SUMMA

Scalable Universal Matrix Multiplication Algorithm
Naïve matrix multiply

For $i = 0$ to $n$
   For $j = 0$ to $n$
      For $k = 0$ to $n$
         $C[i,j] += A[i,k]*B[k,j]$  

Calculates $n^2$ dot products (inner products)
$C[i,j] = A[i,:]*B[:,j]$
Naïve matrix multiply

What happens if we switch the order?

For $k = 0$ to $n$
  For $i = 0$ to $n$
    For $j = 0$ to $n$
      $C[i,j] += A[i,k] * B[k,j]$
Naïve matrix multiply

What happens if we switch the order?

For \( k = 0 \) to \( n \)
    For \( i = 0 \) to \( n \)
        For \( j = 0 \) to \( n \)
            \( C[i,j] += A[i,k] \times B[k,j] \)

Calculates \( n \) outer products
Outer product

\[
\begin{align*}
\begin{bmatrix}
  a \\
  b \\
  c
\end{bmatrix} * \begin{bmatrix}
  e & f & g
\end{bmatrix} &=
\begin{bmatrix}
  a*e & a*f & a*g \\
  b*e & b*f & b*g \\
  c*e & c*f & c*g
\end{bmatrix}
\end{align*}
\]
Introducing SUMMA

- Processors arranged in grid, \( P(i,j) \)
- Example, 2x2 grid:

\[
\begin{array}{cccc}
P(1,1) & P(1,2) & \times & P(1,1) & P(1,2) \\
P(2,1) & P(2,2) & & P(2,1) & P(2,2) \\
\end{array}
\]

\[
\begin{array}{cccc}
P(1,1) & P(1,2) & \times & P(1,1) & P(1,2) \\
P(2,1) & P(2,2) & & P(2,1) & P(2,2) \\
\end{array}
\]
Introducing SUMMA

- Let $A=mxk$, $B=kxn \Rightarrow C=mxn$
- Let each process do $k$ outer products

How do we handle the communication?
SUMMA

- For each k (between 0 and n-1),
  - Owner of partial row K broadcasts that row along its process column
  - Owner of partial column K broadcasts that column along its process row

Image credit: Stephen J. Fink
SUMMA

Complete algorithm. On each process $P(i,j)$:

For $k = 0...n-1$

- Bcast column $k$ of $A$ ($a_i$) within row $i$
- Bcast row $k$ of $B$ ($b_j$) within column $j$
- Do $C +=$ outer product ($a_i$,$b_j$)
Communication cost

• What's the cost of communication? Let $\alpha$ be the startup cost of a message, and $\beta$ be the bandwidth
Communication cost

- What's the cost of communication? Let $\alpha$ be the startup cost of a message, and $\beta$ be the bandwidth.
- Bcast among $p$ processes takes $\log(p) \cdot (\alpha + \beta \cdot s)$ time, where $s$ is the size of the message.
Communication cost

- Bcast among $p$ processes takes $\log(p) (\alpha + \beta s)$ time, where $s$ is the size of the message
- For each $k$, there are one Bcast along columns and one Bcast along rows
- Each partial column has size $m/r$, each partial row has size $n/c$
Communication cost

- Bcast among $p$ processes takes $\log(p)$ $(\alpha+\beta s)$ time, where $s$ is the size of the message
- Total: $k*(\log(c)(\alpha+\beta m/r)+\log(r)(\alpha+\beta n/c))$
- Not very efficient
  - Lots of messages
Improvements?

• Can we reduce the communication cost somehow?
Improvements?

- Can we reduce the communication cost somehow?
- Obvious improvement: Instead of broadcasting single rows and columns, do block rows and columns.
Improvements?

- Obvious improvement: Instead of broadcasting single rows and columns, do block rows and columns.
- Same amount of bytes communicated
- Fewer messages => less overhead
- More efficient (O(n^3) FLOPS, O(n^2) loads)
Pipelined SUMMA

• Another improvement:

• Instead of broadcast one row/column segment to all, pass multiple segments around in a ring

• Each process only communicates with neighbor => no broadcast => fewer messages
Pipelined SUMMA - Example

- Consider the following matrices A and B (color represents process rank)
Pipelined SUMMA - Example

- K=0: All processes multiply their first (block) column/row
Pipelined SUMMA - Example

- K=1: All processes send their first (block) col in A to the right, their first (block) row in B down in a ring pattern.
Pipelined SUMMA - Example

- And so on

(for reference)