Communication Cost Models and a few Lower and Upper Bounds

Edgar Solomonik

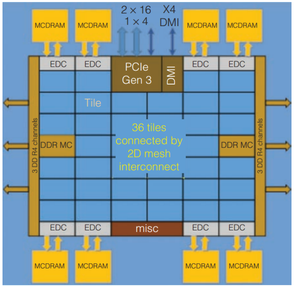
Department of Computer Science
University of Illinois at Urbana-Champaign

March 27, 2017

Communication cost as an architectural bottleneck

Architectures are increasingly constrained by communication

- floating-point units are free, amount of wiring bounds bandwidth/latency
- newest-generation memory models have non-uniform memory access times



- cloud-computing systems generally have much higher latency than supercomputers

Models for communication cost

Communication can be classified as vertical and horizontal

- vertical – data movement through the cache hierarchy (memory–cache)
- horizontal – data movement in the network (processor–processor)

Such communication costs were studied for many decades

- VLSI circuit models with bounded degree yield algorithms with bounded horizontal communication
- External memory algorithms work with bounded local memory and transfers to/from disk, effectively vertical communication

Can these communication measures differ substantively?

- Yes: consider matrix–vector multiplication
  - vertical communication is proportional to the matrix size
  - horizontal communication is proportional to the vector size
- but similar techniques (e.g. blocking) used to lower both

Models for parallelism

Circuits were the first parallel algorithms

- depth – execution time
- size – amount of work
- width – number of processors needed

The PRAM model tries to stay consistent with this view

- instead of building dataflow into hardware, simply consider a shared uniform memory
- different PRAM variants permit different concurrent memory access modes
- how to read a PRAM type:
  - E-exclusive, C-concurrent
  - R-read, W-write
- PRAM types: EREW, CREW, CRCW
- what happens on a concurrent write? more types, e.g. random or highest-priority succeeds

PRAM limitations

PRAM and circuit-style algorithms are usually designed to

1. minimize depth (execution time)
2. minimize number of processors

Each processor performs one unit of work per read/write and is always synchronized with other processors

- Brent's Lemma (coarsening)
  - consider PRAM algorithm with depth T and P processors
  - can simulate using $Q < P$ processors in time QT/P
- problem: Brent's lemma tells us little about communication/synchronization
- PRAM has no notion of local memory or cache

Parallel models with communication cost

How to incorporate communication and synchronization?

- extend PRAM to have a notion of local memory/cache or do away with the global shared memory
- communicate point-to-point n-byte messages in time

  $$\alpha + \beta \cdot n$$

  each processor receives/sends 1 message at a time
- finer details of messaging fixed by LogP and LogGP models
- Bulk Synchronous parallel (BSP) model [Valiant 1990]
  - associate synchronizations with supersteps
  - communication cost with max amount of data n_i sent or received at superstep i

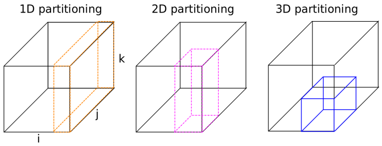
    $$T_{\text{BSP}} = \sum_{i=1}^{s} \alpha + \beta \cdot n_i$$

  - collectives (broadcast, reduce, all-to-all) can be done in 1-2 supersteps with linear bandwidth cost
  - on p processors, often shaves $O(\log p)$ in latency, sometimes (all-to-all) $O(\log p)$ in bandwidth

Matrix multiplication

Lets see how the models work for multiplication of $n \times n$ matrices

$$\boldsymbol{C}_{ij} = \sum_{l=1}^{n} \boldsymbol{A}_{il} \boldsymbol{B}_{lj}$$



- PRAM: $O(\log(n))$ depth, $O(n^3)$ work
- $T_{\text{BSP}}(n,p) = O(\alpha + \beta \cdot (n^3/p)^{2/3})$
- BSP with local memory $M \in [n^2/p, n^2/p^{2/3}]$:

$$T_{\text{BSP}}(n,p,M) = O\left(\alpha \cdot \frac{n^3}{pM^{3/2}} + \beta \cdot \frac{n^3}{p\sqrt{M}}\right)$$

