Communication-Avoiding Parallel Strassen: Implementation and Performance

Grey Ballard, Jim Demmel, Ben Lipshitz, Oded Schwartz

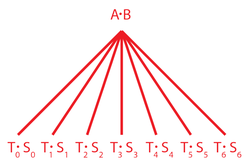



Summary

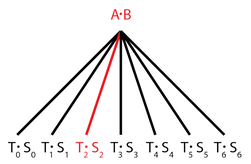
- Our new Communication-Avoiding Parallel Strassen (CAPS) algorithm is communication optimal [1]
  - matches the communication lower bounds [3]
  - moves asymptotically less data than all existing algorithms
- Our implementation is faster in practice [5]
  - than any classical algorithm can be
  - than any Strassen-based algorithm we are aware of
  - on multiple supercomputers and on a projected exascale machine
- We use a performance model to identify possible improvements
- Diagonal scaling can be used to improve Strassen's stability

CAPS Algorithm

- Key parallelization decision is to choose how to compute 7 subproblems
  - breadth first search (BFS) ordering or depth first search (DFS) ordering
- Independent decision can be made at each level of recursion tree



A BFS level of the recursion tree



A DFS level of the recursion tree

Comparison to Previous Strassen Approaches

Asymptotic Analysis

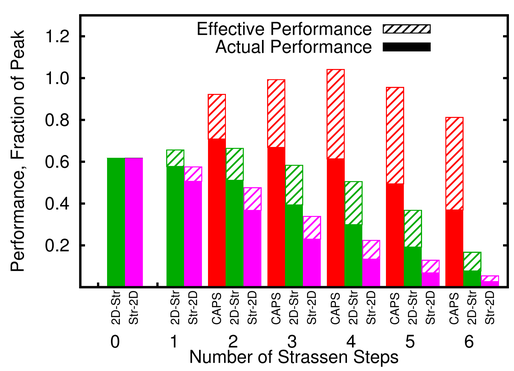
Theoretical asymptotic performance of CAPS and previous Strassen approaches compared with the lower bounds

Strassen	Flops	Bandwidth	Latency
Lower Bound [3]	$\frac{n^\omega}{P}$	$\max\left\{\frac{n^\omega}{PM^{\omega/2-1}}, \frac{n^2}{P^{2/\omega}}\right\}$	$\max\left\{\frac{n^\omega}{PM^{\omega/2}}, 1\right\}$
2D-Strassen [6]	$\frac{n^\omega}{P^{(\omega-1)/2}}$	$\frac{n^2}{P^{1/2}}$	$P^{1/2}$
Strassen-2D [4, 6]	$\left(\frac{7}{8}\right)^\ell \frac{n^3}{P}$	$\left(\frac{7}{4}\right)^\ell \frac{n^2}{P^{1/2}}$	$7^\ell P^{1/2}$
CAPS [1]	$\frac{n^\omega}{P}$	$\max\left\{\frac{n^\omega}{PM^{\omega/2-1}}, \frac{n^2}{P^{2/\omega}}\right\}$	$\max\left\{\frac{n^\omega}{PM^{\omega/2}}\log P, \log P\right\}$

n = Matrix dimension $\quad P$ = Number of processors
M = Local memory size $\quad \omega = \log_2 7 \approx 2.81$
ℓ = Number of Strassen steps taken

Performance Comparison

Measured performance for a fixed problem size ($n = 21952$) and number of processors ($P = 49$) on Intrepid, varying the number of Strassen steps taken



Strong Scaling Results

Strong scaling results for large and small problems on three machines

- Left column: effective performance on large matrices
  - Effective performance is measured as

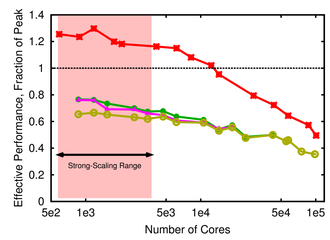
$$\text{Effective flop/s} = \frac{2n^3}{\text{Execution time in seconds}}$$

- Right column: execution time on small matrices
- Legend:

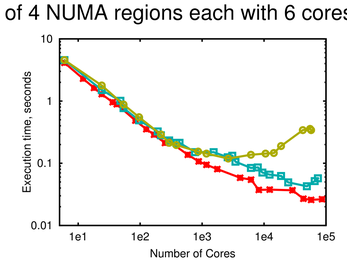
CAPS, 2.5D-Strassen, 2D-Strassen, Strassen-2D, 2.5D, 2D

Hopper (Cray XE6)

NERSC machine with nodes consisting of 4 NUMA regions each with 6 cores



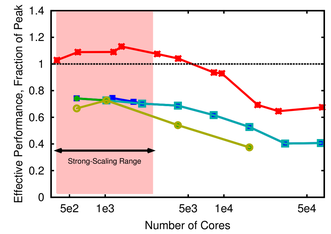
$n = 131712$



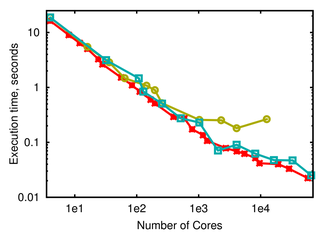
$n = 4704$

Intrepid (IBM BG/P)

