Distributed Matrix Multiply

Ned Nedialkov

McMaster University

7–9 February 2023

Outline

Fox's algorithm

SUMMA

- Cannon's algorithm 1969
  Fox's algorithm 1988
  - Assume square process grid $\sqrt{p} \times \sqrt{p}$
  - Nontrivial for non-square grids.
  - Don't work well when one of the matrix dimensions becomes relatively small.
- SUMMA: Scalable Universal Matrix Multiply Algorithm, 1997
  - avoids the above shortcomings
  - used in practice e.g. ScaLAPACK

See also Martin D. Schatz, Robert A. van de Geijn, and Jack Poulson. Matrix Multiplication: A Systematic Journey

Fox's algorithm

Let A and B be $n \times n$ matrices.

Compute $C = AB$ in parallel.

Let $q = \sqrt{p}$ be an integer such that it divides n, where p is the number of processes.

Create a Cartesian topology with processes (i, j), $i, j = 0, \ldots, q - 1$.

Denote $m = n/q$.

Distribute A and B by blocks such that A_{ij} and B_{ij} are $m \times m$, $m = n/q$, blocks stored on process (i, j).

On process (i,j), we want to compute

$$C_{i,j} = \sum_{k=0}^{q-1} A_{i,k}B_{k,j} = A_{i,0}B_{0,j} + A_{i,1}B_{1,j} + \cdots A_{i,i-1}B_{i-1,j}$$
$$+ A_{i,i}B_{i,j} + A_{i,i+1}B_{i+1,j} + \cdots A_{i,q-1}B_{q-1,j}$$

Rewrite this as

stage	compute
0	$C_{i,j} = A_{i,i}B_{i,j}$
1	$C_{i,j} \mathrel{+}= A_{i,i+1}B_{i+1,j}$
$\vdots$	$\vdots$
	$C_{i,j} \mathrel{+}= A_{i,q-1}B_{q-1,j}$
	$C_{i,j} \mathrel{+}= A_{i,0}B_{0,j}$
	$C_{i,j} \mathrel{+}= A_{i,1}B_{1,j}$
$\vdots$	$\vdots$
$q-1$	$C_{i,j} \mathrel{+}= A_{i,i-1}B_{i-1,j}$

Each process computes in stages

stage 0

- process (i,j) has $A_{i,j}$, $B_{i,j}$ but needs $A_{i,i}$
- process (i,i) broadcasts $A_{i,i}$ across processor row i
- process (i,j) computes $C_{i,j} = A_{i,i}B_{i,j}$

stage 1

- process (i,j) has $A_{i,j}$, $B_{i,j}$, but needs $A_{i,i+1}$, $B_{i+1,j}$
    - shift the jth block column of B by one block up (block 0 goes to block $q-1$)
    - process $(i,i+1)$ broadcasts $A_{i,i+1}$ across processor row i
- process (i,j) computes $C_{i,j} \mathrel{+}= A_{i,i+1}B_{i+1,j}$

Similarly on next stages

Example

Consider multiplying two 3×3 block matrices:

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

Process (i, j) stores A_{ij}, B_{ij}

A_{00}, B_{00}	A_{01}, B_{01}	A_{02}, B_{02}
A_{10}, B_{10}	A_{11}, B_{11}	A_{12}, B_{12}
A_{20}, B_{20}	A_{21}, B_{21}	A_{22}, B_{22}

and computes C_{ij}.

stage 0

process (i,i mod 3)	broadcasts along row i
$(0,0)$	A_{00}
$(1,1)$	A_{11}
$(2,2)$	A_{22}

A_{00}, B_{00} A_{00}	A_{00}, B_{01} A_{01}	A_{00}, B_{02} A_{02}
A_{11}, B_{10} A_{10}	A_{11}, B_{11} A_{11}	A_{11}, B_{12} A_{12}
A_{22}, B_{20} A_{20}	A_{22}, B_{21} A_{21}	A_{22}, B_{22} A_{22}

