Optimal communication for parallel matrix multiplication

Anton Mellit

ICTP, Italy

School on Parallel Programming and Parallel Architecture for HPC and Developer School for HPC applications in Earth Sciences, 2014

Main reference

G. Ballard and J. Demmel and O. Holtz and B. Lipshitz and O. Schwartz

Communication-optimal parallel algorithm for Strassen's matrix multiplication

Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA) 2012.

`http://arxiv.org/abs/1202.3173`

Why matrix multiplication?

- Still simple enough to explain
- Not trivial, so interesting ideas for parallelization can be illustrated

Idea

Simple algorithms are easily parallelizable, more sophisticated algorithms are much harder to parallelize efficiently (though not impossible).

Complexity of parallel algorithms

Serial algorithms

Time, memory. I.e. FFT: $O(N \log N)$. Common bruteforce tactics in cryptography: $O(2^N)$ time, $O(1)$ memory, meet in the middle approach: $O(2^{\frac{N}{2}})$ time, $O(2^{\frac{N}{2}})$ memory.

Parallel algorithms

New variable: the number of processors P. Ideal parallization: If serial time complexity is $T(N)$ and memory complexity is $M(N)$, the parallel version should have $\frac{T(N)}{P}$, $\frac{M(N)}{P}$. Usually impossible!

Massive parallelism: $P \to \infty$.

Communication complexity

Naive (Shared memory, Parallel random-access machine) is unrealistic.
In fact, as $P \to \infty$ parallel algorithms usually become *communication bound*

Conclusion

In a massively parallel setup we must optimize communication!

Things to worry about

- *Bandwidth cost BW* := Amount of data sent (per node)
- *Latency cost L* := Number of messages sent/received (per node)

Matrix multiplication (classical)

```
for i in range(N):
    for k in range(N):
        C[i,k] = 0
        for j in range(N):
            C[i,k] += A[i,j] * B[j,k]
```

Remark

Matrix multiplication respects block decompositions.

Serial complexity

$O(N^3)$

Ways to parallelize

Naive method

Split the resulting matrix (parallelize the outer loop) in an arbitrary way into blocks of size $\frac{N^2}{P}$.

2D

Split all the matrices into blocks of size $\frac{N}{\sqrt{P}} \times \frac{N}{\sqrt{P}}$.

3D

Split each of the 3 loops into $P^{\frac{1}{3}}$ blocks of length $\frac{N}{P^{\frac{1}{3}}}$.

Results $O(\cdot)$

	T	BW	L
Naive	$\frac{N^3}{P}$	$\frac{N^3}{P}$	P
2D	$\frac{N^3}{P}$	$\frac{N^2}{\sqrt{P}}$	$\sqrt{P}$
3D	$\frac{N^3}{P}$	$\frac{N^2}{P^{\frac{2}{3}}}$	$\log P$

Strassen's multiplication

$$\begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \cdot \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} = \begin{pmatrix} A_{00}B_{00} + A_{01}B_{10} & A_{00}B_{01} + A_{01}B_{11} \\ A_{10}B_{00} + A_{11}B_{10} & A_{10}B_{01} + A_{11}B_{11} \end{pmatrix}$$

$$Q_0 = A_{00}B_{00}$$
$$Q_1 = A_{01}B_{10}$$
$$Q_2 = (A_{10} + A_{11})(B_{01} - B_{00})$$
$$Q_3 = (A_{10} + A_{11} - A_{00})(B_{11} - B_{01} + B_{00})$$
$$Q_4 = (A_{00} - A_{10})(B_{11} - B_{01})$$
$$Q_5 = (A_{01} - A_{10} - A_{11} + A_{00})B_{11}$$
$$Q_6 = A_{11}(B_{11} - B_{01} + B_{00} - B_{10})$$

$$C_{00} = Q_0 + Q_1$$
$$C_{01} = Q_0 + Q_2 + Q_3 + Q_5$$
$$C_{10} = Q_0 + Q_3 + Q_4 - Q_6$$
$$C_{11} = Q_0 + Q_2 + Q_3 + Q_4$$

Only 7 multiplications!

