CS601: Software Development for Scientific Computing

Autumn 2022

Week5: Motifs – Matrix Computations with Dense Matrices

Last week..

- Demo of make program
- Motif – Matrix Computation with Dense Matrices
  - Matrix Representation (2D arrays on stack and heap)
  - Matrix storage format (row-major and column-major)
  - Visualizing performance gap with different layouts (demo)
  - Understanding the performance gap:
    - Memory hierarchy
    - Performance API (demo)

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

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}5\begin{bmatrix}1\\3\end{bmatrix} + 7\begin{bmatrix}2\\4\end{bmatrix} & 6\begin{bmatrix}1\\3\end{bmatrix} + 8\begin{bmatrix}2\\4\end{bmatrix}\end{bmatrix}$$

  3. Sum of outer products

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}1\\3\end{bmatrix}\begin{bmatrix}5 & 6\end{bmatrix} + \begin{bmatrix}2\\4\end{bmatrix}\begin{bmatrix}7 & 8\end{bmatrix}\end{bmatrix}$$

Dot Product

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

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

- 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
   y[i] = y[i] +  a*x[i]
..
```

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

Matrix Vector Product

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

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

- Rewriting Matrix-Vector product using dot products:

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

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

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

- Consider the AB part first.

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

Matrix-Matrix Product

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

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

Notice that:

- subscript on a varies from 1 to m in a column (i.e. m rows exist)
- subscript on a varies from 1 to r in a row (i.e. r columns exist)

Suppose that we treat a_i as a vector of size r and there exist m vectors

$$= \begin{bmatrix} a_1^T b_1 & . & . & a_1^T b_n \\ . & . & & \\ a_m^T b_1 & . & . & a_m^T b_n \end{bmatrix}$$

$a_i^T \in \mathbb{R}^{1\times r}, b_j \in \mathbb{R}^{r\times 1}$

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
```

$$c_{ij} = c_{ij} + a_i^T b_j$$

Matrix-Matrix Product using Dot Product Formulation – Data Access

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

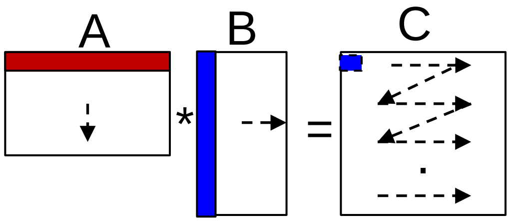
```
..
for i=1 to m
    for j=1 to n
        //compute updates involving dot products
```

$$c_{ij} = c_{ij} + a_i^T b_j$$

- 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 - Cost

- 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:

1. Dot Product or Inner Product: $x^T y$ ← Slide 27, Method 1
2. Scalar **a** times **x** **p**lus **y**: y=y+ax OR saxpy ← Slide 27, Method 2
   - Scalar times **x**: αx
3. Matrix times x plus y: y=y+Ax ← Slide 27, Method 1
   - generalized axpy OR gaxpy
4. Outer product: $C=C+xy^T$ ← Slide 27, Method 3
5. Matrix times Matrix plus Matrix
   - `GEMM` or generalized matrix multiplication

What is dense linear algebra?

- Not just matrix multiplication (matmul!)
- Solving system of equations: Ax=b (e.g. using Gaussian Elimination)
- Computing Least Squares: choose x to minimize $||Ax-b||_2$
  - Overdetermined or underdetermined; Unconstrained, constrained, or weighted
- Computing Eigenvalues and Eigenvectors of Matrices (Symmetric and Unsymmetric)
    - Standard ($Ax = \lambda x$), Generalized ($Ax=\lambda Bx$)
- Representing Different matrix structures
  - Real, complex; Symmetric, Hermitian, positive definite; dense, triangular, banded …
- Capturing level of detail
  - error bounds, extra-precision, other options

slide source: www.cs.berkeley.edu/~demmel (CS267)

Linear Algebra Software

- **Goals**: programmer productivity, readability, robustness, portability, machine efficiency
- Examples
  - EISPACK (for computing eigenvalue problems)
  - BLAS
  - LAPACK
  - Many more..