ALCF machine with nodes consisting of 4 cores



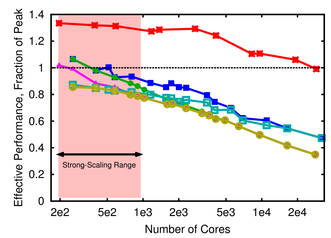
$n = 65856$



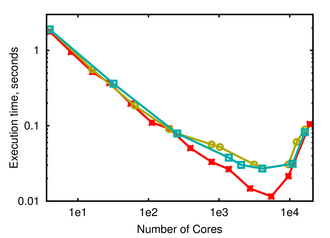
$n = 4704$

Franklin (Cray XT4)

NERSC machine with nodes consisting of 4 cores (retired)



$n = 94080$

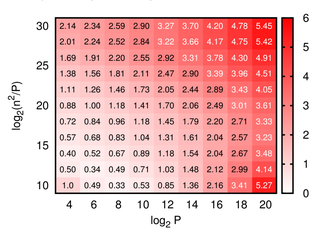


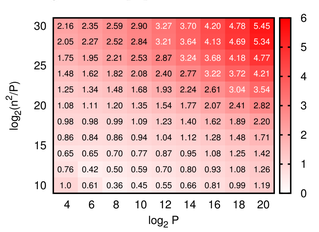
$n = 3136$

Exascale Predictions

Given projected parameters for an exascale machine, our model predicts the following speedups relative to classical algorithms

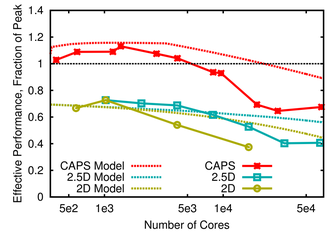
- Left: speedup relative to 2D classical algorithm [8]
- Right: speedup relative to 2.5D classical algorithm [7]

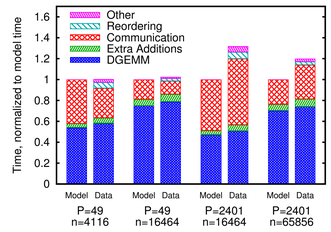




Performance Model

Performance model based on network parameters α and β and a sequential model which is based on measured `GEMM` and `AXPY` performance



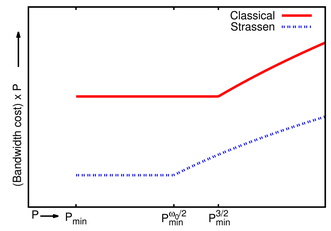


Theoretical Strong Scaling Range

Theoretical strong scaling ranges are

$$P_{\min} \le P \le P_{\min}^{\omega_0/2} \quad \text{for CAPS}$$
$$P_{\min} \le P \le P_{\min}^{3/2} \quad \text{for 2.5D}$$

where $P_{\min} \approx \frac{3n^2}{M}$, the minimum number of processors required to store the problem [2]



Diagonal Scaling for Stability

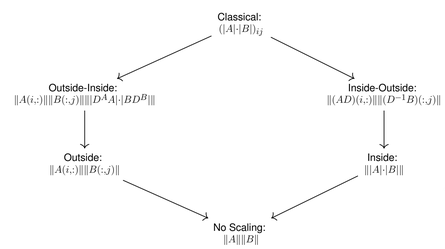
While Strassen does not have the same stability guarantees as classical matrix multiplication, we can choose diagonal matrices D, D^A, D^B and compute

$$D^A C D^B = (D^A A D) \cdot (D^{-1} B D^B)$$

and obtain tighter error bounds

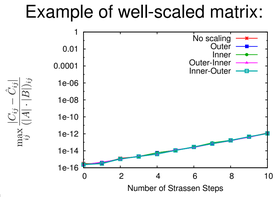
Hierarchy of bounds with scaling techniques

$$|C_{ij} - \hat{C}_{ij}| \le O(\epsilon) f(n) \cdot \ldots$$

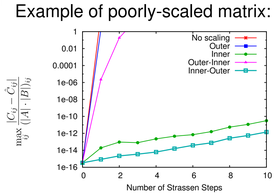


$X \to Y$ means that bound X is stronger than bound Y

Example of well-scaled matrix:



Example of poorly-scaled matrix:



References

[1] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. Communication-optimal parallel algorithm for Strassen's matrix multiplication. Technical Report EECS-2012-32, UC Berkeley, March 2012. To appear in SPAA 2012.

[2] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz. Strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds. Technical Report EECS-2012-31, UC Berkeley, March 2012. To appear in SPAA 2012.

[3] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Graph expansion and communication costs of fast matrix multiplication. In *Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, pages 1–12, New York, NY, USA, 2011. ACM.

[4] B. Grayson, A. Shah, and R. van de Geijn. A high performance parallel Strassen implementation. In *Parallel Processing Letters, Vol 6*, pages 3–12, 1995.

[5] B. Lipshitz, G. Ballard, O. Schwartz, and J. Demmel. Communication-avoiding parallel Strassen: Implementation and performance. Technical Report EECS-2012-90, UC Berkeley, May 2012. Submitted to SC 2012.

[6] Q. Luo and J. Drake. A scalable parallel Strassen's matrix multiplication algorithm for distributed-memory computers. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, SAC '95, pages 221–226, New York, NY, USA, 1995. ACM.

[7] E. Solomonik and J. Demmel. Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms. In *Euro-Par 2011, Part II*, volume 6853 of *Lecture Notes in Computer Science*, pages 90–109. Springer, 2011.

[8] R. A. van de Geijn and J. Watts. SUMMA: scalable universal matrix multiplication algorithm. *Concurrency - Practice and Experience*, 9(4):255–274, 1997.

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, and Samsung.