Lecture 2: Parallel matrix-matrix multiplication

HPC for numerical methods and data analysis

Laura Grigori

EPFL and PSI

September 26, 2023

Plan

Motivation for reducing communication

Minimizing communication in dense linear algebra

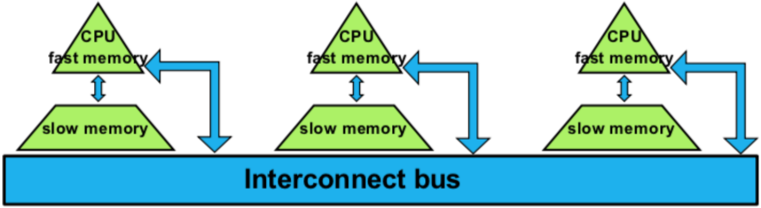
Communication lower bounds with tight constants

Rectangular matrix multiplication

Open questions and conclusions

Motivation: the communication wall

- Runtime of an algorithm is the sum of:
  - #flops x time_per_flop
  - # words_moved / bandwidth
  - # messages x latency
- Time to move data $\gg$ time per flop
  - Gap steadily and exponentially growing over time



The communication wall: compelling numbers

Time/flop 59% annual improvement up to 2004[1]

2008 Intel Nehalem 3.2GHz$\times$4 cores (51.2 GFlops/socket)	1x
2020 A64FX 2.2GHz$\times$48 cores (3.37 TFlops/socket DP)[2]	66x in 12 years

DRAM latency: 5.5% annual improvement up to 2004[1]

DDR2 (2007)	120 ns	1x
DDR4 (2014)	45 ns	2.6x in 7 years
Stacked memory	similar to DDR4	

Network latency: 15% annual improvement up to 2004[1]

Interconnect: a few μs MPI latency

[1] "Getting up to speed, The future of supercomputing" 2004, data from 1995-2004

[2] Fugaku supercomputer `https://www.top500.org/system/179807/`

Approaches for reducing communication

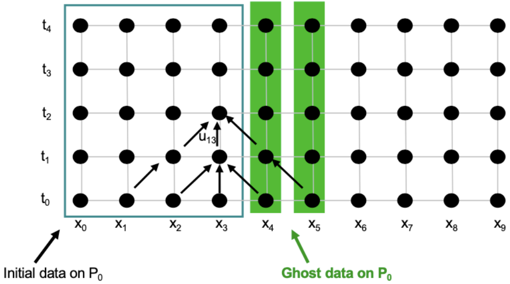
- Tuning
  - Overlap communication and computation, at most a factor of 2 speedup
- Ghosting
  - Standard approach in explicit methods
  - Store redundantly data from neighboring processors for future computations

Example of a parabolic PDE

$$u_t = \alpha \Delta u$$

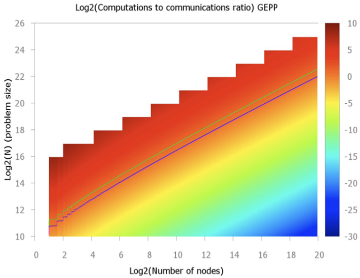
with a finite difference, the solution at a grid point is

$$\begin{aligned} u_{i,j+1} &= u(x_i, t_{j+1}) \\ &= f(u_{i-1,j}, u_{i,j}, u_{i+1,j}) \end{aligned}$$



Approaches for reducing communication (contd)

- Same numerical algorithm, different schedule of the computation
  - Block algorithms for NLA
    Barron and Swinnerton-Dyer, 1960
    - LU factorization used to solve a system with 31 equations - first subroutine written for EDSAC 2
    - Block LU factorization used to solve a system with 100 equations using an auxiliary magnetic-tape
    - The basis of the algorithm used in LAPACK, ScaLAPACK, Blackford et al 97
  - Cache oblivious algorithms: recursive Cholesky, LU, QR (Gustavson '97, Toledo '97, Elmroth and Gustavson '98, Frens and Wise '03, Ahmed and Pingali '00)



