

CS601: Software Development for Scientific Computing

Autumn 2021

Week14:
Matrix Algebra

Course Progress..

- Last week: FMM, PA4, Matrix Algebra
 - FMM ideas - applying 3-step approximation (decomposition), optimizing (reuse computation), better approximation (multipole expansion), Cost.
 - PA4 discussion
 - Matrix algebra
 - Overview: matrix-matrix multiplication (motivation), program representation of a matrix, storage layout and performance implications.
- This week: Matrix algebra contd.

Matrix Multiplication

- Three fundamental ways to think of the computation

1. Dot product

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1.5 + 2.7 & 1.6 + 2.8 \\ 3.5 + 4.7 & 3.6 + 4.8 \end{bmatrix}$$

2. Linear combination of the columns of the left matrix

$$\begin{bmatrix} \textcolor{blue}{1} & \textcolor{green}{2} \\ \textcolor{blue}{3} & \textcolor{green}{4} \end{bmatrix} \times \begin{bmatrix} \textcolor{red}{5} & \textcolor{blue}{6} \\ \textcolor{red}{7} & \textcolor{red}{8} \end{bmatrix} = \left[\textcolor{red}{5} \begin{bmatrix} \textcolor{blue}{1} \\ \textcolor{blue}{3} \end{bmatrix} + 7 \begin{bmatrix} \textcolor{green}{2} \\ \textcolor{green}{4} \end{bmatrix} \quad \textcolor{blue}{6} \begin{bmatrix} \textcolor{blue}{1} \\ \textcolor{blue}{3} \end{bmatrix} + \textcolor{red}{8} \begin{bmatrix} \textcolor{green}{2} \\ \textcolor{green}{4} \end{bmatrix} \right]$$

3. Sum of outer products

$$\begin{bmatrix} \textcolor{blue}{1} & \textcolor{green}{2} \\ \textcolor{blue}{3} & \textcolor{green}{4} \end{bmatrix} \times \begin{bmatrix} \textcolor{red}{5} & \textcolor{blue}{6} \\ \textcolor{red}{7} & \textcolor{red}{8} \end{bmatrix} = \left[\begin{bmatrix} \textcolor{blue}{1} \\ \textcolor{blue}{3} \end{bmatrix} [\textcolor{red}{5} \quad \textcolor{blue}{6}] + \begin{bmatrix} \textcolor{green}{2} \\ \textcolor{green}{4} \end{bmatrix} [\textcolor{red}{7} \quad \textcolor{red}{8}] \right]$$

Dot Product

- Vector $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, Vector $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ $x_i, y_i \in \mathbb{R}$
- $x^T = [x_1 \quad x_2 \quad \dots \quad x_n]$
- Dot Product or Inner Product: $c = x^T y$ $x^T \in \mathbb{R}^{1 \times n}, y \in \mathbb{R}^{n \times 1}, c$ is scalar

$$[x_1 \quad x_2 \quad \dots \quad x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = [x_1 y_1 + x_2 y_2 + \dots + x_n y_n]$$

- E.g. $[1 \quad 2 \quad 3] \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = [1 \times 4 + 2 \times 5 + 3 \times 6] = 32$

AXPY

- Computing the more common (a times x plus y): $y = y + ax$

- $$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y \\ y_2 \\ \vdots \\ y_n \end{bmatrix} + a \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$


```
..  
for i=1 to n  
    c[i] = c[i] + x[i]*y[i]  
..
```

- Cost? n multiplications and n additions = **2n** or **O(n)**

Matrix Vector Product

- Computing Matrix-Vector product: $c = c + Ax$, $A \in \mathbb{R}^{m \times r}$, $x \in \mathbb{R}^{r \times 1}$

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mr} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} + \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1r}x_r \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2r}x_r \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mr}x_r \end{bmatrix}$$



- Rewriting Matrix-Vector product using dot products:

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mr} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} + \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}$$