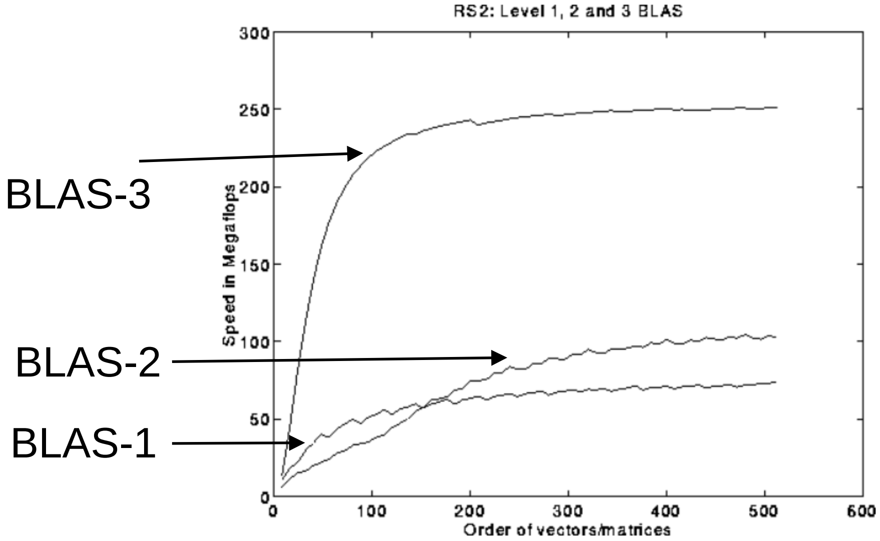
BLAS – Basic Linear Algebra Subroutines

- Level-1 or BLAS-1 (46 operations, routines operating on vectors mostly)
  - axpy, dot product, rotation, scale, etc.
  - 4 versions each: **S**ingle-precision, **d**ouble-precision, **c**omplex, 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 ($\alpha A.x + \beta y$), GER (Rank-1 update $A = A + y.x^T$), Triangular solve ($y = T.x, T\ is\ a\ triangular\ matrix$) 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 ($C = \alpha A . B + \beta C$),
  - Multiple triangular solve ($Y = TX$, T is triangular, X is rectangular)
  - **Do O(n³) operations on O(n²) data.**

- *Why categorize as BLAS-1, BLAS-2, BLAS-3?*
  - *Performance*



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

LAPACK – Linear Algebra Package

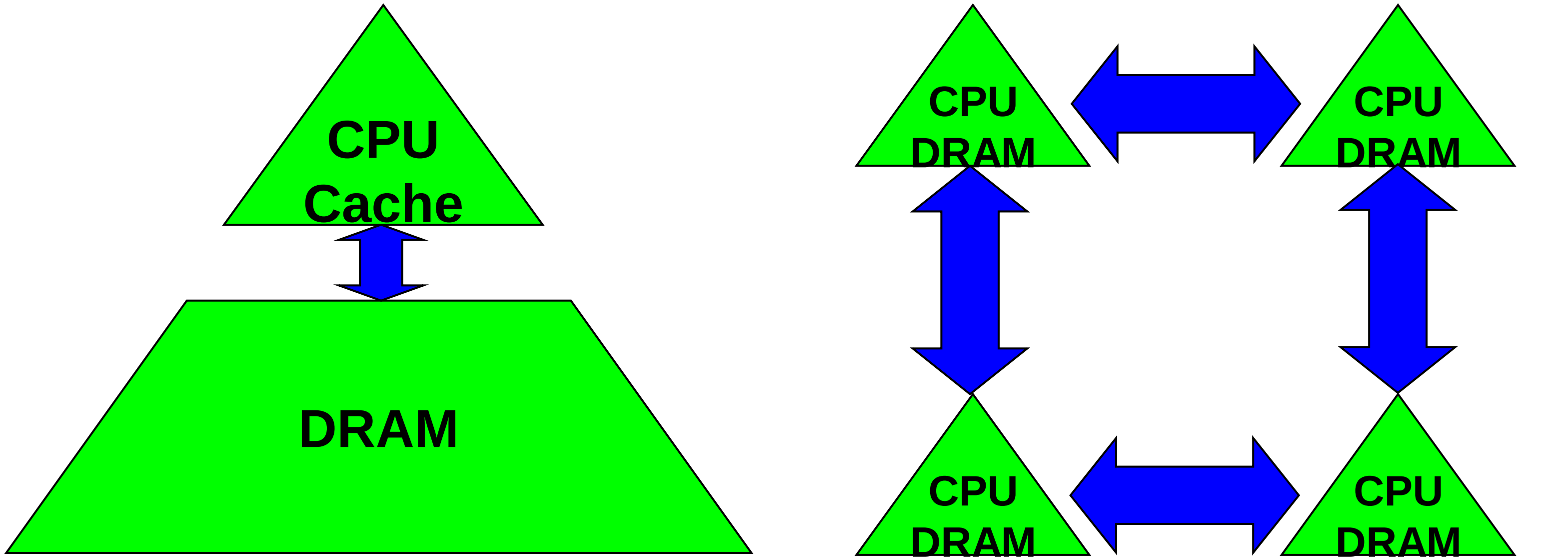
- LAPACK – uses BLAS-3 (1989 – now)
  - Ex: Obvious way to express Gaussian Elimination (GE) is adding multiples of one row to other rows – BLAS-1
    - How do we reorganize GE to use BLAS-3 ?
  - Contents of LAPACK (summary)
    - Algorithms that are (nearly) 100% BLAS-3
      - Linear Systems, Least Squares
    - Algorithms that are only ≈50% BLAS-3
      - Eigenproblems, Singular Value Decomposition (SVD)
    - Generalized problems (eg Ax = l Bx)
    - Error bounds for everything
    - Lots of variants depending on A's structure (banded, $A=A^T$, etc.)
  - How much code? (Release 3.9.0, Nov 2019) (www.netlib.org/lapack)
    - Source: 1982 routines, 827K LOC, Testing: 1210 routines, 545K LOC

