Communication Lower Bounds for Programs that Access Arrays

Nicholas Knight,
Michael Christ, James Demmel, Thomas Scanlon, Katherine Yelick
UC-Berkeley

DEGAS Retreat
June 3, 2013

We acknowledge funding from Microsoft (award #024263) and Intel (award #024894), and matching funding by UC Discovery (award #DIG07-10227), with additional support from ParLab affiliates National Instruments, Nokia, NVIDIA, Oracle, and Samsung, and support from MathWorks. We also acknowledge the support of the US DOE (grants DE-SC0003959, DE-SC0004938, DE-SC0005136, DE-SC0008700, DE-AC02-05CH11231, DE-FC02-06ER25753, and DE-FC02-07ER25799), DARPA (award #HR0011-12-2-0016), and NSF (grant DMS-0901569).

Communication is expensive!

Communication means *moving data*

Serial communication = moving data across memory hierarchy

Parallel communication = moving data across network

- Communication usually dominates runtime, energy cost
  $\Rightarrow$ Avoid communication to save time and energy!
- How much can you avoid? *Lower bound* on data movement
- Attain lower bound $\Rightarrow$ Communication-optimal algorithm

Outline

1. Avoiding Communication in Linear Algebra
   - Lower bounds for matrix multiplication...
   - ...attainable by tiling
   - Lower bounds for linear algebra (...attainable?)

2. Beyond Linear Algebra: Affine Array References
   - E.g., $A(i + 2j, 3k + 4)$
   - Extends previous lower bounds to larger class of programs.
   - Lower bounds are computable.
   - Matching upper bounds (i.e., optimal algorithms) in special case: linear algebra, tensor contraction, direct N–body, database join, etc. (when array references pick a subset of the loop indices)
   - Ongoing work addresses attainability in the general case.

First Lower Bound: Matrix Multiplication ("Matmul")

A, B, C are N–by–N matrices.

$$C := C + A \cdot B \qquad \Rightarrow \qquad \begin{array}{l} \textbf{for } i = 1 : N, \\ \quad \textbf{for } j = 1 : N, \\ \quad\quad \textbf{for } k = 1 : N, \\ \quad\quad\quad C(i,j) \mathrel{+}= A(i,k) * B(k,j) \end{array}$$

Theorem ([HK81])

Consider computing $C + A \cdot B$ as above (in serial), with any order on the N^3 iterations. A processor must move

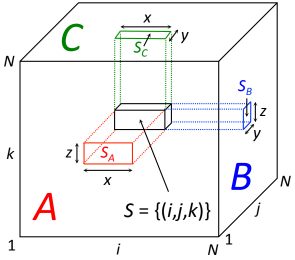
$$\#\text{ Words Moved} = \Omega\left(\frac{\#\text{iterations}}{(\text{fast memory size})^{1/2}}\right) = \boxed{\Omega\left(\frac{N^3}{M^{1/2}}\right)}$$

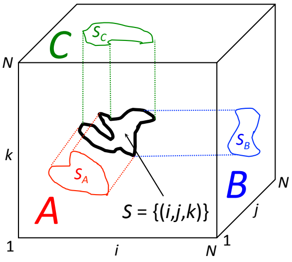
words between slow memory (of unbounded capacity) and fast memory (of size M words).

First Lower Bound: Geometric Intuition [ITT04] (1/2)

for $i = 1 : N$, **for** $j = 1 : N$, **for** $k = 1 : N$
$C(i,j) \mathrel{+}= A(i,k) * B(k,j)$

Idea Bound volume(S) by the areas of the shadows S casts





$$|S| = x \cdot y \cdot z = (xz \cdot zy \cdot yx)^{1/2}$$
$$= |S_A|^{1/2} \cdot |S_B|^{1/2} \cdot |S_C|^{1/2}$$

$$|S| \le |S_A|^{1/2} \cdot |S_B|^{1/2} \cdot |S_C|^{1/2}$$

by Loomis-Whitney ineq. [LW49]

First Lower Bound: Geometric Intuition [ITT04] (2/2)