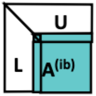
Approaches for reducing communication (contd)

- Same algebraic framework, different numerical algorithm
  - More opportunities for reducing communication, may affect stability
    Dense LU-like factorization (Barron and Swinnerton-Dyer, 60)
    - LU-like factorization based on pairwise pivoting and its block version
      $PA = L_1 L_2 \ldots L_n U$
    - With small modifications, minimizes communication between two levels of fast-slow memory
    - Stable for small matrices, unstable for nowadays matrices

Evolution of numerical libraries

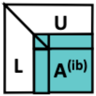
LINPACK (70's)

- vector operations, use BLAS1
- HPL benchmark based on Linpack LU factorization



LAPACK (80's)

- Block versions of the algorithms used in LINPACK
- Uses BLAS3

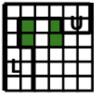


ScaLAPACK (90's)

- Targets distributed memories
- 2D block cyclic distribution of data
- PBLAS based on message passing

PLASMA (2008): new algorithms

- Targets many-core
- Block data layout
- Low granularity, high asynchronicity



Project developed by U Tennessee Knoxville, UC Berkeley, other collaborators.

Source: inspired from J. Dongarra, UTK, J. Langou, CU Denver

Plan

Communication Complexity of Dense Linear Algebra

Matrix multiply, using $2n^3$ flops (sequential or parallel)

- Hong-Kung (1981), Irony/Tishkin/Toledo (2004)
- Lower bound on Bandwidth $= \Omega(\#flops/M^{1/2})$
- Lower bound on Latency $= \Omega(\#flops/M^{3/2})$

Same lower bounds apply to LU using reduction

- Demmel, LG, Hoemmen, Langou, tech report 2008, SISC 2012

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

And to almost all direct linear algebra

[Ballard, Demmel, Holtz, Schwartz, 09]

also extended to fast linear algebra

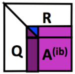
2D Parallel algorithms and communication bounds

Memory per processor $= \Theta(n^2/P)$, lower bounds on communication:

$$\#\text{words\_moved} \geq \Omega(n^2/\sqrt{P}), \quad \#\text{messages} \geq \Omega(\sqrt{P})$$

Most classical algorithms (ScaLAPACK) attain lower bounds on #words_moved

but do not attain lower bounds on #messages



	ScaLAPACK	CA algorithms
LU	partial pivoting	tournament pivoting (TP) [LG, Demmel, Xiang, 08]
QR	column based Householder	reduction based Householder [Demmel, LG, Hoemmen, Langou, 08] [Ballard, Demmel, LG, Jacquelin, Nguyen, Solomonik, 14]
RRQR	column pivoting	tournament pivoting (TP) [Demmel, LG, Gu, Xiang 13] randomized QRCP +TP [Martinsson, Voronin 15], [Duersch, Gu 15]
Eig(A)	Hessenberg/QR alg	[Ballard, Demmel, Dumitriu 10]

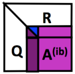
Only several references shown

2D Parallel algorithms and communication bounds

Memory per processor $= \Theta(n^2/P)$, lower bounds on communication:

$$\#\text{words\_moved} \geq \Omega(n^2/\sqrt{P}), \quad \#\text{messages} \geq \Omega(\sqrt{P})$$