Rectangular matrix multiplication

Consider rectangular matrix multiplication, $\boldsymbol{A} \in \mathbb{R}^{m\times k}$, $\boldsymbol{B} \in \mathbb{R}^{k\times n}$, $\boldsymbol{C} \in \mathbb{R}^{m\times n}$,

$$\boldsymbol{C}_{ij} = \sum_{l=1}^{k} \boldsymbol{A}_{il}\boldsymbol{B}_{lj}$$

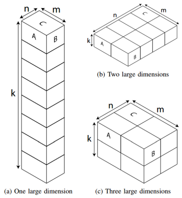


diagram source: Demmel et al 2013

- generalizes matrix-vector product, vector inner/outer products
- best algorithm given by appropriate $p_1 \times p_2 \times p_3$ processor grid
- vertical communication can be asymptotically greater than horizontal communication

Sparse matrix multiplication

A yet more general setting is sparse matrix multiplication

$$\boldsymbol{C}_{ij} = \sum_{l=1}^{n} \boldsymbol{A}_{il} \boldsymbol{B}_{lj}$$

- where $\boldsymbol{A}$ and $\boldsymbol{B}$ have $\text{nnz}(\boldsymbol{A})$ and $\text{nnz}(\boldsymbol{B})$ nonzeros
- let $\boldsymbol{C}$ have $\text{nnz}(\boldsymbol{C})$ nonzeros, two variants:
  - nonzero structure of $\boldsymbol{C}$ can be induced from $\boldsymbol{A}$ and $\boldsymbol{B}$
  - can predefine nonzero structure and compute only those entries
- important variants: SpMV, SpMSpV, SpMM, SpGEMM (SpMSpM)

Communication cost of sparse matrix multiplication

Best algorithm depends on sparsity structure

- however, randomization and partitioning based on total nonzero counts provides reasonable bounds

$$T_{\text{BSP}}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, p) = O\left(\alpha + \beta \cdot \min_{p_1 p_2 p_3 = p} \left[\frac{\text{nnz}(\boldsymbol{A})}{p_1 p_2} + \frac{\text{nnz}(\boldsymbol{B})}{p_2 p_3} + \frac{\text{nnz}(\boldsymbol{C})}{p_1 p_3}\right]\right)$$

- in fact a bit better, e.g. if $p_1 p_2 = p$ no horizontal communication proportional to nnz($\boldsymbol{A}$)
- however, always have vertical communication cost proportional to total nonzero count

Communication lower bounds for matrix multiplication

How close to optimal are these cost upper bounds?

- communication lower bounds give the minimal amount of communication for any schedule to execute an algorithm or space of algorithms
- appropriate representations of algorithms in these setting are: dependency graphs, hypergraphs, algebraic encodings (bilinear algorithms)
- for dependency graphs, we are interesting in expansion, and minimum vertex separators (cuts), for a vertex subset of a given size
- for algebraic encodings (bilinear algorithms), we are interested in the rank of the encoding of any subset of bilinear forms
- very few communication lower bounds apply to *problems*, sorting is one exception (but still only comparison-based)
- proofs don't look like hardness reductions, which are problem-to-problem

Volumetric inequalities

Inequalities that bound surface-to-volume ratio often serve as key components of communication lower bounds proofs

Theorem (Discrete Loomis-Whitney Inequality)

Consider any $V \subseteq [1, n]^d$. Then we have

$$|V| \leq \left(\prod_{j=1}^{d} |\pi_j(V)| \right)^{1/(d-1)},$$

where, for $j \in [1, d]$, $\pi_j : [1, n]^d \to [1, n]^{d-1}$ is the projection

$$\pi_j(i_1, \ldots, i_d) = (i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_d).$$

Generalizations exist to other types of projections

Volumetric inequalities

Theorem (Loomis-Whitney (3D version), 1949)

