Communication-Optimal One-Sided Matrix Factorization

Grey Ballard

ParLab Grand Opening

December 1 2008

ballard@cs.berkeley.edu

Concepts

One-Sided Matrix Factorizations

- Basic dense linear algebra factorizations: LU, QR, Cholesky
- We now have complete set of communication–optimal algorithms for $O(n^3)$ implementations

"Communication" means

- Parallel: Data movement between processors
- Sequential: Data movement between levels of memory hierarchy
- # words (inverse bandwidth) and # messages (latency)

Proving Communication Lower Bounds

Matrix Multiplication Lower Bounds

- Results of [ITT04] and [JWK81] imply that multiplying $n \times n$ matrices with a conventional $O(n^3)$ algorithm requires communicating

$$\Omega\left(\frac{n^3}{P\sqrt{M}} - M\right) \text{ words and } \Omega\left(\frac{n^3}{PM^{3/2}}\right) \text{ messages}$$

on a P processors machine, where M is the memory size for each processor. These bounds hold for the sequential case, setting $P = 1$.

- For distributed storage, $M = \frac{n^2}{P}$, so in this case the lower bounds are

$$\Omega\left(\frac{n^2}{\sqrt{P}}\right) \text{ words and } \Omega(\sqrt{P}) \text{ messages.}$$

Reducing Matrix Multiplication to LU Factorization

- We can easily extend the matrix multiplication lower bounds to LU Factorization via the following reduction

$$\begin{pmatrix} I & 0 & -B \\ A & I & 0 \\ 0 & 0 & I \end{pmatrix} = \begin{pmatrix} I & & \\ A & I & \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & 0 & -B \\ & I & AB \\ & & I \end{pmatrix}.$$

- Given matrices A and B, we can compute the product by forming the matrix above, performing LU factorization, and extracting AB.

Reducing Matrix Multiplication to Cholesky Factorization

- A first attempt at extending the matrix multiplication lower bounds to Cholesky factorization is given by the following reduction

$$\begin{pmatrix} I & A^T & -B \\ A & I + AA^T & 0 \\ -B^T & 0 & D \end{pmatrix} = \begin{pmatrix} I & & \\ A & I & \\ -B^T & (AB)^T & X \end{pmatrix} \begin{pmatrix} I & A^T & -B \\ & I & AB \\ & & X^T \end{pmatrix}$$

- This is not a complete reduction: forming the matrix includes computing AA^T, which requires considerable communication.
- We introduce "masking" numbers 1^* and 0^* which mask computation that does not affect the matrix product AB.
- Using these masking numbers, we create the following reduction

$$\begin{pmatrix} I & A^T & -B \\ A & C & 0 \\ -B^T & 0 & C \end{pmatrix} = \begin{pmatrix} I & & \\ A & L_C & \\ -B^T & (AB)^T & L_C \end{pmatrix} \begin{pmatrix} I & A^T & -B \\ & {L_C}^T & AB \\ & & {L_C}^T \end{pmatrix} \quad \text{where } C = \begin{pmatrix} 1^* & 0^* & \cdots & 0^* \\ 0^* & 1^* & & \vdots \\ \vdots & & \ddots & 0^* \\ 0^* & \cdots & 0^* & 1^* \end{pmatrix}$$

and L_C is the lower triangular Cholesky factor of C.

- In this reduction, we no longer have to construct AA^T, but we still compute AB correctly. For details, see [BDHS09].
- Proving that the matrix multiplication lower bounds apply to QR factorization is a bit trickier, see [DGHL08].

Tall Skinny Factorizations

TSQR

- Communication bottleneck of one-sided factorizations is the column panel factorization (tall skinny submatrix)
- Existing implementations require $\Theta(b \log P)$ messages to factor panel of width b
- New TSQR algorithm requires only $\Theta(\log P)$ messages
- Reduction tree can be chosen to fit sequential, parallel, or heterogenous architectures
- TSQR is as numerically stable as Householder QR, and it meets the communication lower bounds

TSLU

- Pivoting required to maintain numerical stability in LU factorization
- Naive approach (pivoting only within blocks) is unstable for large matrices and small blocks
  - equivalent to parallel pivoting when there are as many processors as rows
- Instead, use TSLU to select best pivot rows
  - each node factors to find the best pivot rows from its children's $2b$ rows
  - best b pivot rows are promoted to parent node

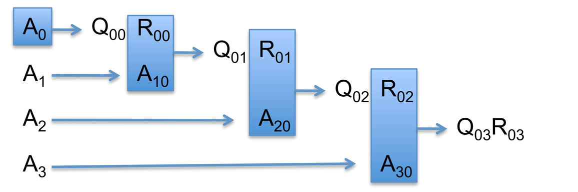

Figure 1: *Execution of the sequential TSQR factorization on a flat tree with four submatrices.*