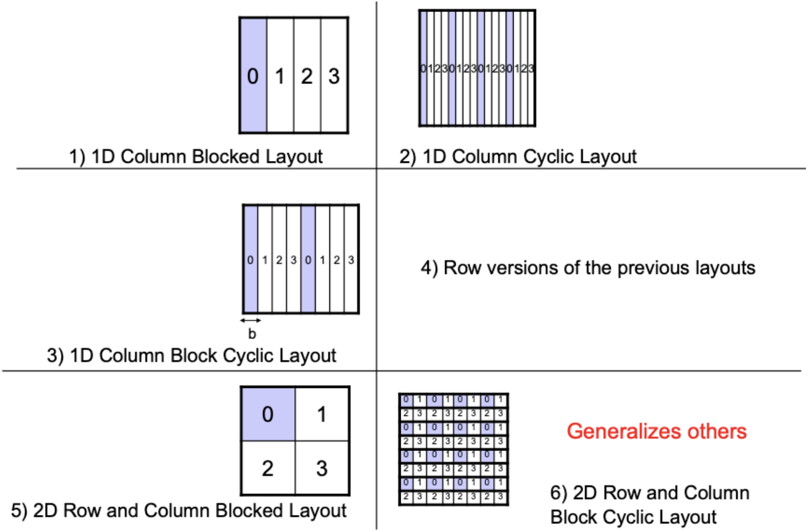
Most classical algorithms (ScaLAPACK) attain
lower bounds on #words_moved
but do not attain lower bounds on #messages



	ScaLAPACK	CA algorithms
LU	partial pivoting	tournament pivoting (TP) [LG, Demmel, Xiang, 08]
QR	column based Householder	reduction based Householder [Demmel, LG, Hoemmen, Langou, 08] [Ballard, Demmel, LG, Jacquelin, Nguyen, Solomonik, 14]
RRQR	column pivoting	tournament pivoting (TP) [Demmel, LG, Gu, Xiang 13] randomized QRCP +TP [Martinsson, Voronin 15], [Duersch, Gu 15]
Eig(A)	Hessenberg/QR alg	[Ballard, Demmel, Dumitriu 10]

Only several references shown

Matrix distributions



Parallel matrix-vector multiplication

Compute $y = y + Ax$ on P procs, where A is a dense matrix

- Consider A is distributed along its rows (1D)
- Each processor i receives a block row A_i of A, a block x_i and y_i of vectors x and y respectively

$$\begin{array}{ccc} y & = & A \quad\quad x \\ \begin{bmatrix} y_0(P_0) \\ y_1(P_1) \\ y_2(P_2) \\ y_3(P_3) \end{bmatrix} & = & \begin{bmatrix} A_0(P_0) \\ A_1(P_1) \\ A_2(P_2) \\ A_3(P_3) \end{bmatrix} \begin{bmatrix} x_0(P_0) \\ x_1(P_1) \\ x_2(P_2) \\ x_3(P_3) \end{bmatrix} \end{array}$$

Algorithm uses the formula $y(i) = A(i,:)x = \sum_j A(i,j)x(j)$:

For each processor i
  Broadcast x_i
  Compute $y_i = A_i x$

Parallel matrix-vector multiplication (contd)

Compute $y = y + Ax$ on $P = \sqrt{P} \times \sqrt{P}$ processors

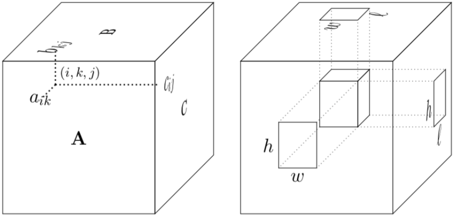
- Consider A is distributed along its rows and columns (2D)
- Each processor i receives a block of A, a block x_i and y_i of vectors x and y respectively as below

$$
\begin{array}{ccccc}
y & = & A & x \\
\begin{bmatrix} P_0 \\ P_4 \\ P_8 \\ P_{12} \end{bmatrix} & = & \begin{bmatrix} P_0 & P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 & P_7 \\ P_8 & P_9 & P_{10} & P_{11} \\ P_{12} & P_{13} & P_{14} & P_{15} \end{bmatrix} & \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}
\end{array}
$$

- Algorithm uses a broadcast and reduction, both on a subset of processors

Plan

Communication lower bounds



(a) One lattice point and its projections (b) Rectangular prism and its projections

$$\sqrt{wh \cdot w\ell \cdot h\ell} = wh\ell$$

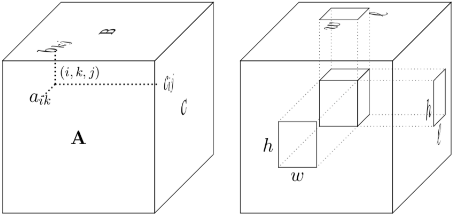
$\rightarrow$ Rectangular prism most efficient shape for maximizing volume