- Cost? m rows involving dot products and having the form $c_i = c_i + x^T y$ (Per row cost = $2r$ (because $a_i, x \in \mathbb{R}^r$), Total cost = $2mr$ or $O(mr)$)

Matrix-Matrix Product

- Computing Matrix-Matrix product $C = C + AB$, $A \in \mathbb{R}^{m \times r}$, $B \in \mathbb{R}^{r \times n}$, $C \in \mathbb{R}^{m \times n}$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mr} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r1} & b_{r2} & \cdots & b_{rn} \end{bmatrix}$$

- Consider the AB part first.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mr} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r1} & b_{r2} & \cdots & b_{rn} \end{bmatrix}$$

Matrix-Matrix Product

$$\begin{array}{c} \text{A} \end{array} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1r} \\ a_{21} & a_{22} & \dots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mr} \end{bmatrix} \begin{array}{c} \text{B} \end{array} \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r1} & b_{r2} & \dots & b_{rn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + \dots + a_{1r}b_{r1} & \dots & a_{11}b_{1n} + a_{12}b_{2n} + \dots + a_{1r}b_{rn} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{11} + a_{m2}b_{21} + \dots + a_{mr}b_{r1} & \dots & a_{m1}b_{1n} + a_{m2}b_{2n} + \dots + a_{mr}b_{rn} \end{bmatrix}$$

$$= \begin{bmatrix} a_1^T b_1 & \dots & a_1^T b_n \\ \vdots & \ddots & \vdots \\ a_m^T b_1 & \dots & a_m^T b_n \end{bmatrix}$$

$a_i^T \in \mathbb{R}^{1 \times r}, b_j \in \mathbb{R}^{r \times 1}$
 i ranges from 1 to m
 j ranges from 1 to n

Matrix-Matrix Product using Dot Product Formulation

- Pseudocode - Matrix-Matrix product: $C = C + AB$, $A \in \mathbb{R}^{m \times r}$, $B \in \mathbb{R}^{r \times n}$, $C \in \mathbb{R}^{m \times n}$

```

..
for i=1 to m
  for j=1 to n
    //compute updates involving dot products
     $c_{ij} = c_{ij} + a_i^T b_j$ 

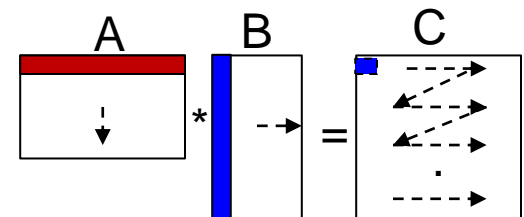
```

- Expanded: ..


```

      for i=1 to m
        for j=1 to n
          for k=1 to r

```



$$c_{ij} = c_{ij} + a_{ik} b_{kj}$$

Elements of C matrix are computed from top to bottom, left to right. Per element computation, you need a row of A and a column of B.

Matrix-Matrix Product using Dot Product Formulation

- Pseudocode - Matrix-Matrix product: $C = C + AB$, $A \in \mathbb{R}^{m \times r}$, $B \in \mathbb{R}^{r \times n}$, $C \in \mathbb{R}^{m \times n}$
..
for i=1 to m
 for j=1 to n
 //compute updates involving dot products
 $c_{ij} = c_{ij} + a_i^T b_j$
- Cost?
 - Per dot-product cost = $2r$ ($a_i, b_j \in \mathbb{R}^r$) Total cost = $2mnr$ or $O(mnr)$

Common Computational Patterns

Some patterns that we see while doing Matrix-Matrix product:

