

Lecture 9

Communication Avoiding Algorithms
Matrix multiplication (SUMMA)

Announcements

- Project proposal: in class on Thursday
 - 1-2 page written proposal
 - 5-7 minute in-class presentation

Today's lecture

- Communication Avoiding Algorithms
- SUMMA matrix multiplication algorithm (Hallgeir Lien)

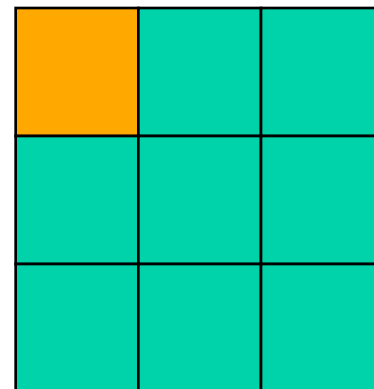
Communication lower bounds (early results)

- Assume we are using an $O(n^3)$ algorithm
- Let M = Size fast memory (cache/local memory)
- Sequential case: # slow memory references
 $\Omega(n^3 / \sqrt{M})$ [Hong and Kung '81]
- Parallel, p = # processors,
 μ = Amount of memory needed to store matrices
 - Refs to remote memory
 $\Omega(n^3 / (p\sqrt{\mu}))$ [Irony, Tiskin, Toledo, '04]
 - If $\mu = 3n^2/p$ (one copy of A, B, C) $\Rightarrow$
lower bound = $\Omega(n^2 / \sqrt{p})$ words
 - Achieved by Cannon's algorithm ("2D algorithm")
 - $T_p = 2n^3/p + 4\sqrt{p}(\alpha + \beta n^2/p)$

Canon's Algorithm - optimality

- General result
 - If each processor has M words of local memory ...
 - ... at least 1 processor must transmit $\Omega (\# \text{ flops} / M^{1/2})$ words of data
- If local memory $M = O(n^2/p)$...
 - at least 1 processor performs $f \geq n^3/p$ flops
 - ... lower bound on number of words transmitted by at least 1 processor

$$\begin{aligned}\Omega ((n^3/p) / (n^2/p)^{1/2}) &= \Omega ((n^3/p) / M^{1/2}) \\ &= \Omega (n^2 / p^{1/2})\end{aligned}$$



Limitations of Cannon's Algorithm

- Difficult to generalize
 - P is not a perfect square
 - A and B are not square, and not evenly divisible by $\sqrt{p}$
- Interoperation with applications and other libraries difficult or expensive
- The SUMMA algorithm offers a practical alternative
 - Uses a shift algorithm to broadcast
 - A variant used in SCALAPACK

R. VAN DE GEIGN AND J. WATTS,
“SUMMA: Scalable universal matrix multiplication algorithm,”
Concurrency: Practice and Experience, **9**:255-74 (1997)
www.netlib.org/lapack/lawns/lawn96.ps