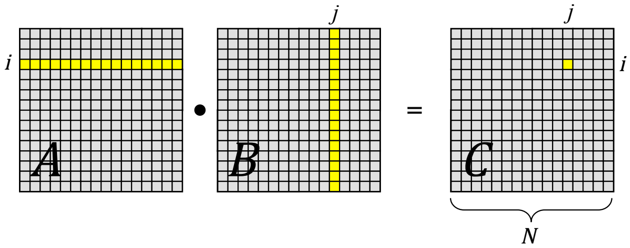
Idea Bound volume(S) by the areas of the shadows S casts

Idea Upper bound on data reuse $\Rightarrow$ lower bound on data movement

- Upper bound on number of operands: M
  - $\max(|S_A|, |S_B|, |S_C|) \leq M$
  - $|S_A| + |S_B| + |S_C| \leq M$
- Upper bound on number of iterations doable given M operands:
  - $|S| \leq |S_A|^{1/2}|S_B|^{1/2}|S_C|^{1/2} \leq M^{3/2}$
- Data reuse $=$ # iterations$/$# operands $= O(M^{1/2})$
  - (See [BDHS11] for precise argument.)
- # words moved $\geq$ total # iterations$/$max data reuse $= \Omega(N^3/M^{1/2})$

Attaining Lower Bounds — Tiling Matmul (1/3)

for $i = 1 : N$,
    for $j = 1 : N$,
        for $k = 1 : N$,
            $C(i, j) \mathrel{+}= A(i, k) * B(k, j)$



Attaining Lower Bounds — Tiling Matmul (1/3)

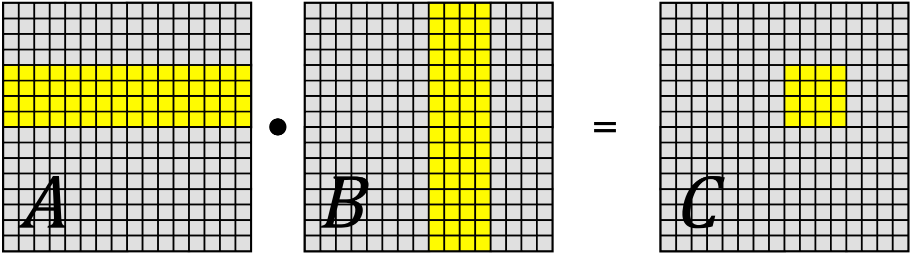
Suppose $M < N$.

for $i = 1 : N$,
  for $j = 1 : N$,
    Load $C(i,j)$ $\ldots N^2$ loads total
    for $k = 1 : N$,
      Load $A(i,k)$ and $B(k,j)$ $\ldots 2N^3$ loads total
      $C(i,j)$ += $A(i,k) * B(k,j)$
    Store $C(i,j)$ $\ldots N^2$ stores total

$$\text{\# Words Moved} = N^2 + 2N^3 + N^2 = O(N^3),$$

which is suboptimal.

Attaining Lower Bounds — Tiling Matmul (2/3)



Attaining Lower Bounds — Tiling Matmul (3/3)

Suppose $M < N$ and block/tile size $b|N$ and $3b^2 \leq M$

for $i = 1 : N/b$,
    for $j = 1 : N/b$,
        Load block $C(i, j)$ ... $(N/b)^2$ block loads total
        for $k = 1 : N/b$,
            Load blocks $A(i, k)$ and $B(k, j)$ ... $2(N/b)^3$ block loads total
            $C(i, j)$ += $A(i, k) * B(k, j)$
        Store block $C(i, j)$ ... $(N/b)^2$ block stores total

and for the choice $b = (M/3)^{1/2}$ (assume integer),

$$\#\text{ Words Moved} = \left(\frac{N^2}{b^2} + 2\frac{N^3}{b^3} + \frac{N^2}{b^2}\right) \cdot b^2 = O\left(\frac{N^3}{M^{1/2}}\right),$$

which is asymptotically optimal.

Lower Bounds for Linear Algebra

Theorem ([BDHS11])

Suppose we are given an index set $\mathcal{Z} \subset \mathbb{Z}^3$. Then the "Matmul-like" program

$$\textbf{for } (i,j,k) \in \mathcal{Z}, \quad C(i,j) = C(i,j) +_{ij} A(i,k) *_{ijk} B(k,j)$$

must move $\boxed{\Omega(|\mathcal{Z}|/M^{1/2})}$ words.