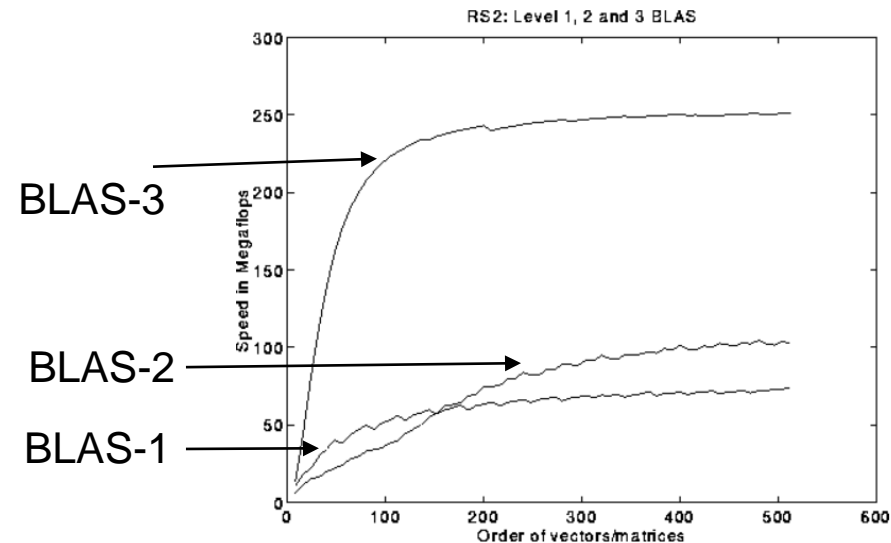
- Dot Product or Inner Product: $x^T y$ ← Slide 4, Method 1
- Scalar **a** times **x** plus **y**: $y = y + ax$ OR $axpy$ ← Slide 4, Method 2
- Scalar times **x**: αx
- **Matrix** times **x** plus **y**: $y = y + Ax$ ← Slide 4, Method 1
 - generalized $axpy$ OR $gaxpy$
- Outer product: $C = C + xy^T$ ← Slide 4, Method 3
- **Matrix** times **Matrix** plus **Matrix**
 - GEMM or generalized matrix multiplication

BLAS – Basic Linear Algebra Subroutines

- Level-1 or BLAS-1 (46 operations, routines operating on vectors mostly)
 - axpy, dot product, rotation, scale, etc.
 - 4 versions each: **Single-precision**, **double-precision**, **complex**, **complex-double (z)**
 - E.g. saxpy, daxpy, caxpy etc.
 - **Do $O(n)$ operations on $O(n)$ data.**
- Level-2 or BLAS-2 (25 operations, routines operating on matrix-vectors mostly)
 - E.g. GEMV ($\alpha A \cdot x + \beta y$), GER (Rank-1 update $A = A + y \cdot x^T$), Triangular solve ($y = T \cdot x, T$ is a triangular matrix) etc.
 - 4 versions each, **do $O(n^2)$ operations on $O(n^2)$ data.**

BLAS – Basic Linear Algebra Subroutines

- Level-3 or BLAS-3 (9 basic operations, routines operating on matrix-matrix mostly)
 - GEMM ($C = \alpha A \cdot B + \beta C$),
 - Multiple triangular solve ($Y = TX$, T is triangular, X is rectangular)
 - **Do $O(n^3)$ operations on $O(n^2)$ data.**
- *Why categorize as BLAS-1, BLAS-2, BLAS-3?*
 - *Performance*



source: <http://people.eecs.berkeley.edu/~demmel/cs267/lecture02.html>

Computational Intensity

- Average number of operations performed per data element (word) read/written from slow memory
 - E.g. Read/written m words from memory. Perform f operations on m words.
 - Computational Intensity $q = f/m$ (*flops per word*).
- We want to *maximize* the computational intensity
- What is q for axpy? Matrix-vector product? Matrix-Matrix product?

Computational Intensity - axpy

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} + [x_1 \quad x_2 \quad \cdots \quad x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} + \begin{bmatrix} x_1 \times y_1 \\ x_2 \times y_2 \\ \vdots \\ x_n \times y_n \end{bmatrix}$$

Read(x) //read x from slow memory

Read(y) //read y from slow memory

Read(c) //read c from slow memory

for i=1 to n

 c[i] = c[i] + x[i]*y[i] //do arithmetic on data read

Write(c) //write c back to slow memory

- Number of memory operations = $4n$ (assuming one word of storage for each component (x_i, y_i, c_i) of vectors x, y, c resp.)
- Number of arithmetic operations = $2n$ (one addition and one multiplication per row.)
- **$q=2n/4n = 1/2$**