Lemma 1 ([Loomis and Whitney, 1949])

V a finite set of lattice points (i, j, k) in $\mathbb{R}^3$,
$\phi_i(V)$ projection of V in i-direction: points (j, k) s.t. $\exists i$ and $(i, j, k) \in V$,
ditto for $\phi_j(V), \phi_k(V)$. Then:

$$|V| \leq \sqrt{|\phi_i(V)| \cdot |\phi_j(V)| \cdot |\phi_k(V)|}$$

Lower bounds for matrix multiplication $C = AB$



(a) One lattice point and its projections (b) Rectangular prism and its projections

- $A \in \mathbb{R}^{m \times k}$, $B \in \mathbb{R}^{k \times n}$, $C \in \mathbb{R}^{m \times n}$
- Instruction stream broken into segments
- Each segment contains x loads and stores
- M fast memory size

Using Lemma 1 and AM-GM inequality, bound total scalar multiplications per segment:

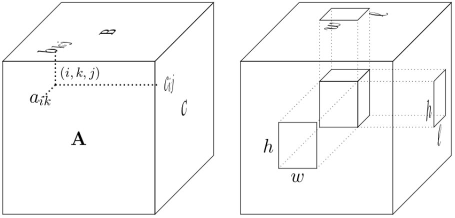
$$|V| \leq \sqrt{|\phi_i(V)| \cdot |\phi_j(V)| \cdot |\phi_k(V)|} \leq \left(\frac{|\phi_i(V)| + |\phi_j(V)| + |\phi_k(V)|}{3} \right)^{3/2} \leq \left(\frac{M + x}{3} \right)^{3/2}$$

$$\#segments \geq \left\lfloor \frac{mnk}{\left(\frac{M+x}{3}\right)^{3/2}} \right\rfloor \implies \#loads/stores \geq x \left\lfloor \frac{mnk}{\left(\frac{M+x}{3}\right)^{3/2}} \right\rfloor$$

or

$$\#loads/stores \geq 2M \left\lfloor \frac{mnk}{M^{3/2}} \right\rfloor \geq \frac{2mnk}{\sqrt{M}} - 2M$$

Lower bounds for matrix multiplication $C = AB$



(a) One lattice point and its projections (b) Rectangular prism and its projections

Lower bound for volume of communication [Smith et al., 2019]:

$$\#loads/stores \geq 2M \left\lfloor \frac{mnk}{M^{3/2}} \right\rfloor \geq \frac{2mnk}{\sqrt{M}} - 2M$$

- Bound attained by a block algorithm if $\min\{m, n, k\} \geq \sqrt{M+1} - 1$
- For square matrices, algorithm uses $b \times 1$ blocks of A, $1 \times b$ blocks of B and $b \times b$ blocks of C, with $b \leq \sqrt{M+1} - 1$

Lower bounds for parallel algorithms

Memory dependent lower bounds

Compute $C = AB$ on P processors, assume $m = n = k$:

$$W = \#words\_moved \geq \frac{2n^3}{P\sqrt{M}} - M$$

- 2D algorithms: $M = \Theta(n^2/P) \implies W = \Omega(n^2/P^{1/2})$
- 3D algorithms: $M = \Theta(n^2/P^{2/3}) \implies W = \Omega(n^2/P^{2/3})$
- 2.5D algorithms: $M = \Theta(cn^2/P) \implies W = \Omega(n^2/(cP)^{1/2})$