Under some technical assumptions, this yields lower bounds for

- BLAS–3, e.g., $A \cdot B$, $A^{-1} \cdot B$;
- One-sided factorizations, e.g., LU, Cholesky, LDL^T, ILU(t);
- Orthogonal factorizations, e.g., Gram–Schmidt, QR, eigenvalue/singular value problems;
- Tensor contractions, some graph algorithms; and
- Sequences of these operations, interleaved arbitrarily.

Outline

1. Avoiding Communication in Linear Algebra
   - Lower bounds for matrix multiplication...
   - ...attainable by tiling
   - Lower bounds for linear algebra (...attainable?)

2. Beyond Linear Algebra: Affine Array References
   - E.g., $A(i+2j, 3k+4)$
   - Extends previous lower bounds to larger class of programs.
   - Lower bounds are computable.
   - Matching upper bounds (i.e., optimal algorithms) in special case: linear algebra, tensor contraction, direct N–body, database join, etc. (when array references pick a subset of the loop indices)
   - Ongoing work addresses attainability in the general case.

Generalization: Affine Array References

Given 'loop iterations' indexed $(i_1, \ldots, i_d) \in \mathbb{Z}^d$, and parameters

Param.	Description	Example: Matmul
d	dim. of iteration space (*rank* of $\mathbb{Z}^d$)	3
$\mathcal{Z}$	iteration space, a subset of $\mathbb{Z}^d$	$\{1, \ldots, N\}^3$
A_j, d_j	Each d_j–dimensional array A_j is subscripted by $\mathbb{Z}^{d_j}$	$\{A, B, C\}, \{2, 2, 2\}$
ϕ_j	subscripts $\phi_j \colon \mathbb{Z}^d \to \mathbb{Z}^{d_j}$ (affine combinations of loop indices)	$\{(i,k), (k,j), (i,j)\}$
m	number of 'arrays' (injections into memory locations)	3

$$\textbf{for } i = (i_1, \ldots, i_d) \in \mathcal{Z} \subseteq \mathbb{Z}^d,$$
$$\text{inner\_loop}_i(A_1(\phi_1(i)), \ldots, A_m(\phi_m(i)))$$

Lower Bound Strategy for Affine Array References

Proof strategy:

1. For any (finite) set $E \subseteq \mathcal{Z}$ of loop iterations that accesses $O(M)$ operands, find σ such that $|E| = O(M^{\sigma})$.
   - Matmul: $\sigma = 3/2$.
2. Since data reuse $= O(M^{\sigma-1})$, we conclude the program must move $\Omega(|\mathcal{Z}|/M^{\sigma-1})$ words.
   - Matmul: $\Omega(N^3/M^{1/2})$ words.

Upper Bounds via Hölder-Brascamp-Lieb (HBL) theory

Theorem (Extension of [BCCT10, Theorem 2.4])

For $j \in \{1, \ldots, m\}$, let $\phi_j \colon \mathbb{Z}^d \to \mathbb{Z}^{d_j}$ be a group homomorphism and s_j be a nonnegative number. Then,

$$\text{for all subgroups } H \text{ of } \mathbb{Z}^d, \qquad \operatorname{rank}(H) \leq \sum_{j=1}^{m} s_j \cdot \operatorname{rank}(\phi_j(H)),$$

if and only if,

$$\text{for all finite subsets } E \text{ of } \mathbb{Z}^d, \qquad |E| \leq \prod_{j=1}^{m} |\phi_j(S)|^{s_j}.$$

Lower Bounds for Affine Array References

Theorem (Communication Lower Bound)

A program of the form

$$\textbf{for } i = (i_1, \ldots, i_d) \in \mathcal{Z}, \quad \text{inner\_loop}_i(A_1(\phi_1(i)), \ldots, A_m(\phi_m(i)))$$

must move $\boxed{\Omega(|\mathcal{Z}|/M^{(\sum_j s_j)-1})}$ words, where $s = (s_1, \ldots, s_m)$ satisfies

$$\text{rank}(H) \leq \sum_j s_j \cdot \text{rank}(\phi_j(H)) \text{ for all subgroups } H \text{ of } \mathbb{Z}^d.$$

Proof sketch:

Let E be any 'cache block'; bound data reuse (#iterations/#operands)

1. Operands: $\max_j |A_j(\phi_j(E))| = \max_j |\phi_j(E)| \leq M$
2. Iterations: $|E| \leq \prod_j |\phi_j(E)|^{s_j} \leq (\max_j |\phi_j(E)|)^{\sum_j s_j}$
3. Data reuse: $O(M^{(\sum_j s_j)-1}) \Rightarrow$ # words moved: $\Omega(|\mathcal{Z}|/M^{(\sum_j s_j)-1})$.
   - (See [CDK+13] for precise argument.)