Computational Intensity – matrix-vector

- Assume $m=r=n =n$

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ & \vdots & & \\ a_{m1} & a_{m2} & \cdots & a_{mr} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} + \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1r}x_r \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2r}x_r \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mr}x_r \end{bmatrix}$$

- Number of memory operations = $n^2 + 3n = n^2 + O(n)$
- Number of arithmetic operations = $2n^2$
- $q \approx 2n^2/n^2 = 2$

Computational Intensity – matrix-matrix

```
for i=1 to n
  //Read row i of A into fast memory
  for j=1 to n
    //Read C(i,j) into fast memory
    //Read column j of B into fast memory
    for k=1 to n
      C(i,j)=C(i,j) + A(i,k)*B(k,j)
    //Write C(i,j) back to slow memory
```

n^2 words read: each row of A read once for each i. Assume that the row read stays in fast memory during the execution of inner two loops.

n^3 words read: each column of B read n^2 times

$2n^2$ words read: read/write each entry of C to memory once.

- Number of memory operations = $n^3 + 3n^2 = n^3 + O(n^2)$
- Number of arithmetic operations = $2n^3$
- $q \approx 2n^3/n^3 = 2$. Same as matrix-vector?
- What if the fast memory has space to hold entire B matrix, a row of A matrix, and one element of C matrix?

Blocked Matrix Multiply

- For N=4:

$$\begin{bmatrix} C1 & C2 & C3 & C4 \end{bmatrix} = \begin{bmatrix} C1 & C2 & C3 & C4 \end{bmatrix} + \begin{bmatrix} A \end{bmatrix} * \begin{bmatrix} B1 & B2 & B3 & B4 \end{bmatrix}$$

$$\begin{bmatrix} C_j \end{bmatrix} = \begin{bmatrix} C_j \end{bmatrix} + \begin{bmatrix} A \end{bmatrix} * \begin{bmatrix} B_j \end{bmatrix} = \begin{bmatrix} C_j \end{bmatrix} + \sum_{k=1}^n \begin{bmatrix} A(:,k) \end{bmatrix} * \begin{bmatrix} B_j(k,:) \end{bmatrix}$$

```

for j=1 to N
//Read entire Bj into fast memory
//Read entire Cj into fast memory
  for k=1 to n
    //Read column k of A into fast memory
    Cj=Cj + A(*,k) * Bj(k,*)
  //Write Cj back to slow memory

```

Blocked Matrix Multiply - Example

$$\begin{array}{c} C_1 \quad C_2 \quad C_3 \quad C_4 \\ \left[\begin{array}{c|c|c|c} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{array} \right] = \begin{array}{c} C_1 \quad C_2 \quad C_3 \quad C_4 \\ \left[\begin{array}{c|c|c|c} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{array} \right] + \begin{array}{c} A \\ \left[\begin{array}{c|c|c|c} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right] \end{array} \begin{array}{c} B_1 \quad B_2 \quad B_3 \quad B_4 \\ \left[\begin{array}{c|c|c|c} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array} \right] \end{array}
 \end{array}$$

for k=1 to n

$$\begin{array}{c} j=1 \\ \left[\begin{array}{c} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{array} \right] = \left[\begin{array}{c} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{array} \right] + \left[\begin{array}{c|c|c|c} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right] * \left[\begin{array}{c} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{array} \right]
 \end{array}$$

.....

for k=1 to n

$$\begin{array}{c} j=4 \\ \left[\begin{array}{c} c_{14} \\ c_{24} \\ c_{34} \\ c_{44} \end{array} \right] = \left[\begin{array}{c} c_{14} \\ c_{24} \\ c_{34} \\ c_{44} \end{array} \right] + \left[\begin{array}{c|c|c|c} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right] * \left[\begin{array}{c} b_{14} \\ b_{24} \\ b_{34} \\ b_{44} \end{array} \right]
 \end{array}$$