Let V be a set of 3-tuples $V \subseteq [1,n]^3$

$$|V| \le \sqrt{|\pi_1(V)||\pi_2(V)||\pi_3(V)|}$$

where

$$\begin{aligned}
\pi_1(V) &= \{(i_2, i_3) : \exists i_1, (i_1, i_2, i_3) \in V\} \\
\pi_2(V) &= \{(i_1, i_3) : \exists i_2, (i_1, i_2, i_3) \in V\} \\
\pi_3(V) &= \{(i_1, i_2) : \exists i_3, (i_1, i_2, i_3) \in V\}
\end{aligned}$$

To minimize comm. in MM, minimize $\Pi = \pi_1(V) \cup \pi_2(V) \cup \pi_3(V)$

$$|V| < |\Pi|^{3/2} \quad \Rightarrow \quad |\Pi| > |V|^{2/3}$$

when $|V| = n^3/p$, we see that $|\Pi| > (n^3/p)^{2/3}$

Lower bounds for matrix multiplication

Aforementioned dense matrix multiplication algorithms are communication-optimal

- sparse matrix multiplication is not yet fully understood
- different settings for optimality question in the sparse case
    - for a given nonzero structure, lower bound can be written as hypergraph partition, attainability is open
    - can also consider lower bounds for worst case structure, e.g. define family of graphs that are nowhere local (special case: maximum number of edges without triangles)
    - can restrict space of algorithms to be structure-oblivious

Beyond matrix multiplication

Why fuss so much about matrix multiplication?

- sparse matrix multiplication is a powerful building block
- can relax elementwise operations to different semirings (e.g. tropical)
- can often reason about communication complexity of algorithms using complexity of MM
- Kleene's algorithm (LU, QR, SVD) – tree of MMs
- Bellman Ford (sparse iterative methods) – repeated SpMV
- BFS – repeated SpMSpV
- unweighted Betweenness centrality – repeated SpMSpM
- weighted Betweenness centrality – repeated SpMM

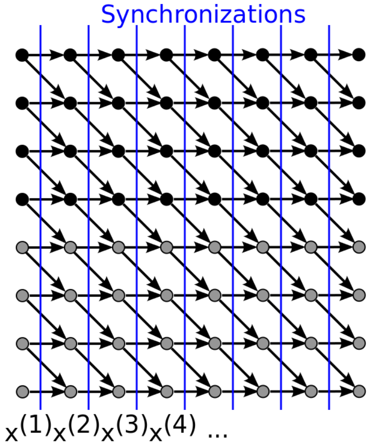
Synchronization complexity

Matrix multiplication has low synchronization cost

- but doing many small dependent MMs is a different story
- example: compare and contrast for weighted SSSP
  - Dijkstra's algorithm is inherently sequential, includes repeated SpMSpV where sparse vector has one nonzero
  - Bellman-Ford has parallelism, corresponds to SpMV (dense vector)
  - however, Bellman-Ford touches each edge only once every iteration (no data reuse), in other words vertical communication cost of SpMV is high
  - in Betweenness centrality, Brandes' algorithm can be done with many concurrent SSSPs, then we can get data reuse and good communication complexity
- but can we be smarter and parallelize/block across matrix multiplications?
- for instance, repeatedly relax all edges in an isolated neighborhood of a graph

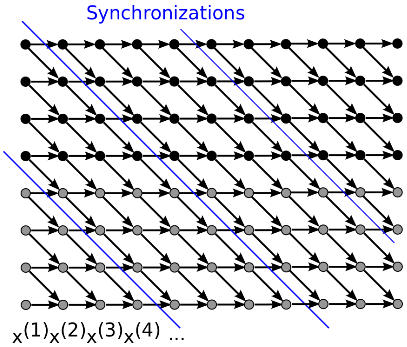
Lets start with a 1D 2-point stencil

Normally, synchronize between every stencil application



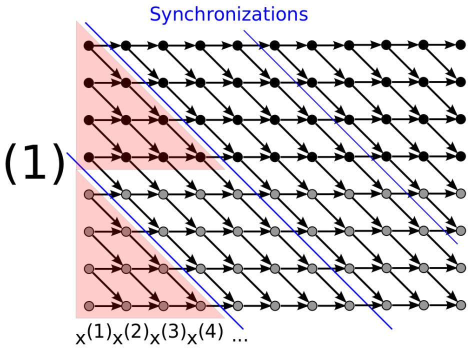
In-time blocking (matrix-powers kernel)