A Linear Program to Compute σ (1/2)

We can write the set of inequalities (for subgroups $H_1, \ldots, H_i, \ldots$ of $\mathbb{Z}^d$)

$$\left\{\begin{array}{lcl} \mathrm{rank}(H_1) & \leq & \sum_{j=1}^{m} s_j \cdot \mathrm{rank}(\phi_j(H_1)) \\ & \vdots & \\ \mathrm{rank}(H_i) & \leq & \sum_{j=1}^{m} s_j \cdot \mathrm{rank}(\phi_j(H_i)) \\ & \vdots & \end{array}\right\}$$

as a system of inequalities

$$\begin{pmatrix} \mathrm{rank}(\phi_1(H_1)) & \cdots & \mathrm{rank}(\phi_m(H_1)) \\ \vdots & & \vdots \\ \mathrm{rank}(\phi_1(H_i)) & \cdots & \mathrm{rank}(\phi_m(H_i)) \\ \vdots & & \vdots \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ \vdots \\ s_m \end{pmatrix} \geq \begin{pmatrix} \mathrm{rank}(H_1) \\ \vdots \\ \mathrm{rank}(H_i) \\ \vdots \end{pmatrix},$$

or more succinctly, as $\boxed{\Delta \cdot s \geq r}$.

A Linear Program to Compute σ (2/2)

Observation

- Any $s \in [0, \infty)^m$ that satisfies $\Delta \cdot s \geq r$ leads to a valid upper bound M^σ, with $\sigma = \sigma(s) = \sum_j s_j = \mathbf{1}^T s$.
- Let s_{HBL} denote the smallest $\sigma(s)$, which leads to the tightest upper bound $M^{\sigma(s)}$, thus the tightest lower bound $\Omega(|\mathcal{Z}|/M^{\sigma(s)-1})$.

Definition (HBL–LP)

$$\text{minimize} \quad \sigma(s) = \mathbf{1}^T s \quad \text{s.t.} \quad \Delta \cdot s \geq r$$

Theorem

We can decidably compute s_{HBL}, the minimizing $\sigma(s)$.

Special Case: Subscripts are Subsets of Indices

$$\text{e.g., } A(i,k), B(k,j), C(i,j), \text{ subsets of } \{i,j,k\}$$

Theorem

Suppose every ϕ_j projects a subset of the loop indices $i_1, \ldots, i_d$. Let Δ_e be the d rows of Δ corresponding to subgroups $H_i = \langle e_i \rangle$ for $i = 1$ to d. then the linear program

$$\text{minimize} \quad \mathbf{1}^T s \quad \text{s.t.} \quad \Delta_e \cdot s \geq \mathbf{1}$$

yields the same optimum $\sigma(s)$ as HBL–LP, and furthermore, the *dual* linear program

$$\text{maximize} \quad \mathbf{1}^T x \quad \text{s.t.} \quad \Delta_e^T \cdot x \leq \mathbf{1}$$

gives the optimal block size M^{x_i} for each loop.

Note: In practice, we only need to solve the dual: $\mathbf{1}^T x = \mathbf{1}^T s = s_{\text{HBL}}$.

