CS 598: Communication Cost Analysis of Algorithms

Lecture 3: communication avoiding algorithms for matrix multiplication

Edgar Solomonik

University of Illinois at Urbana-Champaign

August 29, 2016

Review of LogP, LogGP, and BSP

- LogP model
  - separates network latency from sequential messaging overhead
  - permits overlap between communication and computation or communication with another processor
- LogGP model
  - additional parameter G controls large message bandwidth
  - eliminates packet size that was implicit in LogP
- BSP model
  - each processor sends/receives up to h messages every superstep
  - cost of superstep defined based on greatest amount of computation and communication done by any processor
- LogP and BSP can simulate each other (Bilardi et al 1999)
  - BSP can simulate LogP with constant slowdown
  - LogP can simulate BSP with $O(\log(P))$ slowdown

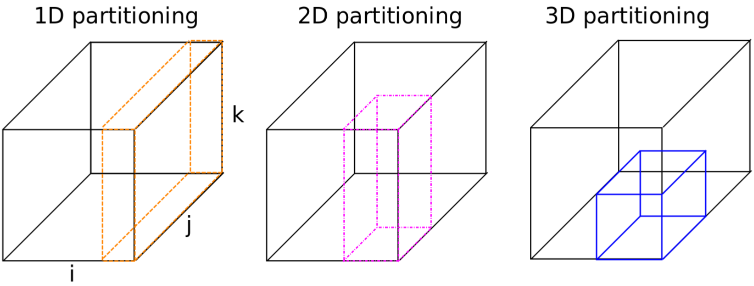
Minimizing surface to volume ratio

Matrix multiplication of n-by-n matrices A and B into C is

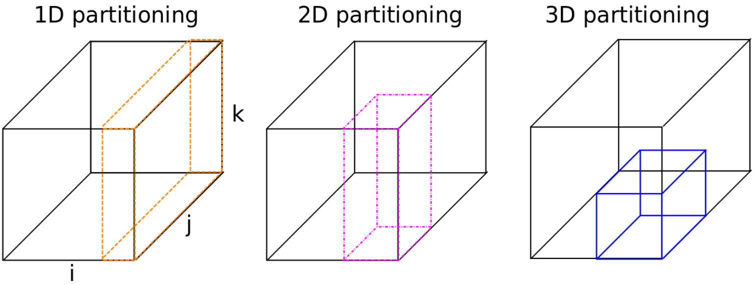
$$C(i,j) = \sum_k A(i,k) \cdot B(k,j)$$

The computation can be visualized as a 3D tensor

$$T(i,j,k) = (A(i,k), B(k,j), C(i,j))$$



Minimizing surface to volume ratio



Consider partitioning T into cuboids of size m as in the diagram

- 1D (blocking along one index): surface area is $O(n^2)$
- 2D (evenly blocking along two indices): surface area is $O(\sqrt{mn})$
- 3D (evenly blocking along all three indices): surface area is $O(m^{2/3})$

Minimizing surface to volume ratio

The surface area of a cuboid in T corresponds to the elements of A, B, and C needed to compute all of the elements in it
Moreover, the best surface area to volume ratio of any subset of T is achieved by selecting a cube.

Theorem (Loomis-Whitney (3D version), 1949)

Let V be a set of 3-tuples $V \subseteq [1, n]^3$

$$|V| \leq \sqrt{|\pi_1(V)||\pi_2(V)||\pi_3(V)|}$$

where

$$\pi_1(V) = \{(i_2, i_3) : \exists i_1, (i_1, i_2, i_3) \in V\}$$
$$\pi_2(V) = \{(i_1, i_3) : \exists i_2, (i_1, i_2, i_3) \in V\}$$
$$\pi_3(V) = \{(i_1, i_2) : \exists i_3, (i_1, i_2, i_3) \in V\}$$

General Loomis-Whitney inequality

Theorem (Discrete Loomis-Whitney Inequality)

