

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

Collaborators and Supporters

- **James Demmel**, **Kathy Yelick**, Michael Anderson, Grey Ballard, Erin Carson, Aditya Devarakonda, Michael Driscoll, David Eliahu, Andrew Gearhart, Evangelos Georganas, Nicholas Knight, Penporn Koanantakool, Ben Lipshitz, Oded Schwartz, Edgar Solomonik, Omer Spillinger
- Austin Benson, Maryam Dehnavi, Mark Hoemmen, Shoaib Kamil, Marghoob Mohiyuddin
- Abhinav Bhatele, Aydin Buluc, Michael Christ, Ioana Dumitriu, Armando Fox, David Gleich, Ming Gu, Jeff Hammond, Mike Heroux, Olga Holtz, Kurt Keutzer, Julien Langou, Devin Matthews, Tom Scanlon, Michelle Strout, Sam Williams, Hua Xiang
- Jack Dongarra, Dulceneia Becker, Ichitaro Yamazaki
- Sivan Toledo, Alex Druinsky, Inon Peled
- Laura Grigori**,** Sebastien Cayrols, Simplice Donfack, Mathias Jacquelin, Amal Khabou, Sophie Moufawad, Mikolaj Szydlarski
- Members of ParLab, ASPIRE, BEBOP, 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, …
algorithms and software
(and compilers)

Don't Communic…

With one exception:

Happy Birthday Michael!