Special Case: Example 1/3: Matmul

Original code:

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

Now we write down and solve the linear program

$$\text{maximize} \quad s_{\text{HBL}} = \mathbf{1}^T x \quad \text{s.t.} \quad \Delta_e^T \cdot x \leq \mathbf{1}$$

$$\Delta_e = \begin{array}{cc} & \begin{array}{ccc} i & j & k \end{array} \\ \begin{array}{c} A \\ B \\ C \end{array} & \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array} \quad \Rightarrow \quad x = \begin{pmatrix} 1/2 \\ 1/2 \\ 1/2 \end{pmatrix} \quad \Rightarrow \quad s_{\text{HBL}} = 3/2$$

Special Case: Example 1/3: Matmul

Corollary

For any execution of the code, the number of words moved is $\Omega(|\mathcal{Z}|/M^{s_{\mathrm{HBL}}-1}) = \Omega(N^3/M^{1/2})$, and this is attained by blocks of size $M^{1/2}$–by–$M^{1/2}$–by–$M^{1/2}$ in the following code ($b = M^{1/2}$).

for $i_1 = 1 : b : N$, **for** $j_1 = 1 : b : N$, **for** $k_1 = 1 : b : N$,
  for $i_2 = 0 : b - 1$, **for** $j_2 = 0 : b - 1$, **for** $k_2 = 0 : b - 1$,
    $(i, j, k) = (i_1, j_1, k_1) + (i_2, j_2, k_2)$
    $\ldots$ // inner loop with index (i, j, k)

The block sizes may have to be smaller by a constant factor, e.g. $b_i = (M/m)^{x_i} = (M/3)^{1/2}$, to fit in cache simultaneously.

Special Case: Example 2/3: N–body

Original code:

$$\textbf{for } i = 1 : N, \quad \textbf{for } j = 1 : N,$$
$$F(i) \mathrel{+}= \text{compute\_force}(P(i), P(j))$$

Now we write down and solve the linear program

$$\text{maximize} \quad s_{\text{HBL}} = \mathbf{1}^T x \quad \text{s.t.} \quad \Delta_e^T \cdot x \le \mathbf{1}$$

$$\Delta_e = \begin{array}{c} \\ F \\ P_1 \\ P_2 \end{array} \begin{array}{c} \begin{array}{cc} i & j \end{array} \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array} \quad \Rightarrow \quad x = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \Rightarrow \quad s_{\text{HBL}} = 2$$

Special Case: Example 2/3: N–body

Corollary

For any execution of the code, the number of words moved is $\Omega(|\mathcal{Z}|/M^{s_{\text{HBL}}-1}) = \Omega(N^2/M)$, and this is attained by blocks of size M–by–M in the following code ($b = M$).

for $i_1 = 1 : b : N$, **for** $j_1 = 1 : b : N$,
  for $i_2 = 0 : b - 1$, **for** $j_2 = 0 : b - 1$,
    $(i, j) = (i_1, j_1) + (i_2, j_2)$
    $\ldots$ // inner loop with index (i, j)

The block sizes may have to be smaller by a constant factor, e.g. $b_i = (M/m)^{x_i} = M/3$, to fit in cache simultaneously.

Special Case: Example 3/3: Complicated Code

Original code:

$$\textbf{for } i_1 = 1:N, \quad \textbf{for } i_2 = 1:N, \quad \ldots, \quad \textbf{for } i_6 = 1:N,$$
$$A_1(i_1, i_3, i_6) \mathrel{+}= \mathrm{func}_1(A_2(i_1, i_2, i_4), A_3(i_2, i_3, i_5), A_4(i_3, i_4, i_6))$$
$$A_5(i_2, i_6) \mathrel{+}= \mathrm{func}_2(A_6(i_1, i_4, i_5), A_3(i_3, i_4, i_6))$$

Now we write down and solve the linear program

$$\text{maximize} \quad s_{\mathrm{HBL}} = \mathbf{1}^T x \quad \text{s.t.} \quad \Delta_e^T \cdot x \le \mathbf{1}$$

$$\Delta_e = \begin{array}{l} \\ A_1 \\ A_2 \\ A_{3,1} \\ A_4, A_{3,2} \\ A_5 \\ A_6 \end{array} \begin{array}{c} \begin{array}{cccccc} i_1 & i_2 & i_3 & i_4 & i_5 & i_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array} \Rightarrow \; x = \begin{pmatrix} 2/7 \\ 3/7 \\ 1/7 \\ 2/7 \\ 3/7 \\ 4/7 \end{pmatrix}$$

$$\Rightarrow \; s_{\mathrm{HBL}} = 15/7$$

Special Case: Example 3/3: Complicated Code

Corollary

For any execution of the code, the number of words moved is $\Omega(|\mathcal{Z}|/M^{s_{\text{HBL}}-1}) = \Omega(N^6/M^{8/7})$, and this is attained by blocks of size $M^{2/7}$–by–$M^{3/7}$–by–$M^{1/7}$–by–$M^{2/7}$–by–$M^{3/7}$–by–$M^{4/7}$ in the following code ($b_i = M^{x_i}$).