Consider any $V \subseteq [1, n]^d$. Then we have

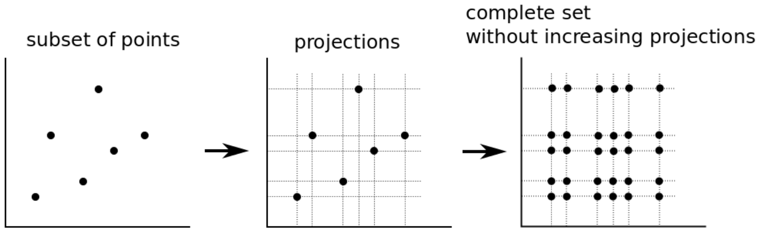
$$|V| \leq \left(\prod_{j=1}^{d} |\pi_j(V)| \right)^{1/(d-1)},$$

where, for $j \in [1, d]$, $\pi_j : [1, n]^d \to [1, n]^{d-1}$ is the projection

$$\pi_j(i_1, \ldots, i_d) = (i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_d).$$

Proof of Loomis-Whitney inequality in 2 dimensions

Theorem for $d = 2$: $|V| \leq |\pi_1(V)||\pi_2(V)|$



dashed lines denote projections, for any set V define $W = \pi_1(V) \otimes \pi_2(V)$, $V \subseteq W$ and $|V| \leq |W| = |\pi_1(V)||\pi_2(V)|$.

Proof of Loomis-Whitney inequality in 3 dimensions

Theorem for $d = 3$: $|V| \leq \sqrt{|\pi_1(V)||\pi_2(V)||\pi_3(V)|}$

Determine number of indices contained in V in each of three dimensions

$$k_1 = |\{i_1 : (i_1, i_2, i_3) \in V\}|$$
$$k_2 = |\{i_2 : (i_1, i_2, i_3) \in V\}|$$
$$k_3 = |\{i_3 : (i_1, i_2, i_3) \in V\}|$$

Enumerate the hyperplanes $(i_1, :, :) \subseteq V$ adjacent to the k_1 unique i_1 in V, $V_i \subseteq V$ for $i \in [1, k_1]$

Define vectors $p_2(i) = |\pi_2(V_i)|$ and $p_3(i) = |\pi_3(V_i)|$

The projections are disjoint, so $|\pi_2(V)| = \sum_{i=1}^{k_1} p_2(i)$, $|\pi_3(V)| = \sum_{i=1}^{k_1} p_3(i)$

By the $d = 2$ case, we have that $|V| \leq \sum_{i=1}^{k_1} p_2(i)p_3(i)$,

Proof of Loomis-Whitney inequality in 3 dimensions

We arrive at an optimization problem

$$\max\left(\sum_{i=1}^{k_1} p_2(i)p_3(i)\right), \qquad |\pi_2(V)| = \sum_{i=1}^{k_1} p_2(i), \quad |\pi_3(V)| = \sum_{i=1}^{k_1} p_3(i)$$

We can apply the method of Lagrange multipliers

$$f(\vec{p}, \vec{q}, \lambda_1, \lambda_2) = \sum_{i=1}^{k_1} p_2(i)p_3(i) + \lambda_1\left(|\pi_2(V)| - \sum_{i=1}^{k_1} p_2(i)\right) + \lambda_2\left(|\pi_3(V)| - \sum_{i=1}^{k_1} p_3(i)\right)$$

Now we fine the critical point $\nabla f = 0$, differntiating with respect to each variable leads to the following constraints

$$\forall i, \quad p_2(i) = \lambda_2, \quad p_3(i) = \lambda_1 \quad \text{and} \quad |\pi_2(V)| = \sum_{i=1}^{k_1} p_2(i), \quad |\pi_3(V)| = \sum_{i=1}^{k_1} p_3(i)$$

So, all hyperplanes have the same dimensions: $p_2(i) = \frac{|\pi_2(V)|}{k_1}$, $p_3(i) = \frac{|\pi_3(V)|}{k_1}$