Avoid synchronization by applying stencil repeatedly before synchronizing



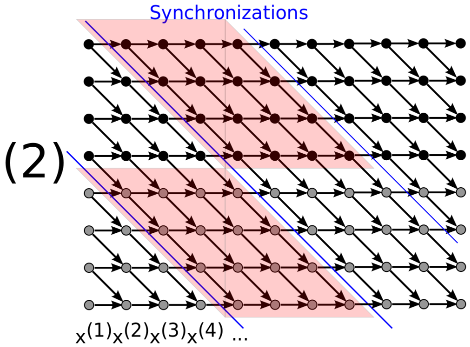
In-time blocking (matrix-powers kernel)

Avoid synchronization by applying stencil repeatedly before synchronizing



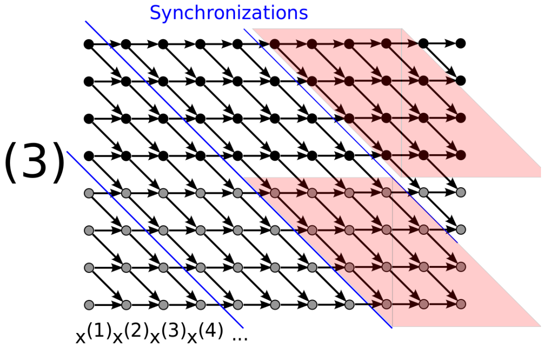
In-time blocking (matrix-powers kernel)

Avoid synchronization by applying stencil repeatedly before synchronizing



In-time blocking (matrix-powers kernel)

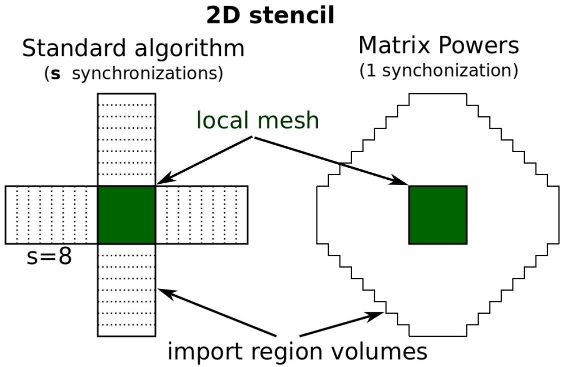
Avoid synchronization by applying stencil repeatedly before synchronizing



Analysis of in-time blocking for dD mesh

For dD mesh, there is more complexity

- again consider t steps, and execute s without synchronization
- we are constrained by $s \leq (n/p)^{1/d}$
  - otherwise we need to do asymptotically more computation and interprocessor communication

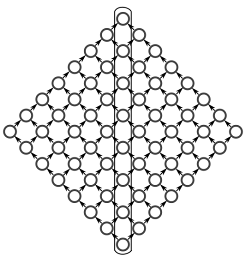


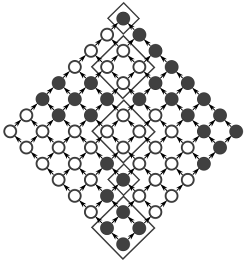
Generalizing dependency expansion

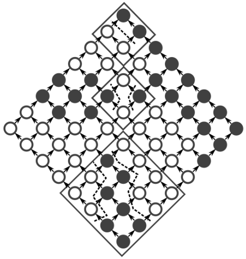
Definition ((ϵ, σ)-path-expander)

Graph $G = (V, E)$ is a (ϵ, σ)-**path-expander** if there exists a path $(u_1, \dots u_n) \subset V$, such that the dependency interval $[u_i, u_{i+b}]_G$ for each i, b has size $\Theta(\sigma(b))$ and a minimum cut of size $\Omega(\epsilon(b))$.