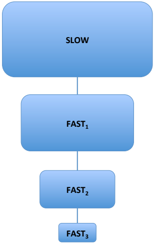
for $i_{1,1} = 1 : b_1 : N, \quad \ldots, \quad$ **for** $i_{1,6} = 1 : b_6 : N,$
  for $i_{2,1} = 0 : b_1 - 1, \quad \ldots, \quad$ **for** $i_{2,6} = 0 : b_6 - 1,$
    $(i_1, \ldots, i_6) = (i_{1,1}, \ldots, i_{1,6}) + (i_{2,1}, \ldots, i_{2,6})$
    $\ldots$ // inner loop with index $(i_1, \ldots, i_6)$

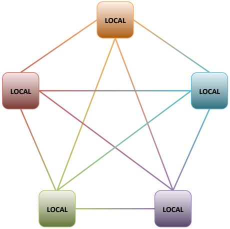
The block sizes may have to be smaller by a constant factor, e.g. $b_i = (M/m)^{x_i} = (M/6)^{x_i}$, to fit in cache simultaneously.

Extending to Other Machine Models

Our lower bounds extend to more complicated machines:

- Multiple levels of memory: apply lower bound to each pair of adjacent levels
- Homogeneous parallel processors: $|\mathcal{Z}|/P$ work per processor
- Hierarchical parallel processors
- Heterogeneous machines: optimization problem to balance $|\mathcal{Z}|$ work





Optimal Parallel Algorithms (1/2)

- Sequential tiling suggests parallel 'working sets.'
- Optimal parallel algorithms for 'special case' and $\mathcal{Z} = N^d$:

Partition domain into tiles of size N/M^{x_1}–by–$\cdots$–by–N/M^{x_d}
While there are unexecuted tiles
    Assign unexecuted tiles to P processors
    Communicate the data to each processor
    Execute tiles

$$\overbrace{\left(\prod_{j=1}^{m} \frac{N^d}{M^{x_j}}\right) \cdot \frac{1}{P}}^{\text{tiles per processor}} \cdot \overbrace{O(M)}^{\substack{\text{words moved}\\ \text{per tile}}} = \overbrace{O\left(\frac{N^d}{PM^{s_{\mathrm{HBL}}-1}}\right)}^{\substack{\text{words moved}\\ \text{per processor}}},$$

attaining the lower bound of $\Omega((|\mathcal{Z}|/P)/M^{s_{\mathrm{HBL}}-1})$.

Optimal Parallel Algorithms (2/2)

How much of the machine's memory should you use?

M each processor's working set size

M_{cap} each processor's memory capacity

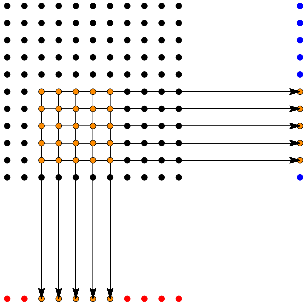
M_{arr} total storage required for arrays, $|\bigcup_j A_j(\phi_j(\mathcal{Z}))|$

$$\frac{M_{\text{arr}}}{P} \leq M \leq \min\left(M_{\text{cap}}, \left(\frac{|\mathcal{Z}|}{P}\right)^{\frac{1}{s_{\text{HBL}}}}\right)$$

- Lower bound on M: store all arrays (across machine)
- Upper bounds on M:
  - working sets must fit in processors' memories
  - load balance (need at least P tiles)
- ('N.5D algorithms') Writing $M = CM_{\text{arr}}/P$, it is beneficial to use up to $C \leq \left(\frac{|\mathcal{Z}|^{\frac{1}{s_{\text{HBL}}}}}{M_{\text{arr}}}\right) P^{\frac{s_{\text{HBL}}-1}{s_{\text{HBL}}}}$ copies of the data (Matmul: $C \leq P^{\frac{1}{3}}$)

Ongoing Work: Optimal Algorithms

- Goal: generalize duality argument beyond "special case"
- Goal: bound constants hidden in 'big-O'
  - N–body: $O(M^2)$ particle-particle interactions

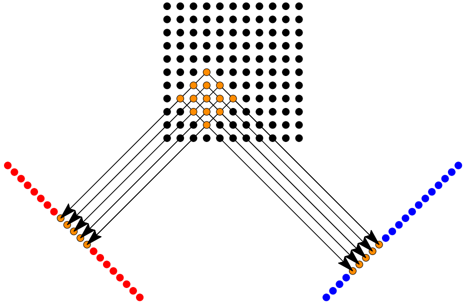


This tiling is optimal.