Blocked Matrix Multiply - Example

$$\begin{array}{c|c|c|c} C_1 & C_2 & C_3 & C_4 \\ \hline \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} & c_{12} & c_{13} & c_{14} \\ \hline c_{22} & c_{23} & c_{24} & \\ \hline c_{32} & c_{33} & c_{34} & \\ \hline c_{42} & c_{43} & c_{44} & \end{array} = \begin{array}{c|c|c|c} C_1 & C_2 & C_3 & C_4 \\ \hline \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} & c_{12} & c_{13} & c_{14} \\ \hline c_{22} & c_{23} & c_{24} & \\ \hline c_{32} & c_{33} & c_{34} & \\ \hline c_{42} & c_{43} & c_{44} & \end{array} + \begin{array}{c|c|c|c} \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{bmatrix} & a_{12} & a_{13} & a_{14} \\ \hline a_{22} & a_{23} & a_{24} & \\ \hline a_{32} & a_{33} & a_{34} & \\ \hline a_{42} & a_{43} & a_{44} & \end{array} A \begin{array}{c|c|c|c} B_1 & B_2 & B_3 & B_4 \\ \hline \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix} & b_{12} & b_{13} & b_{14} \\ \hline b_{22} & b_{23} & b_{24} & \\ \hline b_{32} & b_{33} & b_{34} & \\ \hline b_{42} & b_{43} & b_{44} & \end{array}$$

for k=1 to n



j=1

$$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{bmatrix} \begin{bmatrix} a_{12} & a_{13} & a_{14} \\ a_{22} & a_{23} & a_{24} \\ a_{32} & a_{33} & a_{34} \\ a_{42} & a_{43} & a_{44} \end{bmatrix} * \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix}$$

k=1

$$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{bmatrix} * [b_{11}] \quad \leftarrow \text{First row of } B_1$$

$$= \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + \begin{bmatrix} a_{11}b_{11} \\ a_{21}b_{11} \\ a_{31}b_{11} \\ a_{41}b_{11} \end{bmatrix}$$

-  What is required to be in fast memory
-  What is operated upon

Blocked Matrix Multiply - Example

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

for k=1 to n

j=1

$$\begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \\ C_{41} \end{bmatrix} = \begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \\ C_{41} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix}$$

k=2

$$\begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \\ C_{41} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} \\ a_{21}b_{11} \\ a_{31}b_{11} \\ a_{41}b_{11} \end{bmatrix} + \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ a_{42} \end{bmatrix} * [b_{21}]$$

Second row of B_1

Comes from partial sum for C_1 computed for k=1 (previous slide)

$$\begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \\ C_{41} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} \\ a_{21}b_{11} \\ a_{31}b_{11} \\ a_{41}b_{11} \end{bmatrix} + \begin{bmatrix} a_{12}b_{21} \\ a_{22}b_{21} \\ a_{32}b_{21} \\ a_{42}b_{21} \end{bmatrix}$$

Blocked Matrix Multiply - Example

$$\begin{array}{c|c|c|c} C_1 & C_2 & C_3 & C_4 \\ \hline \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} & c_{12} & c_{13} & c_{14} \\ \hline c_{22} & c_{23} & c_{24} & \\ \hline c_{32} & c_{33} & c_{34} & \\ \hline c_{42} & c_{43} & c_{44} & \end{array} = \begin{array}{c|c|c|c} C_1 & C_2 & C_3 & C_4 \\ \hline \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} & c_{12} & c_{13} & c_{14} \\ \hline c_{22} & c_{23} & c_{24} & \\ \hline c_{32} & c_{33} & c_{34} & \\ \hline c_{42} & c_{43} & c_{44} & \end{array} + \begin{array}{c|c|c|c} A & & & \\ \hline a_{11} & a_{12} & \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \\ a_{43} \end{bmatrix} & a_{14} \\ \hline a_{21} & a_{22} & & a_{24} \\ \hline a_{31} & a_{32} & & a_{34} \\ \hline a_{41} & a_{42} & & a_{44} \end{array} + \begin{array}{c|c|c|c} B_1 & B_2 & B_3 & B_4 \\ \hline \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix} & b_{12} & b_{13} & b_{14} \\ \hline b_{22} & b_{23} & b_{24} & \\ \hline b_{32} & b_{33} & b_{34} & \\ \hline b_{42} & b_{43} & b_{44} & \end{array}$$