slide source: www.cs.berkeley.edu/~demmel (CS267)

Costs Involved

Algorithms have two costs:

1. Arithmetic (FLOPS)
2. Communication: moving data between
   - levels of a memory hierarchy (sequential case)
   - processors over a network (parallel case).



Computational Intensity

- Connection between computation and communication cost
- 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*).
- Goal: we want to *maximize* the computational intensity
  - We want to minimize words moved (read/written)
  - We want to minimize messages sent

What is the computational intensity, q, for:
axpy?
***M**atrix-**V**ector product? (e.g. GEMV)*
***M**atrix-**M**atrix product? (e.g. GEMM)*

Computational Intensity - axpy

Note: a slightly changed variant of axpy. There are n scalars (x_i) here.

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

. indicates component-wise multiplication*

```
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

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

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

Communication Cost – Matrix-Matrix Product

```
//Assume A, B, C are all nxn
for i=1 to n
 for j=1 to n
   for k=1 to n
     C(i,j)=C(i,j) + A(i,k)*B(k,j)
```

- n^2 words read: each row of A read once for each i.
- Assume that row i of A stays in fast memory during j=2, .. J=n
- Reading a row i of A

- loop k=1 to n: read C(i,j) into fast memory and update in fast memory
- End of loop k=1 to n: write C(i,j) back to slow memory

n^2 words read and n^2 words written (each entry of C read/written to memory once). = 2 n^2 words read/written

total cost = 3 n^2 +n^3 (if the cache size is n+n+1)

- Reading column j of B
- Suppose there is space in fast memory to hold only one column of B (in addition to one row of A and 1 element of C), then every column of B is read from slow memory to fast memory once in **inner two loops.**
- Each column of B read n times including **outer i loop =** n^3 words read

Computational Intensity – Matrix-Matrix Product

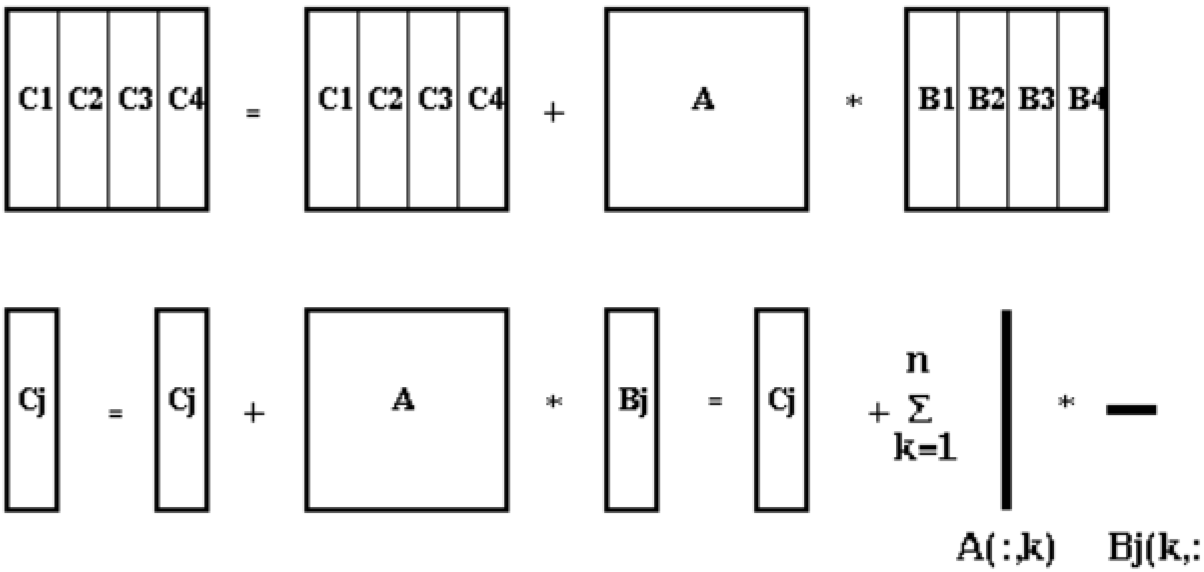
- Words moved = $n^3+3n^2 = n^3+O(n^2)$
- Number of arithmetic operations = $2n^3$ (from slide 35)
- computational intensity $q \approx 2n^3/n^3 = 2$. (computation to communication ratio)

  Same as q for matrix-vector?
  What if the fast memory has more space ? more than just two columns + one element space?

- Can we do better?

Blocked Matrix Multiply

- For N=4:




```
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
```


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

Blocked Matrix Multiply - Example

$$\begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 & C_4 \end{matrix} \\ \left[\begin{array}{c|c|c|c} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{array}\right] \end{array} = \begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 & C_4 \end{matrix} \\ \left[\begin{array}{c|c|c|c} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{array}\right] \end{array} + \begin{array}{c} \text{A} \\ \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} B_1 & B_2 & B_3 & B_4 \end{matrix} \\ \left[\begin{array}{c|c|c|c} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array}\right] \end{array}$$

```
for k=1 to n
```

j=1

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

....

```
for k=1 to n
```

j=4

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

Blocked Matrix Multiply - Example

$$\begin{array}{c}\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \end{array}\\ \begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}\end{array} = \begin{array}{c}\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \end{array}\\ \begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}\end{array} + \begin{array}{c}\text{A}\\ \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}\end{array} \begin{array}{c}\begin{array}{cccc} B_1 & B_2 & B_3 & B_4 \end{array}\\ \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}\end{array}$$

```
for k=1 to n
```

j=1 $$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} * \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix}$$

k=1 $$\begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{bmatrix} * [b_{11}]$$

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

$$\begin{array}{c} \begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \end{array} \\ \begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \end{array} = \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \end{array} \\ \begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \end{array} + \begin{array}{c} \text{A} \\ \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{cccc} B_1 & B_2 & B_3 & B_4 \end{array} \\ \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \end{array}$$

`for k=1 to n`

j=1

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

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

$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \end{array} \quad\quad \begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \end{array} \quad\quad \text{A} \quad\quad \begin{array}{cccc} B_1 & B_2 & B_3 & B_4 \\ \end{array}$$

$$\begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

```
for k=1 to n
```

j=1

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

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

$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \end{array} \qquad \begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \end{array} \qquad \text{A} \qquad \begin{array}{cccc} B_1 & B_2 & B_3 & B_4 \end{array}$$

$$\begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

`for k=1 to n`

j=1

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

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

$$
\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \end{array}
\qquad\qquad
\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \end{array}
\qquad\qquad\qquad \text{A} \qquad\qquad\qquad
\begin{array}{cccc} B_1 & B_2 & B_3 & B_4 \\ \end{array}
$$

$$
\left[\begin{array}{c|c|c|c} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{array}\right] = \left[\begin{array}{c|c|c|c} c_{11} & c_{12} & c_{!3} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{array}\right] + \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \left[\begin{array}{c|c|c|c} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array}\right]
$$

`for k=1 to n`

j=2

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

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

n^2 words read: each column of B read once.

Nn^2 words read: each column of A read N times

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

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

Blocked Matrix Multiply - General

$$C = \begin{bmatrix} C_{11} & C_{12} & .. & C_{1r} \\ C_{21} & C_{22} & .. & C_{2r} \\ & \vdots & & \\ C_{q1} & C_{q2} & .. & C_{qr} \end{bmatrix} \quad A = \begin{bmatrix} A_{11} & A_{12} & .. & A_{1p} \\ A_{21} & A_{22} & .. & A_{2p} \\ & \vdots & & \\ A_{q1} & A_{q2} & .. & A_{qp} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} & .. & B_{1r} \\ B_{21} & B_{22} & .. & B_{2r} \\ & \vdots & & \\ B_{p1} & B_{p2} & .. & B_{pr} \end{bmatrix}$$

r q   p q   r p

- $A, B, C \in \mathbb{R}^{n \times n}$
- We wish to update C block-by-block: $C_{ij} = C_{ij} + \Sigma_{k=1}^{p} A_{ik}B_{kj}$
  - 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?