Proof of Loomis-Whitney inequality for $d = 3$ cont'd

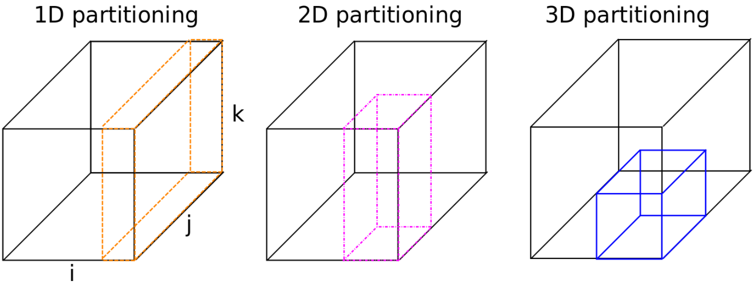
We have shown that if we consider the unique points in V along the first dimension, the size of V is maximized for projections $\pi_2(V)$ and $\pi_3(V)$ of any given size, when the hyperplanes are of equal dimensions $\frac{|\pi_2(V)|}{k_1} \times \frac{|\pi_3(V)|}{k_1}$

The projection $\pi_1(V)$ is clearly minimized by aligning these hyperplanes

Therefore, the optimal shape of V is a $k_1 \times k_2 \times k_3$ cuboid with faces of size $|\pi_1(V)| = k_2 k_3$, $|\pi_2(V)| = k_1 k_3$, $|\pi_3(V)| = k_1 k_2$

Thus, $|V| \leq k_1 k_2 k_3 = \sqrt{|\pi_1(V)||\pi_2(V)||\pi_3(V)|}$ $\square$

Minimizing surface to volume ratio



Consider partitioning T into cuboids of size $m = n^3/P$ as in the diagram

- 1D (blocking along one index): surface area is $O(n^2)$
- 2D (evenly blocking along two indices): surface area is $O(n^2/\sqrt{P})$
- 3D (evenly blocking along all three indices): surface area is $O(n^2/P^{2/3})$

Matrix multiplication

Matrix multiplication of n-by-n matrices A and B into C, $C = A \cdot B$ is defined as, for all i, j,

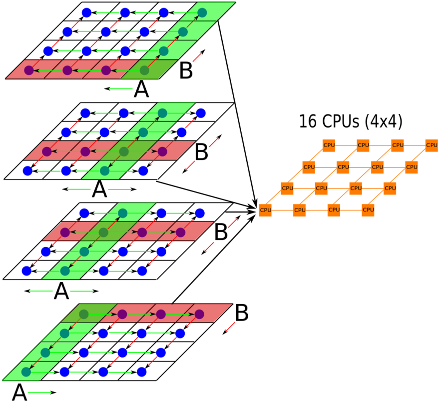
$$C(i,j) = \sum_k A(i,k) \cdot B(k,j)$$

A standard approach to parallelization of matrix multiplication is commonly referred to as **SUMMA** (Agarwal et al. 1995, Van De Geijn et al. 1997), which uses a 2D processor grid, so blocks A_{lm}, B_{lm}, and C_{lm} are owned by processor $\Pi(l, m)$

- SUMMA variant 1: iterate for $k = 1$ to $\sqrt{P}$ and for all $i, j \in [1, \sqrt{P}]$
  - broadcast A_{ik} to $\Pi(i, :)$
  - broadcast B_{kj} to $\Pi(:, j)$
  - compute $C_{ij} = C_{ij} + A_{ik} \cdot B_{kj}$ with processor $\Pi(i, j)$