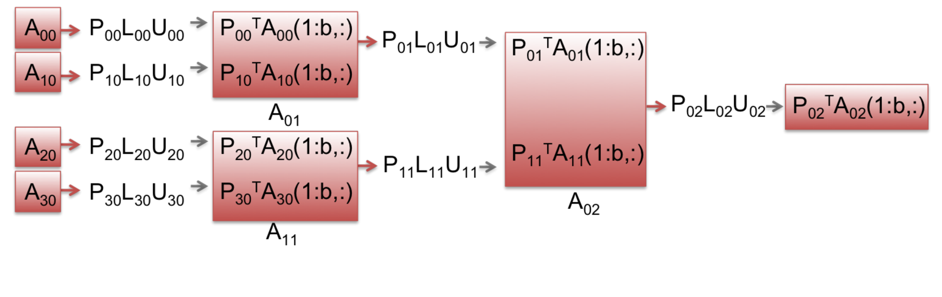

Figure 2: *Execution of the parallel TSLU pivot extraction on a binary tree of four processors.*

Communication-Avoiding LU

- CALU uses TSLU to choose best pivots first
- After applying all row permutations, factor without pivoting
- Requires extra computation (factor the panel twice)

Algorithm 1: *CALU (see [DGX08])*

> **for** $j = 1$ to n/b **do**
> find permutation for j^{th} column panel using TSLU
> ▷ broadcast pivot information across processor rows
> apply all row permutations
> compute diagonal block (j, j) factorization locally
> ▷ broadcast lower triangular factor across processor row
> ▷ broadcast upper triangular factor down processor column
> compute block column of L, block row of U
> ▷ broadcast block column of L across processor rows
> ▷ broadcast block row of U down processor columns
> update trailing matrix
> **end for**

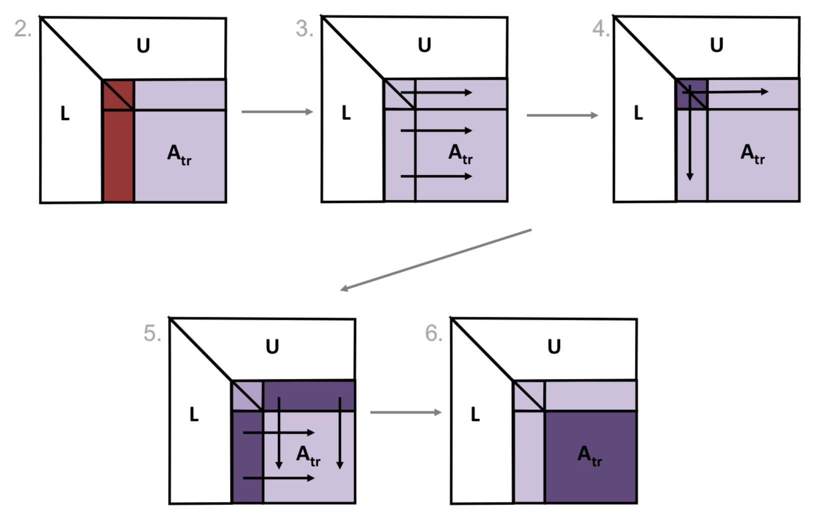

Figure 3: *Computation and communication in CALU with TSLU preprocessing highlighted in red.*

	Par. CALU	ScaLAPACK	Lower bound
# words	$O\left(\frac{n^2}{\sqrt{P}}\log P\right)$	$O\left(\frac{n^2}{\sqrt{P}}\log P\right)$	$\Omega\left(\frac{n^2}{\sqrt{P}}\right)$
# messages	$O\left(\sqrt{P}\log P\right)$	$O(n\log P)$	$\Omega\left(\sqrt{P}\right)$

Table 1: *Comparison of parallel CALU and ScaLAPACK's parallel LU factorization* `PDGETRF` *on a square $n \times n$ matrix with P processors. Maximum measured speedup: 2.3X on IBM Power5, $n = 10^3$, $P = 8 \times 8$.*

Communication-Avoiding QR

- Using TSQR to factor column panels, CAQR attains bandwidth and latency lower bounds
- CAQR is superior to existing LAPACK and ScaLAPACK routines in theory and practice
  - For performance results, see [DGHL08]

	Par. CAQR	ScaLAPACK	Lower bound
# words	$O\left(\frac{n^2}{\sqrt{P}}\log P\right)$	$O\left(\frac{n^2}{\sqrt{P}}\log P\right)$	$\Omega\left(\frac{n^2}{\sqrt{P}}\right)$
# messages	$O\left(\sqrt{P}\log^3 P\right)$	$O\left(n\log^2 P\right)$	$\Omega\left(\sqrt{P}\right)$