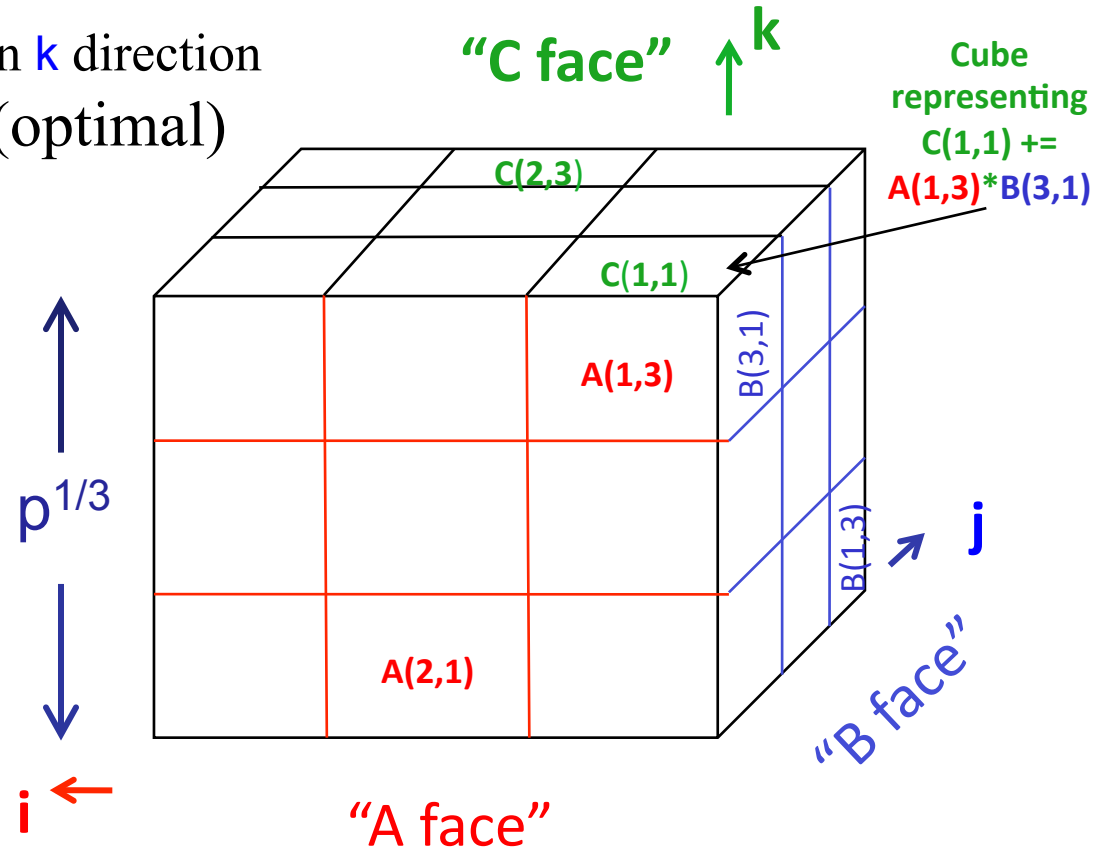
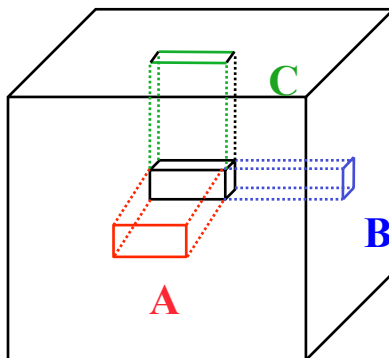
- Hallgeir Lien

New communication lower bounds – direct linear algebra [Ballard & Demmel '11]

- Let M = amount of fast memory per processor
- Lower bounds
 - # words moved by at least 1 processor
 $\Omega(\# \text{ flops} / M^{1/2})$
 - # messages sent by at least 1 processor
 $\Omega(\# \text{ flops} / M^{3/2})$
- Holds not only for Matrix Multiply but many other “direct” algorithms in linear algebra
- Identify 3 values of M
 - 2D (Cannon’s algorithm)
 - 3D (Johnson’s algorithm)
 - 2.5D (Ballard and Demmel)