for k=1 to n

j=1

$$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \\ a_{43} \end{bmatrix} & a_{14} \\ a_{21} & a_{22} & & a_{24} \\ a_{31} & a_{32} & & a_{34} \\ a_{41} & a_{42} & & a_{44} \end{bmatrix} * \begin{bmatrix} b_{11} \\ b_{21} \\ \begin{bmatrix} b_{31} \\ b_{41} \end{bmatrix} \end{bmatrix}$$

k=3

$$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} \\ a_{21}b_{11} + a_{22}b_{21} \\ a_{31}b_{11} + a_{32}b_{21} \\ a_{41}b_{11} + a_{42}b_{21} \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \\ a_{43} \end{bmatrix} * [b_{31}]$$

← Third row of B_1

$$= \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} \\ a_{21}b_{11} + a_{22}b_{21} \\ a_{31}b_{11} + a_{32}b_{21} \\ a_{41}b_{11} + a_{42}b_{21} \end{bmatrix} + \begin{bmatrix} a_{13}b_{31} \\ a_{23}b_{31} \\ a_{33}b_{31} \\ a_{43}b_{31} \end{bmatrix}$$

Blocked Matrix Multiply - Example

$$\begin{array}{c|c|c|c} C_1 & C_2 & C_3 & C_4 \\ \hline \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} & \begin{bmatrix} c_{12} \\ c_{22} \\ c_{32} \\ c_{42} \end{bmatrix} & \begin{bmatrix} c_{13} \\ c_{23} \\ c_{33} \\ c_{43} \end{bmatrix} & \begin{bmatrix} c_{14} \\ c_{24} \\ c_{34} \\ c_{44} \end{bmatrix} \\ \hline \end{array} = \begin{array}{c|c|c|c} C_1 & C_2 & C_3 & C_4 \\ \hline \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} & \begin{bmatrix} c_{12} \\ c_{22} \\ c_{32} \\ c_{42} \end{bmatrix} & \begin{bmatrix} c_{13} \\ c_{23} \\ c_{33} \\ c_{43} \end{bmatrix} & \begin{bmatrix} c_{14} \\ c_{24} \\ c_{34} \\ c_{44} \end{bmatrix} \\ \hline \end{array} + \begin{array}{c|c|c|c} A & & & \\ \hline \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} & & & \\ \hline \end{array} \begin{array}{c|c|c|c} B_1 & B_2 & B_3 & B_4 \\ \hline \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix} & \begin{bmatrix} b_{12} \\ b_{22} \\ b_{32} \\ b_{42} \end{bmatrix} & \begin{bmatrix} b_{13} \\ b_{23} \\ b_{33} \\ b_{43} \end{bmatrix} & \begin{bmatrix} b_{14} \\ b_{24} \\ b_{34} \\ b_{44} \end{bmatrix} \\ \hline \end{array}$$

for k=1 to n

j=1

$$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} * \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix}$$

Fourth row of B_1

k=4

$$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} \\ a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} \end{bmatrix} + \begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \\ a_{44} \end{bmatrix} * [b_{41}]$$

$$= \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} \\ a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} \end{bmatrix} + \begin{bmatrix} a_{14}b_{41} \\ a_{24}b_{41} \\ a_{34}b_{41} \\ a_{44}b_{41} \end{bmatrix}$$