Access $\phi_1 = i$, $\phi_2 = j$

Ongoing Work: Optimal Algorithms

- Goal: generalize duality argument beyond "special case"
- Goal: bound constants hidden in 'big-O'
  - N–body: $O(M^2)$ particle-particle interactions

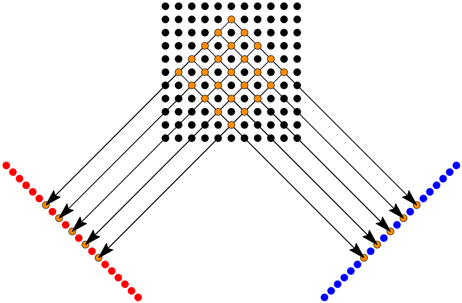


This tiling is suboptimal.

Access $\phi_1 = i - j$, $\phi_2 = i + j$

Ongoing Work: Optimal Algorithms

- Goal: generalize duality argument beyond "special case"
- Goal: bound constants hidden in 'big-O'
  - N–body: $O(M^2)$ particle-particle interactions



This tiling is optimal. Note:

- *two* sets of tiles
- generalizes to arbitrary linear combinations of i and j
- group theory reveals the optimal tiling

Access $\phi_1 = i - j$, $\phi_2 = i + j$

Ongoing Work: Data Dependencies

- Partial order on $\mathcal{Z}$ encoded as DAG
- Some sets E are inadmissible (cannot be blocked)

Question

Can we tighten our bound $|E| \leq \prod_j |\phi_j(E)|^{s_j}$ for admissible sets?

Question

Can we extend our parallel theory to expose tradeoffs between:

- Concurrency, efficiency, memory, communication, ...

Ongoing Work: Cost Model

- Only discussed communication volume (bandwidth cost).
- Extend model to address
  - # Messages/synchronizations (latency cost)

Claim

Message size $1 \le w \le M$ words, so a latency lower bound is

$$\lceil \#\text{ words moved}/w \rceil = \Omega(|\mathcal{Z}|/(wM^{s_{\text{HBL}}-1})) = \Omega(|\mathcal{Z}|/M^{s_{\text{HBL}}}) \text{ messages.}$$

  - Energy/power costs
  - Network topology, congestion

Conclusions

- Communication is slowing you down!
- Lower bounds motivate new/improved algorithms
  - Previous work: Matmul [HK81, ITT04], linear algebra [BDHS11]
  - This work: programs with affine array references
- Goal: Compiler generates communication-optimal code
- Tech. report [CDK$^+$13] at `bebop.eecs.berkeley.edu`, or

`http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-61.pdf`

Thank You

References I

J. Bennett, A. Carbery, M. Christ, and T. Tao.
Finite bounds for Hölder-Brascamp-Lieb multilinear inequalities.
Mathematical Research Letters, 17(4):647–666, 2010.

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.
Minimizing communication in numerical linear algebra.
SIAM Journal on Matrix Analysis and Applications, 32(3):866–901, 2011.

M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. Yelick.
Communication lower bounds and optimal algorithms for programs that reference arrays — part I.
Technical Report UCB/EECS-2013-61, Department of Electrical Engineering and Computer Science, University of California, Berkeley, April 2013.

J.-W. Hong and H.T. Kung.
I/O complexity: the red-blue pebble game.
In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pages 326–333. ACM, 1981.

D. Irony, S. Toledo, and A. Tiskin.
Communication lower bounds for distributed-memory matrix multiplication.
Journal of Parallel and Distributed Computing, 64(9):1017–1026, 2004.

L.H. Loomis and H. Whitney.
An inequality related to the isoperimetric inequality.
Bulletin of the American Mathematical Society, 55:961–962, 1949.