Process (i, j) computes

$$
\begin{array}{lll}
C_{00} = A_{00}B_{00} & C_{01} = A_{00}B_{01} & C_{02} = A_{00}B_{02} \\
C_{10} = A_{11}B_{10} & C_{11} = A_{11}B_{11} & C_{12} = A_{11}B_{12} \\
C_{20} = A_{22}B_{20} & C_{22} = A_{22}B_{21} & C_{12} = A_{22}B_{22}
\end{array}
$$

Shift-rotate on the columns of B:

A_{00}, B_{10} A_{00}	A_{00}, B_{11} A_{01}	A_{00}, B_{12} A_{02}
A_{11}, B_{20} A_{10}	A_{11}, B_{21} A_{11}	A_{11}, B_{22} A_{12}
A_{22}, B_{00} A_{20}	A_{22}, B_{01} A_{21}	A_{22}, B_{02} A_{22}

stage 1

process (i,(i+1)mod 3)	broadcasts along row i
$(0,1)$	A_{01}
$(1,2)$	A_{12}
$(2,0)$	A_{20}

A_{01}, B_{10} A_{00}	A_{01}, B_{11} A_{01}	A_{01}, B_{12} A_{02}
A_{12}, B_{20} A_{10}	A_{12}, B_{21} A_{11}	A_{12}, B_{22} A_{12}
A_{20}, B_{00} A_{20}	A_{20}, B_{01} A_{21}	A_{20}, B_{02} A_{22}

Process (i,j) computes

$$\begin{array}{lll} C_{00} \mathrel{+}= A_{01}B_{10} & C_{01} \mathrel{+}= A_{01}B_{11} & C_{02} \mathrel{+}= A_{01}B_{12} \\ C_{10} \mathrel{+}= A_{12}B_{20} & C_{11} \mathrel{+}= A_{12}B_{21} & C_{12} \mathrel{+}= A_{12}B_{22} \\ C_{20} \mathrel{+}= A_{20}B_{00} & C_{21} \mathrel{+}= A_{20}B_{01} & C_{22} \mathrel{+}= A_{20}B_{02} \end{array}$$

Shit-rotate on columns of B:

A_{01}, B_{20} A_{00}	A_{01}, B_{21} A_{01}	A_{01}, B_{22} A_{02}
A_{12}, B_{00} A_{10}	A_{12}, B_{01} A_{11}	A_{12}, B_{02} A_{12}
A_{20}, B_{10} A_{20}	A_{20}, B_{11} A_{21}	A_{10}, B_{02} A_{22}

stage 2

process (i,(i+2)mod 3)	broadcasts along row i
$(0,2)$	A_{02}
$(1,0)$	A_{10}
$(2,1)$	A_{21}

A_{02}, B_{20} A_{00}	A_{02}, B_{21} A_{01}	A_{02}, B_{22} A_{02}
A_{10}, B_{00} A_{10}	A_{10}, B_{01} A_{11}	A_{10}, B_{02} A_{12}
A_{21}, B_{10} A_{20}	A_{21}, B_{11} A_{21}	A_{21}, B_{12} A_{22}

Process (i, j) computes

$$\begin{array}{lll} C_{00} \mathrel{+}= A_{02}B_{20} & C_{01} \mathrel{+}= A_{02}B_{21} & C_{02} \mathrel{+}= A_{02}B_{22} \\ C_{10} \mathrel{+}= A_{10}B_{00} & C_{11} \mathrel{+}= A_{10}B_{01} & C_{12} \mathrel{+}= A_{10}B_{02} \\ C_{20} \mathrel{+}= A_{21}B_{10} & C_{21} \mathrel{+}= A_{21}B_{11} & C_{22} \mathrel{+}= A_{21}B_{12} \end{array}$$

Parallel time

For mesh architecture:

$$T_p = \frac{n^3}{p} + 2t_w \frac{n^2}{\sqrt{p}} + t_s p \tag{1}$$

On a hypercube, the parallel execution time can be improved to

$$T_p = \frac{n^3}{p} + 2t_w \frac{n^2}{\sqrt{p}} + t_s \sqrt{p} \log p + 2n\sqrt{t_s t_w \log p} \tag{2}$$

See A. Gupta and V. Kumar. Scalability of Parallel Algorithms for Matrix Multiplication