Blocked Matrix Multiply - Example

$$\begin{bmatrix} C_1 & C_2 & C_3 & C_4 \\ \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 \\ \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \\ \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} B_1 & B_2 & B_3 & B_4 \\ \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \\ \end{bmatrix}$$

for k=1 to n

$$\begin{matrix} j=2 \\ \begin{bmatrix} c_{12} \\ c_{22} \\ c_{32} \\ c_{42} \end{bmatrix} = \begin{bmatrix} c_{12} \\ c_{22} \\ c_{32} \\ c_{42} \end{bmatrix} + \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{bmatrix} \begin{bmatrix} a_{12} & a_{13} & a_{14} \\ a_{22} & a_{23} & a_{24} \\ a_{32} & a_{33} & a_{34} \\ a_{42} & a_{43} & a_{44} \end{bmatrix} * \begin{bmatrix} b_{12} \\ b_{22} \\ b_{32} \\ b_{42} \end{bmatrix} \end{matrix}$$

- And so on..
- At any point, you need C_j , B_j , and one column of A to be in fast memory

Computational Intensity - Blocked Matrix Multiply

```
for j=1 to N
  //Read entire Bj into fast memory →  $n^2$  words read: each column of B read once.
  //Read entire Cj into fast memory
  for k=1 to n
    //Read column k of A into fast memory →  $Nn^2$  words read: each column of A read N times
    C(*,j)=C(*,j) + A(*,k)*Bj(k,*) //outer-product
    //Write Cj back to slow memory →  $2n^2$  words read: read/write each entry of C to memory once.
```

- Number of arithmetic operations = $2n^3$
- $q = 2n^3 / (N + 3)n^2 = 2n/N$. **Good!**

Blocked Matrix Multiply - General

$$\begin{array}{ccc}
 C & A & B \\
 \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1r} \\ C_{21} & C_{22} & \dots & C_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ C_{q1} & C_{q2} & \dots & C_{qr} \end{bmatrix} & \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1p} \\ A_{21} & A_{22} & \dots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{q1} & A_{q2} & \dots & A_{qp} \end{bmatrix} & \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1r} \\ B_{21} & B_{22} & \dots & B_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ B_{p1} & B_{p2} & \dots & B_{pr} \end{bmatrix} \\
 \begin{array}{c} \downarrow \quad \rightarrow \\ q \quad r \end{array} & \begin{array}{c} \downarrow \quad \rightarrow \\ q \quad p \end{array} & \begin{array}{c} \downarrow \quad \rightarrow \\ p \quad r \end{array}
 \end{array}$$

- $A, B, C \in \mathbb{R}^{n \times n}$
- We wish to update C block-by-block: $C_{ij} = C_{ij} + \sum_{k=1}^p A_{ik} B_{kj}$
 - Assume that blocks of A , B , and C fit in cache. C_{ij} is roughly n/q by n/r , A_{ij} is roughly n/q by n/p , B_{ij} is roughly n/p by n/r .
 - But how to choose block parameters p, q, r such that assumption holds for a cache of size M ?
 - i.e. given the constraint that $\frac{n}{q} \times \frac{n}{r} + \frac{n}{q} \times \frac{n}{p} + \frac{n}{p} \times \frac{n}{r} \leq M$

Blocked Matrix Multiply - General

- Maximize $\frac{2n^3}{qrp}$ subject to $\frac{n}{q} \times \frac{n}{r} + \frac{n}{q} \times \frac{n}{p} + \frac{n}{p} \times \frac{n}{r} \leq M$
 - $q_{opt} = p_{opt} = r_{opt} \approx \sqrt{\frac{n^2}{3M}}$
- Each block should roughly be a square matrix and occupy one third of the cache size
- Can we design algorithms that are independent of cache size?

Recursive Matrix Multiply

- Cache-oblivious algorithm
 - No matter what the size of the cache is, the algorithm performs at a near-optimal level
- Divide-conquer approach

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

- Apply the formula recursively to $A_{11}B_{11}$ etc.
 - Works neat when n is a power of 2.
- What layout format is preferred for this algorithm?
 - Row-major or Col-major? Neither.