Lower bounds for parallel algorithms

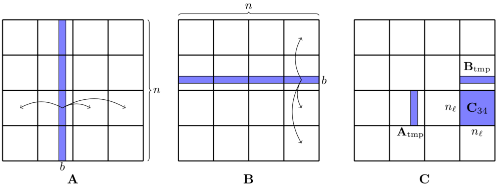
Memory dependent lower bounds

Compute $C = AB$ on P processors, assume $m = n = k$:

$$W = \#words\_moved \geq \frac{2n^3}{P\sqrt{M}} - M$$

- 2D algorithms: $M = \Theta(n^2/P) \implies W = \Omega(n^2/P^{1/2})$
- 3D algorithms: $M = \Theta(n^2/P^{2/3}) \implies W = \Omega(n^2/P^{2/3})$
- 2.5D algorithms: $M = \Theta(cn^2/P) \implies W = \Omega(n^2/(cP)^{1/2})$

SUMMA matrix multiplication



Cost of communication:

$$\beta \cdot O\left(\frac{n^2}{\sqrt{P}}\right) + \alpha \cdot O\left(\frac{n}{b} \log P\right)$$

Require: A, B, C are $n \times n$ matrices in identical 2D block distribution across processors
Require: Processors arranged in $\sqrt{P} \times \sqrt{P}$ grid where $n_\ell = n/\sqrt{P}$ is an integer
Require: Proc (I, J) owns $n_\ell \times n_\ell$ submatrix $M_{IJ} = M((I-1)n_\ell+1 : In_\ell, (J-1)n_\ell+1 : Jn_\ell)$

function $C = \mathrm{SUMMA}(C, A, B, b)$
    $(I, J) = MyProcID()$
    for $K = 1$ to $\sqrt{P}$ **do**
        for $k = 1$ to $\frac{n_\ell}{b}$ **do**
            Proc (I, K) broadcasts $A_{IK}(:, (k-1)b+1{:}kb)$ to $proc(I, :)$, store in A_{tmp}
            Proc (K, J) broadcasts $B_{KJ}((k-1)b+1{:}kb, :)$ to $proc(:, J)$, store in B_{tmp}
            $C_{IJ} = C_{IJ} + A_{\mathrm{tmp}} \cdot B_{\mathrm{tmp}}$
        end for
    end for
end function

Presentation from van de Geijn and Watts '96

3D Parallel Matrix Multiplication

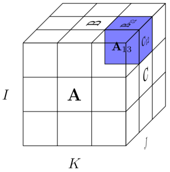


Figure: Proc(1,2,3)

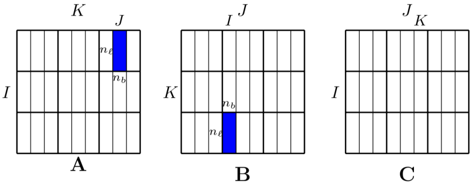


Figure: Initial distribution of A and B

Processors arranged in $\sqrt[3]{P} \times \sqrt[3]{P} \times \sqrt[3]{P}$ grid
A, B are $n \times n$ matrices in 2D block distribution across $\sqrt[3]{P} \times (\sqrt[3]{P})^2$ processor grid where $n_\ell = n/\sqrt[3]{P}$ and $n_b = n/(\sqrt[3]{P})^2$ are integers
Processor (I, J, K) owns $n_\ell \times n_b$ submatrices

$$A_{IKJ} = A_{IK}(:, (J-1)n_b+1 : Jn_b)$$

$$B_{KJI} = B_{KJ}(:, (I-1)n_b+1 : In_b)$$

$$C_{IJK} = C_{IJ}(:, (K-1)n_b+1 : Kn_b)$$

References: Dekel, Nassimi, Sahni [81], Bernsten [89], Agarwal, Chandra, Snir [90], Johnson [93], Agarwal, Balle, Gustavson, Joshi, Palkar [95]

3D Parallel Matrix Multiplication

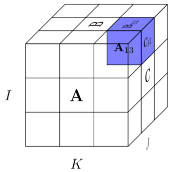


Figure: Proc(1,2,3)