Analysis

For problem of size $N = 2^n$ "only" $O(7^n) \approx O(N^{2.8074})$ operations needed comparing to $O(8^n) = O(N^3)$. This is not much, for $N = 10^5$ we have $N^{2.8074} \approx 10^{14}$ vs. $N^3 = 10^{15}$.
Assume we do r steps of this recursion and $N = 2^r N_0$.

$$\boxed{1 \text{ problem of size } 2^r N_0 \times 2^r N_0}$$
$$\Downarrow$$
$$\boxed{7 \text{ problems of size } 2^{r-1} N_0 \times 2^{r-1} N_0}$$
$$\Downarrow$$
$$\ldots$$
$$\Downarrow$$
$$\boxed{7^r \text{ problems of size } N_0 \times N_0}$$

How do we parallelize this?

Going parallel

Idea

Suppose we have $P = 7^r$ nodes, i.e. $7^4 = 2401$. We want each of the problems of size $N_0 \times N_0$ to be solved on one node.

Naive implementation

r all-to-all requests

Warning

Note that the memory usage grows exponentially! The original input had $\sim 4^r N_0^2$ words. After r recursion steps we need $\sim 7^r N_0^2$. This means that if memory is limited we may want to run first few recursion steps sequentially.

Optimizing communication

How do we map the 4^r pieces of the original input to 7^r nodes?

Idea

Distribute data in such a way that formations of the matrices Q_i are done locally at each recursion step.

The solution is to further split each of the 4^r pieces into 7^r pieces. Suppose $N = 2^r 7^{\frac{r}{2}} N_1$, so that both input and output matrices are split into $4^r 7^r$ square blocks of size $N_1 \times N_1$. Label each node by a sequence $b_1, b_2, \ldots, b_r$ with $b_i \in \{0, 1, \ldots, 6\}$. Label each block by a sequence $a_1, a_2, \ldots, a_r, b_1, b_2, \ldots, b_r$ with $a_i \in \{0, 1, 2, 3\}$, $b_i \in \{0, 1, \ldots, 6\}$. For $r = 2$ each of A, B, C are decomposed in 16 big matrices, each big matrix is decomposed into 49 small matrices. Initially the first node contains the first small matrix in each big matrix and so on.

Initial data distribution

Node $b_1, b_2, \ldots, b_r$ contains all the blocks
$a_1, a_2, \ldots, a_r, b_1, b_2, \ldots, b_r$.

After the first recursion step

Node $b_1, b_2, \ldots, b_r$ contains all the blocks $a_2, \ldots, a_r, *, b_2, \ldots, b_r$ of the matrices we need to multiply to obtain Q_{b_1}. This is achieved by an all-to-all request in each group of 7 nodes $*, b_2, \ldots, b_r$.

After the second recursion step

Node $b_1, b_2, \ldots, b_r$ contains all the blocks
$a_3, \ldots, a_r, *, *, b_3, \ldots, b_r$ of the matrices we need to multiply to obtain $Q_{b_1 b_2}$. This is achieved by an all-to-all request in each group of 7 nodes $b_1, *, b_3, \ldots, b_r$,

...

Conclusion

At each recursion step we need an all-to-all communication in a group of 7 nodes.

$$BW \sim 7 \cdot 4^r N_1^2 + 7^2 4^{r-1} N_1^2 + \cdots + 7^{r+1} N_1^2 \sim 7^r N_1^2 = \frac{N^2}{4^r} \approx \frac{N^2}{P^{0.7124}} < \frac{N^2}{P^{0.6667}}$$

$$L \sim 7r \sim \log P$$

$\approx 2\times$ speed-up on 2401 nodes for matrices of size 94080×94080.

Remarks

- Use libraries!
- Estimates do not reflect reality because of hidden constants.
- Estimates do not reflect reality because communication is more complicated.
- Data distribution is the key.