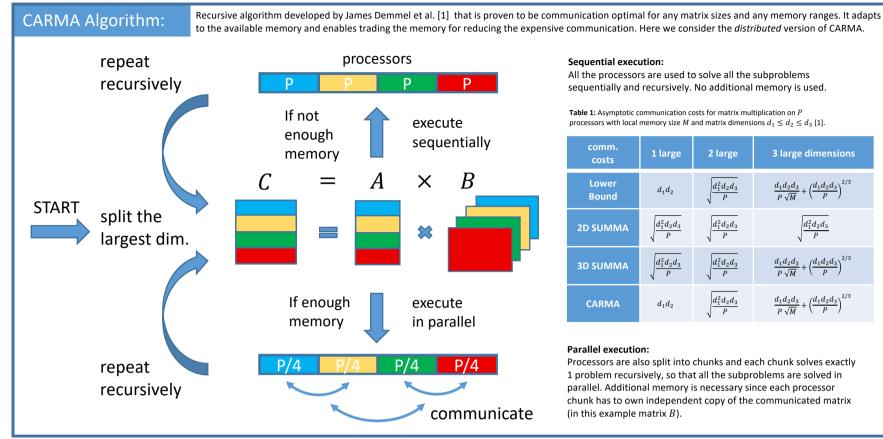
Practical Communication-Optimal Algorithm for Dense Matrix-Matrix Multiplication

Marko Kabić, Raffaele Solcà, Thibault Notargiacomo, Joost VandeVondele

Scientific Software & Libraries Group, Swiss National Supercomputing Center



Sequential execution:

All the processors are used to solve all the subproblems sequentially and recursively. No additional memory is used.

Table 1: Asymptotic communication costs for matrix multiplication on P processors with local memory size M and matrix dimensions $d_1 \leq d_2 \leq d_3$ [1].

comm. costs	1 large	2 large	3 large dimensions	
Lower Bound	d_1d_2	$\sqrt{\frac{d_1^2d_2d_3}{P}}$	$\frac{d_1 d_2 d_3}{P \sqrt{M}} + \left(\frac{d_1 d_2 d_3}{P}\right)^{2/3}$	
2D SUMMA	$\sqrt{\frac{d_1^2d_2d_3}{P}}$	$\sqrt{\frac{d_1^2d_2d_3}{P}}$	$\sqrt{\frac{d_1^2d_2d_3}{P}}$	
3D SUMMA	$\sqrt{\frac{d_1^2d_2d_3}{P}}$	$\sqrt{\frac{d_1^2d_2d_3}{P}}$	$\frac{d_1 d_2 d_3}{P \sqrt{M}} + \left(\frac{d_1 d_2 d_3}{P}\right)^{2/3}$	
CARMA	d_1d_2	$\sqrt{\frac{d_1^2d_2d_3}{P}}$	$\frac{d_1 d_2 d_3}{P \sqrt{M}} + \left(\frac{d_1 d_2 d_3}{P}\right)^{2/3}$	

Parallel execution:

Processors are also split into chunks and each chunk solves exactly 1 problem recursively, so that all the subproblems are solved in parallel. Additional memory is necessary since each processor chunk has to own independent copy of the communicated matrix (in this example matrix B).

Main Contributions:

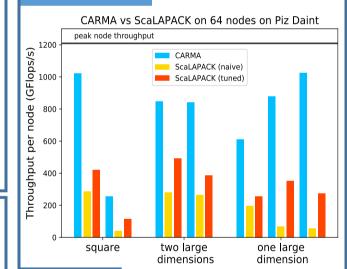
CARMA before:

- Only powers of 2: (m, n, k, P) assumed to be powers of 2 or that at each step the number of processor left and the largest dimension share common divisors.
- Cyclic data base-case layout: Requires complete data reshuffling after each communication. The corresponding mapper not provided. Compatibility issues with other layouts
- Impatient buffers allocations: every parallel step allocates and deallocates a new buffer. Buffers not reused.
- Limited division strategies: Not all division schedules produced correct results.

CARMA now:

- Generalized implementation: Works for any (m, n, k, P)!
- **Blocked data base-case layout:** Requires local data reshuffling only in some case and only on the level of blocks, instead of single elements.
- Less memory, more performance: Buffers carefully allocated and reused throughout the algorithm. Using ~25% less memory in total.
- Any division strategy available

Performance:



	square		two large		one large dimension		
m	64k	8k	64k	64k	64k	8704	17408
n	64k	8k	64k	8k	8k	8704	17408
k	64k	8k	8k	64k	8k	933888	3735552

Configuration

Piz Daint (multicore)	Cray XC40: 2x18-core Broadwell per node		
Number of nodes	64 nodes		
MPI implementation	Cray MPICH		
ScaLAPACK implementation	Intel MKL		

New Data Layout:

Cyclic Layout (before) before comm after comm • • • 000 \bullet \circ \bullet \circ \bullet aior ordering after interleaving

Blocked Layout (now)

 B_1 B_3 B_5

Idea: knowing the communication sche to minimize the local data reshufflings. dule, assign blocks to processors

 $B_1 \ B_2 \ B_3 \ B_4 \ B_5 \ B_6$

Properties:

Step 1: Divisions split matrix into blocks Step 2: Assign blocks to ranks: processors who communicate should own consecutive blocks.

A. Reshuffling **not** always needed and if needed: interleaves blocks, not elements. 2. Simpler mapper (using interval trees). 3. Better compatibility with other layouts.

Buffers Reuse:

entation decreases the total amount of memory used and the total number llocated. All the buffers are allocated just once and are then being reused

Before: each sequential subproblem allocates a new list of buffers Each next buffer has more information than the communication previous one, so no need to maintain all the buffers. All-gather

Evolution of a buffer holding local data of a single matrix

Now: All subproblems reuse the same buffers. Moreover, send and receive buffers keep swapping, so that only 2 buffers suffice If only parallel schedule is used then the number of allocated buffers goes from $O(\log P)$ to just 2 and the total memory used decreases by ~25%.

Keep swapping the two buffers, repeatedly using each once as a send buffer and once as a receive buffer.

receive buffer

References:

[1] Demmel, James, et al. "Communication-optimal parallel recursive rectangular matrix multiplication." Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on. IEEE. 2013.





Contact us:

source

github.com/eth-cscs/CARMA Marko Kabić (marko.kabic@cscs.ch)