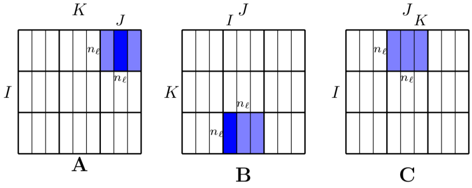


Figure: All-gathers and local computation

Assert: $C = C + AB$ is $n \times n$ matrix in 2D block distribution across processors so that processor (I, J, K) owns C_{IJK}

```
function C = 3D-MATMUL(C, A, B)
    (I, J, K) = MyProcID()
    All-gather A_{IKJ} across Proc(I, :, K), store in A_{IK}
    All-gather B_{KJI} across Proc(:, J, K), store in B_{KJ}
    \overline{C}_{IJ} = A_{IK} · B_{KJ}
    Reduce-scatter \overline{C}_{IJ} across Proc(I, J, :), combine result with C_{IJK}
end function
```

3D Parallel Matrix Multiplication

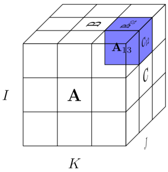


Figure: Proc(1,2,3)

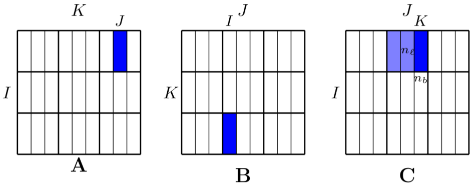


Figure: Reduce-scatter to obtain final distribution of C

Assert: $C = C + AB$ is $n \times n$ matrix in 2D block distribution across processors so that processor (I, J, K) owns C_{IJK}

function $C = \mathrm{3D\text{-}M{\scriptsize ATMUL}}(C, A, B)$
- $(I, J, K) = MyProcID()$
- All-gather A_{IKJ} across $Proc(I, :, K)$, store in A_{IK}
- All-gather B_{KJI} across $Proc(:, J, K)$, store in B_{KJ}
- $\overline{C}_{IJ} = A_{IK} \cdot B_{KJ}$
- Reduce-scatter $\overline{C}_{IJ}$ across $Proc(I, J, :)$, combine result with C_{IJK}

end function

3D Parallel Matrix Multiplication

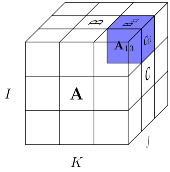


Figure: Proc(1,2,3)

Communication cost

$$\beta \cdot O\left(\frac{n^2}{P^{2/3}}\right) + \alpha \cdot O\left(\log P\right)$$

Assert: $C = C + AB$ is $n \times n$ matrix in 2D block distribution across processors so that processor (I, J, K) owns C_{IJK}

function $C = \text{3D-}\textsc{Matmul}(C, A, B)$
  $(I, J, K) = MyProcID()$
  All-gather A_{IKJ} across $Proc(I, :, K)$, store in A_{IK}
  All-gather B_{KJI} across $Proc(:, J, K)$, store in B_{KJ}
  $\overline{C}_{IJ} = A_{IK} \cdot B_{KJ}$
  Reduce-scatter $\overline{C}_{IJ}$ across $Proc(I, J, :)$, combine result with C_{IJK}
end function

Rectangular case: derivation of the bound

Consider multiplying $m \times n$ and $n \times k$ matrices (with $m \geq n \geq k$) using P processors

Key optimization problem:

$$\min x + y + z$$

such that

$$\left(\frac{mnk}{P}\right)^2 \leq xyz$$
$$\frac{nk}{P} \leq x$$
$$\frac{mk}{P} \leq y$$
$$\frac{mn}{P} \leq z$$

Analytical solution:

1. if $1 \leq P \leq \frac{m}{n}$, then

$$x^* = nk, \; y^* = mk/P, \; z^* = mn/P$$

2. if $\frac{m}{n} \leq P \leq \frac{mn}{k^2}$, then

$$x^* = y^* = (mnk^2/P)^{1/2}, \; z^* = mn/P$$

3. if $\frac{mn}{k^2} \leq P$, then

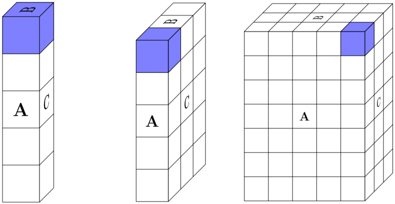
$$x^* = y^* = z^* = (mnk/P)^{2/3}$$

- Variables represent amount of data required by a processor from each matrix
- Constraints derived from properties of matrix multiplication