The ScaLAPACK library (Blackford et al 1997) uses this type of algorithms

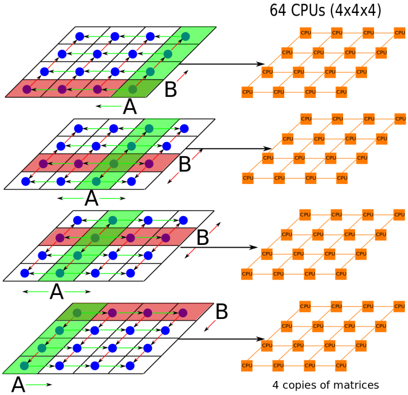
SUMMA algorithm



$$T^{\alpha,\beta}_{\mathrm{SUMMA}} = 2\sqrt{P} \cdot T^{\alpha,\beta}_{\mathrm{broadcast}}(n^2/P, \sqrt{P}) \leq 2\sqrt{P} \cdot \log(P) \cdot \alpha + \frac{4n^2}{\sqrt{P}} \cdot \beta$$

3D Matrix multiplication algorithm

Reference: Agarwal et al. 1995 and others



$$T^{\alpha,\beta}_{\mathrm{3D-MM}} = 2\,T^{\alpha,\beta}_{\mathrm{broadcast}}(n^2/P^{2/3}, P^{1/3}) + T^{\alpha,\beta}_{\mathrm{reduce}}(n^2/P^{2/3}, P^{1/3})$$
$$\leq 2\log(P)\cdot\alpha + \frac{6n^2}{P^{2/3}}\cdot\beta$$

Matrix multiplication with a cyclic layout

We now consider SUMMA a cyclic distribution on a 2D processor grid, so processor $\Pi(l, m)$ owns each $A(i, j)$, $B(i, j)$, and $C(i, j)$ with $i \equiv l \mod \sqrt{P}$ and $j \equiv m \mod \sqrt{P}$

- SUMMA variant 1: iterate for $k = 1$ to $\sqrt{P}$ and for all $i, j \in [1, \sqrt{P}]$
  - allgather block A_{ik} to $\Pi(i, :)$
  - allgather block B_{kj} to $\Pi(:, j)$
  - compute $C_{ij} = C_{ij} + A_{ik} \cdot B_{kj}$ with processor $\Pi(i, j)$

The Elemental library (Poulson et all 2013) uses this layout

SUMMA with a cyclic layout

The advantage of a cyclic layout is due to allgather costinga factor of two less than broadcast

- new cost for 2D SUMMA

$$T^{\alpha,\beta}_{\mathrm{SUMMA-ag}} = 2\sqrt{P} \cdot T^{\alpha,\beta}_{\mathrm{allgather}}(n^2/p, \sqrt{P}) \leq \sqrt{P} \cdot \log(P) \cdot \alpha + \frac{2n^2}{\sqrt{P}} \cdot \beta$$

- new cost for 3D algorithm

$$T^{\alpha,\beta}_{\mathrm{3D-MM-ag}} = 2\,T^{\alpha,\beta}_{\mathrm{allgather}}(n^2/P^{2/3}, P^{1/3}) + T^{\alpha,\beta}_{\mathrm{reduce}}(n^2/P^{2/3}, P^{1/3})$$
$$\leq (4/3)\log(P) \cdot \alpha + \frac{4n^2}{P^{2/3}} \cdot \beta$$

Other SUMMA variants

Instead of moving the data of A and B, we can alternatively move C

- rather than have $\Pi(i,j)$ compute block $C_{ij} = \sum_{k=1}^{n/\sqrt{P}} A_{ik} \cdot B_{kj}$,
  - have $\Pi(i,j)$ compute $\bar{C}_{ik} = A_{ij} \cdot B_{jk}$ for all $k \in [1, \sqrt{P}]$
  - then reduce(-scatter) along fibers $\Pi(i,:)$ to get $\sum_{k=1}^{n/\sqrt{P}} \bar{C}_{ik}$
- can similarly keep B in place and move A and C
- the three SUMMA versions have the same cost for square matrices, but different costs when matrices are rectangular (have different size)
- they also work for different initial distributions of A, B, and C