An example of a (b, b^2)-**path-expander**


Dependency chain P


Monochrome dependency intervals


Multicolored dependency intervals

Scheduling tradeoffs of path-expander graphs

Theorem (Path-expander communication lower bound)

Any parallel schedule of an algorithm with a (ϵ, σ)-**path-expander** *dependency graph about a path of length* n *and some* $b \in [1, n]$ *incurs computation* (F)*, communication* (W)*, and synchronization* (S) *costs:*

$$F = \Omega\left(\sigma(b) \cdot n/b\right), \quad W = \Omega\left(\epsilon(b) \cdot n/b\right), \quad S = \Omega\left(n/b\right).$$

Corollary

If $\sigma(b) = b^d$ *and* $\epsilon(b) = b^{d-1}$*, the above theorem yields,*

$$F \cdot S^{d-1} = \Omega\left(n^d\right), \;\; W \cdot S^{d-2} = \Omega\left(n^{d-1}\right).$$

Comp.(F) – comm.(W) – sync.(S) tradeoffs

Dependency interval expansion occurs in many algorithms

- for $n \times n$ Cholesky factorization

$$F_{\text{Cholesky}} \cdot S^2_{\text{Cholesky}} = \Omega(n^3)$$

$$W_{\text{Cholesky}} \cdot S_{\text{Cholesky}} = \Omega(n^2)$$

  typical algorithms for LU, QR, SVD and Kleene's APSP algorithm have similar dependency structure
- Note: APSP is cheaper via path doubling [Tiskin 2001]
  - any shortest path of length $[k/2, k]$ is composed of a shortest path of length exactly $k/2$ and a shortest path of length $\leq k/2$
  - APSP can be done using $O(\log(P))$ SpMMs with geometrically decreasing comm./comp. costs
- for computing s applications of a $(2m+1)^d$-point stencil

$$F_{\text{St}} \cdot S^d_{\text{St}} = \Omega\left(m^{2d} \cdot s^{d+1}\right), \qquad W_{\text{St}} \cdot S^{d-1}_{\text{St}} = \Omega\left(m^d \cdot s^d\right)$$

  sparse iterative methods generally look like this

Krylov subspace methods of the future

Krylov subspace methods can be used to construct a basis for the kernel of sparse matrix $\boldsymbol{A}$

$$\{\boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}, \boldsymbol{A}^2\boldsymbol{x}, \ldots, \boldsymbol{A}^k\boldsymbol{x}\}$$

- ubiquitous in scientific computing, can solve linear-systems, least-squares, and eigenvalue problems
- dominated by repeated SpMV, most sparse matrices will give high interval expansion
- randomized projection-methods replace SpMV with SpMM
  - define $n \times (k + 10)$ Gaussian random matrix $\boldsymbol{X}$
  - computation of $\boldsymbol{A}\boldsymbol{X}$ can be shown to contain a good subspace for the kernel of $\boldsymbol{A}$!
  - if high accuracy guarantees are necessary, can use

    $$(\boldsymbol{A}\boldsymbol{A}^T)^q\boldsymbol{A}\boldsymbol{X}$$

    to improve accuracy exponentially with q
    [Halko, Martinsson, Tropp 2011]

Conclusion and some references

Key points

- matrix multiplication is general if permitting sparsity
- multiplying two large matrices is comm. and sync. efficient
- multiplying a matrix by a (sparse) vector is inefficient in vertical communication and usually sync. inefficient as a building block
- more scalable algorithms often require radically different approaches (e.g. path-doubling vs Floyd-Warshall)

Key references

- Tiskin, Alexander. All-pairs shortest paths computation in the BSP model. 2001.
- Halko, N., Martinsson, P.G. and Tropp, J.A. Finding structure with randomness. SIAM review. 2011
- E. S., Erin Carson, Nicholas Knight, and James Demmel. Tradeoffs between synchronization, communication, and computation in parallel linear algebra computations. 2017.
- E. S., Maciej Besta, Flavio Vella, and Torsten Hoefler. Betweenness centrality is more parallelizable than dense matrix multiplication. 2016.
- CS598 ES, Fall 2016: Communication cost analysis of algorithms
  `http://solomon2.web.engr.illinois.edu/teaching/cs598_fall2016/index.html`

Backup slides