Rectangular Matrix Multiplication

Any classical parallel matrix multiplication algorithm multiplying $m \times n$ and $n \times k$ matrices (with $m \geq n \geq k$) using P processors must communicate at least [Daas et al., 2022]

$$\begin{cases} 1 \cdot nk - \frac{nk}{P} & \text{words if} \quad 1 \leq P \leq \frac{m}{n}, \\ 2 \cdot \left(\frac{mnk^2}{P}\right)^{1/2} - \frac{mk+nk}{P} & \text{words if} \quad \frac{m}{n} \leq P \leq \frac{mn}{k^2}, \\ 3 \cdot \left(\frac{mnk}{P}\right)^{2/3} - \frac{mn+mk+nk}{P} & \text{words if} \quad \frac{mn}{k^2} \leq P. \end{cases}$$



(a) 1D (b) 2D (c) 3D

Rectangular Matrix Multiplication

Any classical parallel matrix multiplication algorithm multiplying $m \times n$ and $n \times k$ matrices (with $m \geq n \geq k$) using P processors must communicate at least [Daas et al., 2022]

$$\begin{cases} 1 \cdot nk - \frac{nk}{P} & \text{words if} \quad 1 \leq P \leq \frac{m}{n}, \\ 2 \cdot \left(\frac{mnk^2}{P}\right)^{1/2} - \frac{mk+nk}{P} & \text{words if} \quad \frac{m}{n} \leq P \leq \frac{mn}{k^2}, \\ 3 \cdot \left(\frac{mnk}{P}\right)^{2/3} - \frac{mn+mk+nk}{P} & \text{words if} \quad \frac{mn}{k^2} \leq P. \end{cases}$$

This lower bound is tight (with tight constants) in each of the three cases, as it is attained by a grid-based algorithm with optimal 1D, 2D, or 3D processor grid

Related Work

	1D	2D	3D
Processor range	$1 \leq P \leq \frac{m}{n}$	$\frac{m}{n} \leq P \leq \frac{mn}{k^2}$	$\frac{mn}{k^2} \leq P$
Leading term	nk	$\left(\frac{mnk^2}{P}\right)^{1/2}$	$\left(\frac{mnk}{P}\right)^{2/3}$
[Aggarwal et al., 1990]	-	-	$\left(\frac{1}{2}\right)^{2/3} \approx .63$
[Irony et al., 2004]	-	-	$\frac{1}{2} = .5$
[Demmel et al., 2013]	$\frac{16}{25} = .64$	$\left(\frac{2}{3}\right)^{1/2} \approx .82$	1
[Daas et al., 2022]	1	2	3

Table: Explicit constants of communication lower bounds from related work

Plan

Open questions

Bounds with tight constants for different algorithms

- No tight constants for fast linear algebra

Symmetric operations as $C = AA^T$, $A \in \mathbb{R}^{n_1 \times n_2}$ (SYRK)

- Parallel implementations of SYRK have same communication cost as GEMM for twice less flops
- Recent results on lower bounds on communication (bandwidth):
  - Sequential SYRK a factor of $2^{3/2}$ smaller than GEMM [Beaumont et al., 2022]
  - Memory independend bounds for parallel SYRK a factor of 2 less than GEMM [Daas et al., 2023]
  - Bounds attained by using a triangular blocking [Beaumont et al., 2022]
- Constraints on projections derived from Loomis-Whitney inequality

Open questions

Tensor operations Nested loops with dependencies

Compilers perspective

- Automate generation of lower bounds and tilings
- Integrate dependencies
- Work by [Olivry et al, 20] , [Kwasniewski et al, 21], applied to PolyBench

Conclusions

- Some of the methods discussed available in libraries
  - LAPACK, ScaLAPACK, SLATE, Spark, GNU Scientific library, Cray scientific library
- Upcoming book on *Communication-Avoiding Algorithms*, with G. Ballard, E. Carson, J. Demmel

Acknowledgements

- Many figures taken from our upcoming book
- EMC^2 ERC Synergy project. This project has received funding from the European Commission under the Horizon 2020 research and innovation programme Grant agreement No. 810367.