Recursive matrix multiplication

Now lets consider a recursive parallel algorithm for matrix multiplication

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

This requires 8 recursive calls to matrix multiplication of $n/2$-by-$n/2$ matrices, as well as matrix additions at each level, which can be done in linear time

Recursive matrix multiplication: analysis

If we execute all 8 recursive multiplies in parallel with $P/8$ processors, we obtain a cost recurrence of

$$T_{\mathrm{MM}}^{\alpha,\beta}(n,P) = T_{\mathrm{MM}}^{\alpha,\beta}(n/2,P/8) + O(\alpha) + O\left(\frac{n^2}{P}\cdot\beta\right)$$

The bandwidth cost is dominated by the base cases, where it is proportionate to

$$\left(n/2^{\log_8(P)}\right)^2 = (n/P^{\log_8(2)})^2 = (n/P^{1/3})^2 = n^2/P^{2/3}$$

for a total that we have seen before (3D algorithm)

$$T_{\mathrm{MM}}^{\alpha,\beta}(n,P) = O(\log(P)\cdot\alpha) + O\left(\frac{n^2}{P^{2/3}}\cdot\beta\right)$$

Memory usage in 2D and 3D algorithms

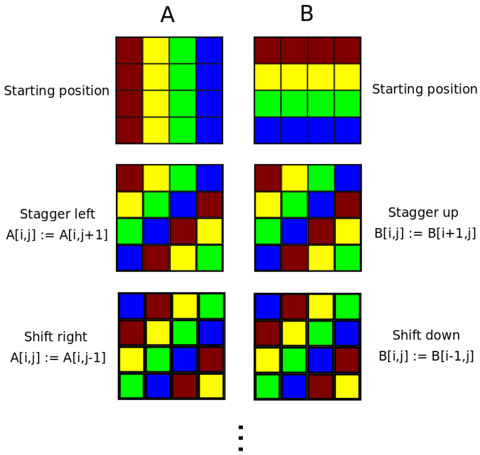
In the SUMMA algorithm, each processor requires at most one block of A, B, and C at each step, with one kept in-place, for a memory usage of

$$M_{\mathrm{SUMMA}} = 5n^2/P$$

In the 3D algorithm, however, each processor receives two blocks of size $n^2/p^{2/3}$ and computes one of the same size, so the memory usage is (to leading order)

$$M_{\mathrm{3D-MM}} = 3n^2/P^{2/3}$$

Cannon's algorithm



[Cannon, 1969]

Cannon's algorithm

Advantages over SUMMA

- uses only near-neighbor sends rather than multicasts
  - **lower latency cost** by factor of $\log(p)$
- can be done in-place given near-neighbor data-swaps

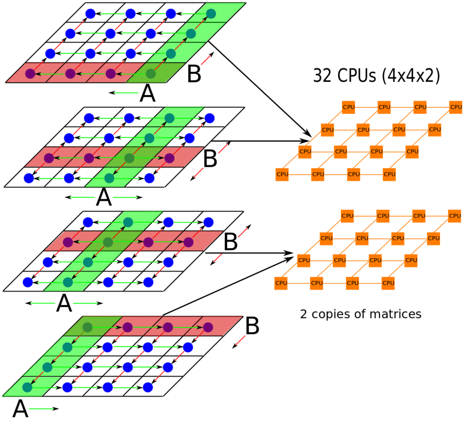
$$M_{\mathrm{Cannon}} = 3n^2/P$$

Disadvantages with respect to SUMMA

- does not generalize well to non-square processor grids
- cannot exploit topology-aware broadcasts

2.5D matrix multiplication

[McColl and Tiskin 99]



