Improving Communication Lower Bounds for Matrix-Matrix Multiplication
Sparse Days 2014 - Toulouse, France

Bradley R. Lowery and Julien Langou
University of Colorado Denver
June 5, 2014
Algorithms have two costs (cost in time, energy, power):

1. **Computation**: Cost to perform computation
   - # of operations to be performed

2. **Communication**: Cost to move data
   - volume of data to be moved (bandwidth)
   - # of messages (latency)

**Motivations**

1. **Time**. On current architecture, communication is much slower than computation. Trend is not in favor of communication.

2. **Energy/Power**. Communication (moving data) consumes a lot of energy, power

3. **Co-design**.
Mission Statement

We study communication costs for the ordinary dense (OD) matrix-matrix multiplication in the sequential model.
Mission Statement

We study communication costs for the ordinary dense (OD) matrix-matrix multiplication in the sequential model.

- **dense**: because we wanted to totally be on topic for the Sparse Days.
Mission Statement

We study communication costs for the ordinary dense (OD) matrix-matrix multiplication in the **sequential** model.

- **dense**: because we wanted to totally be on topic for the *Sparse Days*.
- **sequential**: two levels of memory
  - fast memory of size $M$
  - slow memory
  - computation happens in fast memory
  - just look at volume of communication (bandwidth), no latency
Mission Statement

We study communication costs for the ordinary dense (OD) matrix-matrix multiplication in the sequential model.

- **dense**: because we wanted to totally be on topic for the *Sparse Days.*
- **sequential**: two levels of memory
  » fast memory of size $M$
  » slow memory
  » computation happens in fast memory
  » just look at volume of communication (bandwidth), no latency
- **ordinary**: we compute all ($n^3$)

$$c_{ijk} = a_{ik} \cdot b_{kj}$$

(consequence: Strassen-like matrix-matrix multiplications are not allowed.)
Important to realize that this generalizes to

- **# of messages** (latency related) (as opposed to “total volume of messages”, bandwidth related)
- **parallel distributed**
- **hierarchical memories**

![Sequential Model](image1)

![Hierarchical Model](image2)

![Parallel Model](image3)
Mission Statement

We study communication costs for the ordinary dense (OD) matrix-matrix multiplication in the sequential model.

Communication Cost for (OD) Matrix-Matrix Multiplication

Dense matrix-matrix multiplication moves $n^2$ data for $n^3$ computation.

$$ n \begin{bmatrix} C \\ \hline n \end{bmatrix} + = \begin{bmatrix} A \\ \hline \end{bmatrix} \times \begin{bmatrix} B \\ \hline \end{bmatrix} $$

- Computation cost is $2n^3$
  
  for $i=1:n$, for $j=1:n$, for $k=1:n$, $c_{ij} = c_{ij} + a_{ik}b_{kj}$; end; end; end;

- Communication cost is $3n^2$

Conclusion of the study

When $n$ increases, communication cost ($n^2$) becomes negligible with respect to computation cost ($n^3$).
\[
\beta^{-1} = 10^8 \text{words/sec} \quad \gamma^{-1} = 10^{10} \text{flops/sec} \quad M = 10^6 \text{words}
\]
Mission Statement
We study communication costs for the ordinary dense (OD) matrix-matrix multiplication in the sequential model.

Communication Cost for (OD) Matrix-Matrix Multiplication
Dense matrix-matrix multiplication moves $n^2$ data for $n^3$ computation.

\[
\begin{bmatrix}
C
\end{bmatrix}_n + = \begin{bmatrix}
A
\end{bmatrix} \times \begin{bmatrix}
B
\end{bmatrix}
\]

- Computation cost is $2n^3$
  for $i=1:n$, for $j=1:n$, for $k=1:n$, $c_{ij} = c_{ij} + a_{ik}b_{kj}$; end; end; end;
- Communication cost is $3n^2$

Conclusion of the study
When $n$ increases, communication cost ($n^2$) becomes negligible with respect to computation cost ($n^3$).
Limitation of the previous study: The previous study assumes that the three $n$-by-$n$ matrix $A$, $B$, and $C$ fit in cache.
- **Limitation of the previous study:** The previous study assumes that the three $n$-by-$n$ matrix $A$, $B$, and $C$ fit in cache.
- **Note:** this is a pretty serious limitation ... (In particular when $n$ goes to infinity ... )
- **Limitation of the previous study:** The previous study assumes that the three $n$-by-$n$ matrix $A$, $B$, and $C$ fit in cache.
- **Note:** this is a pretty serious limitation ... (In particular when $n$ goes to infinity ...)
- **Easy fix:** A common easy fix is to block the matrix-matrix multiplication with square blocks so that the square blocks fit in cache.

Let $M$ be the size of our cache. Let $b = \sqrt{\frac{M}{3}}$ (so that $3b^2 = M$). Then,

\[
\begin{align*}
\text{for } i &= 1:n/b, \text{ for } j = 1:n/b, \text{ for } k = 1:n/b, \\
\quad \quad b \begin{bmatrix}
C_{ij}
\end{bmatrix} &\quad = \quad A_{ik} \times B_{kj} \\
\end{align*}
\]

end; end; end;

Then, at each loop, we are moving $2b^2$ data and computing $2b^3$ so ... (Note: $C_{ij}$ stays in cache.)
Limitation of the previous study: The previous study assumes that the three $n$-by-$n$ matrix $A$, $B$, and $C$ fit in cache.

Note: this is a pretty serious limitation ...
(In particular when $n$ goes to infinity ... )

Easy fix: A common easy fix is to block the matrix-matrix multiplication with square blocks so that the square blocks fit in cache.