The speedup in (1) is

$$S = \frac{n^3}{\frac{n^3}{p} + 2t_w \frac{n^2}{\sqrt{p}} + t_s p} = \frac{1}{\frac{1}{p} + 2t_w \frac{1}{n\sqrt{p}} + t_s \frac{p}{n^3}}$$

Example

Consider a 2×2 process grid. Process (i, j) stores A_{ij} and B_{ij}:

A_{00}, B_{00}	A_{01}, B_{01}
A_{10}, B_{10}	A_{11}, B_{11}

process	broadcasts	across
$(0, 0)$	A_{00}	row 0
$(1, 0)$	A_{10}	row 1
$(0, 0)$	B_{00}	column 0
$(0, 1)$	B_{01}	column 1

A_{00}, B_{00} $\color{red}{A_{00}}, \color{blue}{B_{00}}$	A_{01}, B_{01} $\color{red}{A_{00}}, \color{blue}{B_{01}}$
A_{10}, B_{10} $\color{red}{A_{10}}, \color{blue}{B_{00}}$	A_{11}, B_{11} $\color{red}{A_{10}}, \color{blue}{B_{01}}$

Process (i, j) does:

$C_{00} = \color{red}{A_{00}}\color{blue}{B_{00}}$	$C_{01} = \color{red}{A_{00}}\color{blue}{B_{01}}$
$C_{10} = \color{red}{A_{00}}\color{blue}{B_{00}}$	$C_{11} = \color{red}{A_{10}}\color{blue}{B_{01}}$

Process (i,j) stores A_{ij} and B_{ij}:

A_{00}, B_{00}	A_{01}, B_{01}
A_{10}, B_{10}	A_{11}, B_{11}

process	broadcasts	across
$(0,1)$	A_{01}	row 0
$(1,1)$	A_{11}	row 1
$(1,0)$	B_{10}	column 0
$(1,1)$	B_{11}	column 1

A_{00}, B_{00} $\color{red}{A_{01}}, \color{blue}{B_{10}}$	A_{01}, B_{01} $\color{red}{A_{01}}, \color{blue}{B_{11}}$
A_{10}, B_{10} $\color{red}{A_{11}}, \color{blue}{B_{10}}$	A_{11}, B_{11} $\color{red}{A_{11}}, \color{blue}{B_{11}}$

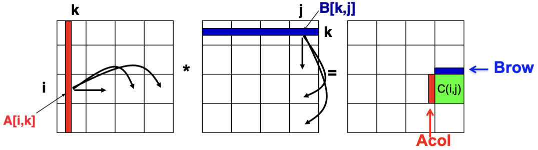
Process (i,j) does:

$C_{00} \mathrel{+}= \color{red}{A_{01}}\color{blue}{B_{10}}$	$C_{01} \mathrel{+}= \color{red}{A_{01}}\color{blue}{B_{11}}$
$C_{10} \mathrel{+}= \color{red}{A_{11}}\color{blue}{B_{10}}$	$C_{11} \mathrel{+}= \color{red}{A_{11}}B_{11}$

This computes

$$
\begin{aligned}
\begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix} &= \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{bmatrix} \\
&= \begin{bmatrix} A_{00}B_{00} & A_{00}B_{01} \\ A_{10}B_{00} & A_{10}B_{01} \end{bmatrix} + \begin{bmatrix} A_{01}B_{10} & A_{01}B_{10} \\ A_{11}B_{10} & A_{11}B_{11} \end{bmatrix}
\end{aligned}
$$

SUMMA



$A[i,k]$ is a block matrix, similarly $B[k,j]$. $p_r \times p_c$ process grid.

For $k = 0 : n/b - 1$ % b is block size
 for all $i = 1 : p_r$ in parallel
  owner of $A[i,k]$ broadcasts it to whole processor row
 for all $j = 1 : p_c$ in parallel
  owner of $B[k,j]$ broadcasts it to whole processor column
 receive $A[i,k]$ into Acol
 receive $B[k,j]$ into Brow
 C_myproc = C_myproc +Acol*Brow

See also ▸ J. Demmel. Dense Linear Algebra: History and Structure, Parallel Matrix Multiplication