$O(n^3/p)$ flops

$O(n^2/\sqrt{c \cdot p})$ words moved

$O(\sqrt{p/c^3} \log p)$ messages

$O(c \cdot n^2/p)$ bytes of memory

2.5D strong scaling

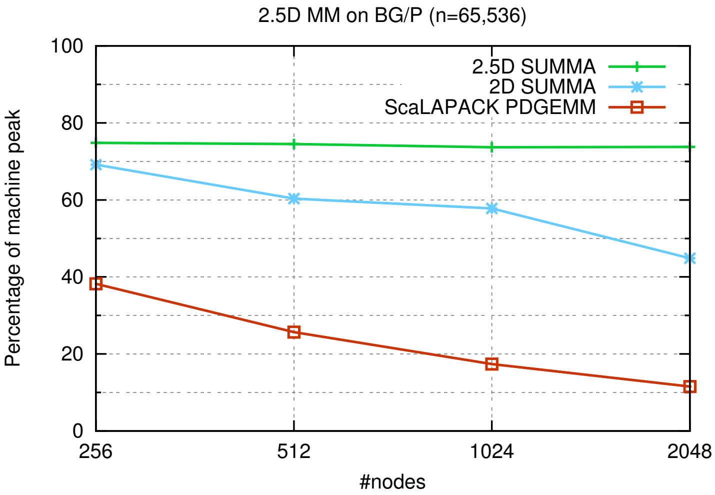
n = dimension, p = #processors, c = #copies of data

- must satisfy $1 \leq c \leq p^{1/3}$
- special case: $c = 1$ yields 2D algorithm
- special case: $c = p^{1/3}$ yields 3D algorithm

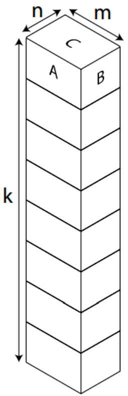
$$\begin{aligned}\text{cost}(\text{2.5D MM}(\mathbf{c} \cdot p, \mathbf{c})) &= O(n^3/(\mathbf{c} \cdot p)) \text{ flops} \\ &\quad + O(n^2/(\mathbf{c} \cdot \sqrt{p})) \text{ words moved} \\ &\quad + O(\sqrt{p} \log p/\mathbf{c}) \text{ messages} \\ &= \text{cost}(\text{2D MM}(p))/\mathbf{c}\end{aligned}$$

can achieve perfect strong scaling

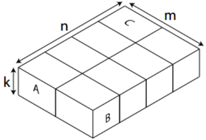
Strong scaling matrix multiplication



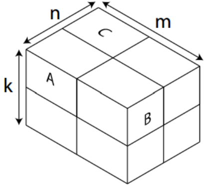
Rectangular Matrix Multiplication



(a) One large dimension



(b) Two large dimensions



(c) Three large dimensions

[Demmel et al, *Communication-optimal parallel recursive rectangular matrix multiplication*, 2013]

Choosing the optimal 3D grid yields bandwidth cost

$$W(m,n,k,p) = O\left(\min_{p_1p_2p_3=p}\left[\frac{mk}{p_1p_2}+\frac{kn}{p_1p_3}+\frac{mn}{p_2p_3}\right]-\frac{mk+mn+kn}{p}\right)$$

Break

короткий перерыв

Homeworks

First homework assignment:

- How is it going? Questions?
- We'll return to segmented scan later in the course, it can be used to formulate many parallel sorting algorithms!
- reminder: please send in pdf form to solomon2@illinois.edu, with email title including "CS 598" by Aug 31, 9:30 AM

Advertisement and enrollment

- please (re)consider enrolling, its not too late!
- homework load should not be overwhelming, will be tuned accordingly
    - each problem is designed to help you understand an important concept
    - policy is flexible, exceptions will be made
- project can be something you are already working on
- if you complete the coursework, you can expect a good grade
- if you are auditing, please complete form `https://registrar.illinois.edu/Media/Default/RGSTRNS/Auditors_Permit.pdf`
- please advertise the course to your colleagues, its not too late to join the course!

Course projects

- the choice of project will be flexible
- doing something in your current research area is encouraged
- setting up a meeting with me prior to first proposal is recommended
    - especially if you are not sure what you want to do
    - can give you feedback on your ideas (gauge difficulty) or suggest others