Let $M$ be the size of our cache. Let $b = \sqrt{\frac{M}{3}}$ (so that $3b^2 = M$). Then,

$$\text{for } i=1:n/b, \text{ for } j=1:n/b, \text{ for } k=1:n/b,$$

$$b \begin{bmatrix} C_{ij} \\ b \end{bmatrix} = A_{ik} \times B_{kj}$$

end; end; end;

Then, at each loop, we are moving $2b^2$ data and computing $2b^3$ so ...
(Note: $C_{ij}$ stays in cache.)

Computation cost is $\left(\frac{n}{b}\right)^3 (2b^3) \rightarrow 2n^3 \rightarrow \text{perfect.}$
- **Limitation of the previous study:** The previous study assumes that the three $n$-by-$n$ matrix $A$, $B$, and $C$ fit in cache.
- **Note:** this is a pretty serious limitation ...
  (In particular when $n$ goes to infinity ... )
- **Easy fix:** A common easy fix is to block the matrix-matrix multiplication with square blocks so that the square blocks fit in cache.

Let $M$ be the size of our cache. Let $b = \sqrt{\frac{M}{3}}$ (so that $3b^2 = M$). Then,

for $i = 1:n/b$, for $j = 1:n/b$, for $k = 1:n/b$,

$$
\begin{bmatrix}
C_{ij}
\end{bmatrix}
\begin{bmatrix}
A_{ik} \\
B_{kj}
\end{bmatrix}
= A_{ik} \times B_{kj}
$$

end; end; end;

Then, at each loop, we are moving $2b^2$ data and computing $2b^3$ so ... 
(Note: $C_{ij}$ stays in cache.)

- Computation cost is $(\frac{n}{b})^3 (2b^3) \rightarrow 2n^3 \rightarrow \text{perfect}$.
- Communication cost is $(\frac{n}{b})^3 (2b^2) \rightarrow (\frac{2}{b}) n^3 \rightarrow \text{oopsee}$.
We see that the previous algorithm
- performs $2n^3$ floating point operations
- performs a volume of data movement of

$$\left(\frac{2\sqrt{3}}{\sqrt{M}}\right)n^3.$$

Therefore the time of a OD matrix-matrix multiplication is

$$\left(\frac{2\sqrt{3}}{\sqrt{M}}\right)\beta n^3 + 2\gamma n^3$$

(1) assuming no overlap between communication and computations; (2) with $\beta$ being the time to move one unit of data (inverse of bandwidth) and $\gamma$ being the time to perform one floating-point operation.
We see that the previous algorithm
- performs $2n^3$ floating point operations
- performs a volume of data movement of

$$\left(\frac{2\sqrt{3}}{\sqrt{M}}\right)n^3.$$

Therefore the time of a OD matrix-matrix multiplication is

$$\left(\frac{2\sqrt{3}}{\sqrt{M}}\right) \beta n^3 + 2\gamma n^3$$

(1) assuming no overlap between communication and computations; (2) with $\beta$ being the time to move one unit of data (inverse of bandwidth) and $\gamma$ being the time to perform one floating-point operation.

**Study with $n$.** Communication is not negligible against computation. Both computation and communication are of order $n^3$. 
We see that the previous algorithm
- performs $2n^3$ floating point operations
- performs a volume of data movement of

$$\left(\frac{2\sqrt{3}}{\sqrt{M}}\right)^n.$$

Therefore the time of a OD matrix-matrix multiplication is

$$\left(\frac{2\sqrt{3}}{\sqrt{M}}\right) \beta n^3 + 2\gamma n^3$$

(1) assuming no overlap between communication and computations; (2) with $\beta$ being the time to move one unit of data (inverse of bandwidth) and $\gamma$ being the time to perform one floating-point operation.

**Study with $n$.** Communication is not negligible against computation. Both computation and communication are of order $n^3$.

*If $\beta/\sqrt{M} << \gamma$ then*, communication is negligible against computation.
Consider any ordinary dense matrix-matrix multiplication algorithm for multiplying an $m$–by–$n$ matrix with an $n$–by–$p$ matrix, consider a computer with fast memory of size $M$, then

**Theorem (Hong and Kung, 1981)**
Consider any ordinary dense matrix-matrix multiplication algorithm for multiplying an $m$–by–$n$ matrix with an $n$–by–$p$ matrix, consider a computer with fast memory of size $M$, then

**Theorem (Hong and Kung, 1981)**

The number of words transferred between slow and fast memory is at least

$$\frac{1}{2\sqrt{2}} \frac{mnp}{\sqrt{M}} - M.$$
Consider any ordinary dense matrix-matrix multiplication algorithm for multiplying an \( n \times n \) matrix with an \( n \times n \) matrix, consider a computer with fast memory of size \( M \), then

**Upper bound :: square tile matrix-matrix multiplication**

The number of words transferred between slow and fast memory is at most

\[
3.46 \left( \frac{n^3}{\sqrt{M}} \right).
\]

**Lower Bound :: Irony, Toledo, and Tiskin, 2004**

The number of words transferred between slow and fast memory is at least

\[
0.35 \left( \frac{n^3}{\sqrt{M}} \right) - M.
\]
The time of an OD matrix-matrix multiplication is

\[(?) \beta n^3 + 2\gamma n^3\]

(1) assuming no overlap between communication and computations; (2) with \(\beta\) being the time to move one unit of data (inverse of bandwidth) and \(\gamma\) being the time to perform one floating-point operation.

We know that (?) is between 0.35 and 3.46.