Recursive Matrix Multiply

- Cache-oblivious Data structure

$$\begin{bmatrix} 1 & 2 & 5 & 6 & 17 & 18 & 21 & 22 \\ 3 & 4 & 7 & 8 & 19 & 20 & 23 & 24 \\ 9 & 10 & 13 & 14 & 25 & 26 & 29 & 30 \\ 11 & 12 & 15 & 16 & 27 & 28 & 31 & 32 \\ 33 & 34 & 37 & 38 & 49 & 50 & 53 & 54 \\ 35 & 36 & 39 & 40 & 51 & 52 & 55 & 56 \\ 41 & 42 & 45 & 46 & 57 & 58 & 61 & 62 \\ 43 & 44 & 47 & 48 & 59 & 60 & 63 & 64 \end{bmatrix}.$$

- Matrix entries are stored in the order shown
 - E.g. row-major would have 1-8 in the first row, followed by 9-16 in the second and so on.

Efficiency Considerations

- Cache details (size)
- Data movement overhead
- Storage layout
- Parallel functional Units (Vector units)

Data Movement Overhead - Example

- gaxpy ($y = y + Ax$) vs. Outer product ($A = A + yx^T$)
- What is the data movement overhead? *assume a vector of dimension n can be read with one memory read*

gaxpy

```
// Read y into fast memory
// Read x into fast memory
for i=1 to n
    //Read column  $c_i$  of A into fast memory
    for j=1 to n
         $y[j] = y[j] + c_i x[j]$ 
//Write y into slow memory
```

$$\begin{array}{|c|} \hline y \\ \hline \end{array} = \begin{array}{|c|} \hline y \\ \hline \end{array} + \begin{array}{|c|} \hline c_i \\ \hline \end{array} \cdot \begin{array}{|c|} \hline x[j] \\ \hline \end{array}$$

Outer product

```
// Read y into fast memory
// Read x into fast memory
for j=1 to n
    //Read a column of A into fast memory
    for i=1 to n
         $A[i,j] = A[i,j] + y[i] x[j]$ 
//Write  $A[:,j]$  into slow memory
```

$$\begin{array}{|c|} \hline A[j] \\ \hline \end{array} = \begin{array}{|c|} \hline A[j] \\ \hline \end{array} + \begin{array}{|c|} \hline y[i] \\ \hline \end{array} \cdot \begin{array}{|c|} \hline x[j] \\ \hline \end{array}$$

Storage Layout Considerations

- Assume column-order storage for A, B, and C. Which implementation scheme for matmul is better? Why?

ijk

vs.

jki

```
for i=1 to m
  for j=1 to n
    for k=1 to r
      c[i][j]=c[i][j]+
        a[i][k]*b[k][j]
```

```
for j=1 to n
  for k=1 to r
    for i=1 to m
      c[i][j]=c[i][j]+
        a[i][k]*b[k][j]
```


Unblocked Matrix Multiplication - Loop Orderings and Properties

Loop Order	Inner Loop	Inner Two Loops	Inner Loop Data Access
i j k	dot	Vector x Matrix	A by row, B by column
j k i	saxpy	gaxpy	A by column, C by column
k j i	saxpy	Outer product	A by column, C by column

Ref: Matrix Computations, 4th Ed., Golub and Van Loan

Parallel Functional Units

- IBM's RS/6000 and Fused Multiply Add (FMA)
 - Fuses multiply and an add into one functional unit ($c=c+a*b$)
 - The functional unit consists of 3 independent subunits
 - Pipelining
 - Example:

```
sum=0.0
for (i=0;i<n;i++)
    sum=sum+a[i]*b[i]
```
 - Suppose the FMA unit takes 3 cycles to complete, how many cycles do you need to execute the above code snippet?
 - With loop unrolled 4 times? Assume n is divisible by 4.

Matrix Structure and Efficiency

- Sparse Matrices
 - E.g. banded matrices
 - Diagonal
 - Tridiagonal etc.
- Symmetric Matrices

Admit optimizations w.r.t.

- Storage
- Computation