Table 2: *Comparison of parallel CAQR and ScaLAPACK's parallel QR factorization* `PDGEQRF` *on a square $n \times n$ matrix with P processors. Maximum predicted speedup: 9.7X on IBM Power5, $n = 10^3$, $P = 512$.*

	Seq. CAQR	Householder QR	Lower bound
# words	$O\left(\frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^4}{M} + n^2\right)$	$\Omega\left(\frac{n^3}{\sqrt{M}}\right)$
# messages	$O\left(\frac{n^3}{M^{3/2}}\right)$	$O\left(\frac{n^3}{M}\right)$	$\Omega\left(\frac{n^3}{M^{3/2}}\right)$

Table 3: *Comparison of sequential CAQR and blocked sequential Householder QR (LAPACK's* `DGEQRF`*) on a square $n \times n$ matrix with fast memory size M.*

ScaLAPACK Parallel Cholesky Algorithm

- For parallel case, existing ScaLAPACK routine meets bandwidth lower bound, and with the right block size, it meets latency lower bound
- For sequential case, existing LAPACK routines meet bandwidth lower bound but not latency lower bound
  - Block-contiguous storage required for blocked algorithms to meet latency lower bound

Algorithm 2: *ScaLAPACK routine* `PxPOTRF`

> **for** $j = 1$ to n/b **do**
> compute diagonal block (j, j) factorization locally
> ▷ broadcast result down processor column
> compute block column
> ▷ broadcast block column across processor rows
> ▷ re-broadcast down processor columns
> **end for**

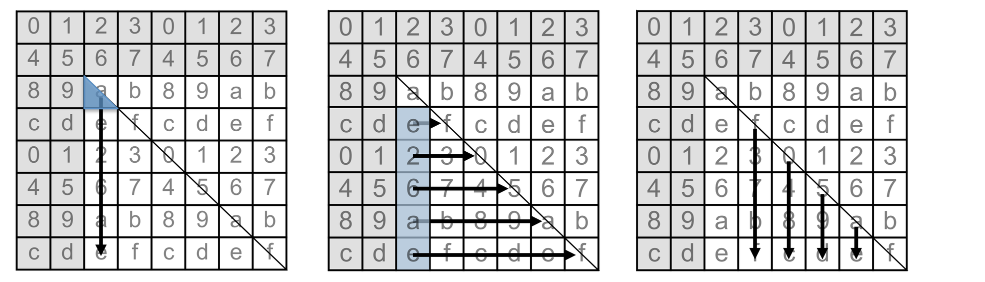

Figure 4: *Paths of communication in* `PxPTORF`.

	LAPACK	Seq. Lower bound	ScaLAPACK	Par. Lower Bound
# words	$O\left(\frac{n^3}{\sqrt{M}}\right)$	$\Omega\left(\frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^2}{\sqrt{P}}\log P\right)$	$\Omega\left(\frac{n^2}{\sqrt{P}}\right)$
# messages	$O\left(\frac{n^3}{M}\right)$	$\Omega\left(\frac{n^3}{M^{3/2}}\right)$	$O\left(\sqrt{P}\log P\right)$	$\Omega\left(\sqrt{P}\right)$

Table 4: *Comparison of existing Cholesky factorizations (LAPACK's* `DPOTRF` *and ScaLAPACK's* `PDPOTRF`*) with theoretical lower bounds.*

Future Work

- Explore lower bounds for asymptotically faster algorithms (e.g. Strassen)
- Explore lower bounds for two-sided factorizations and eigenvalue/SVD problems
- Implement sequential Cholesky algorithm with correct data structure to minimize latency
- Implement these algorithms on manycore architectures

References

[BDHS09] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Communication-optimal parallel and sequential Cholesky-decomposition. 2009.

[DGHL08] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. UC Berkeley Technical Report EECS-2008-89, Aug 1, 2008; Submitted to SIAM. J. Sci. Comp., 2008.

[DGX08] J. Demmel, L. Grigori, and H. Xiang. Communication-avoiding Gaussian elimination. Supercomputing 08, 2008.

[ITT04] D. Irony, S. Toledo, and A. Tiskin. Communication lower bounds for distributed-memory matrix multiplication. *J. Parallel Distrib. Comput.*, 64(9):1017–1026, 2004.

[JWK81] Hong Jia-Wei and H. T. Kung. I/O complexity: The red-blue pebble game. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on theory of computing*, pages 326–333, New York, NY, USA, 1981. ACM.