Johnson's 3D Algorithm

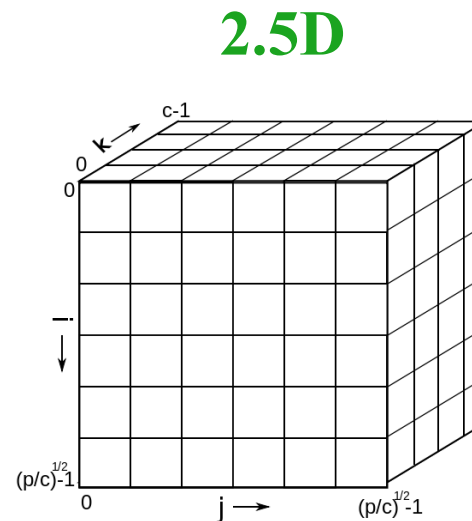
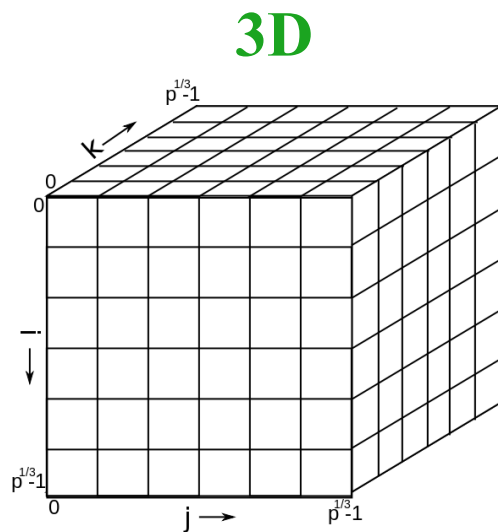
- 3D processor grid: $p^{1/3} \times p^{1/3} \times p^{1/3}$
 - Bcast A (B) in j (i) direction ($p^{1/3}$ redundant copies)
 - Local multiplications
 - Accumulate (Reduce) in k direction
- Communication costs (optimal)
 - Volume = $O(n^2/p^{2/3})$
 - Messages = $O(\log(p))$
- Assumes space for $p^{1/3}$ redundant copies
- Trade memory for communication



Source: Edgar Solomonik

2.5D Algorithm

- What if we have space for only $1 \leq c \leq p^{1/3}$ copies ?
- $M = \Omega(c \cdot n^2 / p)$
- Communication costs : lower bounds
 - Volume = $\Omega(n^2 / (cp)^{1/2})$; Set $M = c \cdot n^2 / p$ in $\Omega(\# \text{ flops} / M^{1/2})$
 - Messages = $\Omega(p^{1/2} / c^{3/2})$; Set $M = c \cdot n^2 / p$ in $\Omega(\# \text{ flops} / M^{3/2})$
- 2.5D algorithm “interpolates” between 2D & 3D algorithms



Source: Edgar Solomonik

2.5D Algorithm

- Interpolate between 2D (Cannon) and 3D
 - c copies of A & B
 - Perform $p^{1/2}/c^{3/2}$ Cannon steps on each copy of A&B
 - Sum contributions to C over all c layers
- Communication costs (not quite **optimal**, but not far off)

- Volume:

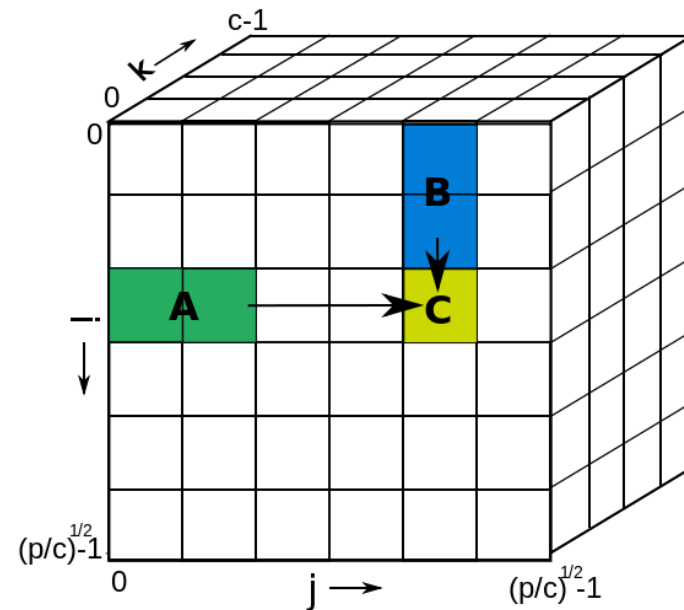
$$O(n^2 / (cp)^{1/2})$$

$$[\Omega(n^2 / (cp)^{1/2})]$$

- Messages:

$$O(p^{1/2} / c^{3/2} + \log(c))$$

$$[\Omega(p^{1/2} / c^{3/2})]$$



Source: Edgar Solomonik