References (1)

Aggarwal, A., Chandra, A. K., and Snir, M. (1990).
Communication complexity of PRAMs.
Theor. Comp. Sci., 71(1):3–28.

Beaumont, O., Eyraud-Dubois, L., Langou, J., and Vérité, M. (2022).
I/o-optimal algorithms for symmetric linear algebra kernels.
In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '22, pages 423–433, New York, NY, USA. ACM.

Beaupère, M. and Grigori, L. (2021).
Communication avoiding low rank approximation based on QR with tournament pivoting.
working paper or preprint.

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

Businger, P. A. and Golub, G. H. (1965).
Linear least squares solutions by Householder transformations.
Numer. Math., 7:269–276.

Christ, M., Demmel, J., Knight, N., Scanlon, T., and Yelick, K. A. (2013).
Communication lower bounds and optimal algorithms for programs that reference arrays - part 1.
Technical Report UCB/EECS-2013-61, EECS Department, University of California, Berkeley.

Daas, A. A., Ballard, G., Grigori, L., Kumar, S., and Rouse, K. (2023).
Parallel memory-independent communication bounds for syrk.
Technical report, arxiv.

References (2)

Daas, H. A., Ballard, G., Grigori, L., Kumar, S., and Rouse, K. (2022).
Tight memory-independent parallel matrix multiplication communication lower bounds.

Demmel, J., Eliahu, D., Fox, A., Kamil, S., Lipshitz, B., Schwartz, O., and Spillinger, O. (2013).
Communication-optimal parallel recursive rectangular matrix multiplication.
In *IPDPS*, pages 261–272.

Demmel, J., Grigori, L., Gu, M., and Xiang, H. (2015).
Communication-avoiding rank-revealing qr decomposition.
SIAM Journal on Matrix Analysis and its Applications, 36(1):55–89.

Demmel, J. and Rusciano, A. (2016).
Parallelepipeds obtaining hbl lower bounds.

Dinh, G. and Demmel, J. (2020).
Communication-optimal tilings for projective nested loops with arbitrary bounds.
In *SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures*.

Duersch, J. A. and Gu, M. (2017).
Randomized QR with column pivoting.
SIAM Journal on Scientific Computing, 39(4):C263–C291.

Gu, M. and Eisenstat, S. C. (1996).
Efficient algorithms for computing a strong rank-revealing QR factorization.
SIAM J. Sci. Comput., 17(4):848–869.

Irony, D., Toledo, S., and Tiskin, A. (2004).
Communication lower bounds for distributed-memory matrix multiplication.
J. Par. and Dist. Comp., 64(9):1017–1026.

References (3)

Loomis, L. H. and Whitney, H. (1949).
An inequality related to the isoperimetric inequality.
Bulletin of the American Mathematical Society, 55(10):961 – 962.

Martinsson, P.-G. and Voronin, S. (2016).
A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices.
SIAM Journal on Scientific Computing, 38(5):S485–S507.

Smith, T. M., Lowery, B., Langou, J., and van de Geijn, R. A. (2019).
A tight i/o lower bound for matrix multiplication.
Technical Report 1702.02017, arXiv.

Xiao, J., Gu, M., and Langou, J. (2017).
Fast Parallel Randomized QR with Column Pivoting Algorithms for Reliable Low-rank Matrix Approximations.
2017 IEEE 24th International Conference on High Performance Computing (HiPC), pages 233–242.
arXiv: 1804.05138.