Welcome to Ma221!

Goal: understand real cost in time of running an alg.
Traditionally count flops but this is cheapest of, costs orders of magnitude more to move data:

Goal: minimize data movement cache $\rightarrow$ main mem
Notation: "communication"

Matmul: Theorem: gives a lower bound on \# words moved between main memory and cache of size \( M \) assuming do usual \( 2n^3 \) flops, done in any order (Hong, Kung 1981).

There is a widely known and used alg at attaining lower bound.

2004: extended to parallel case
2011: extended to any algorithm that "smells like 3 nest loops": includes matmul, G-E, LS, eig...
Usual algorithms for GE, LS, eig,... cannot attain lower bound, just by reordering ops. Will sketch some, widely used, new algorithms.

Extends to other computer architectures (eg parallel)
(many possible class projects)

Need simple model of comm. costs:
Bandwidth (bw), Latency

Intuition: freeway from Berkeley to Sacramento

\[ BW = \text{#cars/hour that can go from B to S} \]

\[ \text{#cars/hour} = \text{density (#cars/mile/lane)} \times \text{velocity (miles/hr)} \times \text{lanes} \]

Latency = how long it takes 1 car to get from B to S

\[ \text{Time (hours)} = \frac{\text{distance (miles)}}{\text{velocity (miles/hour)}} \]
So minimum time to move \( n \) cars from \( B \rightarrow S \) when they all travel in one "convoy" as close as possible

\[
\text{Time (hrs)} = \text{time for 1st car} + \text{time for remaining cars} = \text{latency} + \frac{n}{bw}
\]

Same idea for moving data

Time to move \( w \) words from DRAM to Cache

\[
= \text{latency} + \frac{w}{bw}
\]

assuming all words in one "message"

Moving \( w \) words in \( m \) messages costs

\[
m \cdot \text{latency} + \frac{w}{bw}
\]

Notation:

\[
m \cdot \alpha + w \cdot \beta = \\
\alpha = \text{latency} \quad \text{comm cost} \\
\beta = \frac{1}{bw} \\
g = \text{time per flop} \\
f = \# \text{flops}
\]
Total time = \( t \cdot f + w \cdot \beta + m \cdot \alpha \)

Today: \( f \ll \beta \ll \alpha \)
growing apart exponentially
same model for energy

Flops dominates if
\( f \cdot \beta \leq w \cdot \beta + m \cdot \alpha \)

Comm dominates if
\( f \cdot \beta < w \cdot \beta + m \cdot \alpha \)

**Notation:** Computational Intensity:
\[
q = \frac{f}{w} = \text{"flops per word moved"}
\]

\( q \) needs to be large to be fast
\( f \cdot \beta \geq w \cdot \beta \)

\( \Rightarrow q = \frac{f}{w} \frac{\beta}{f} \gg 1 \)
History of how this has influenced algorithms:

In the beginning, was the do-loop
Enough for first libraries eg EISPACK (mid 1960s)

People didn't worry about commun, just flops and accuracy

BLAS-1 Library (Basic Linear Algebra Subprograms)
Standard library of 15 ops mostly on vectors:

1) \( y = \alpha \cdot x + y \) \( x, y \) vectors \( \alpha \) scalar
   "AXPY" for short inner loop of GE

2) dot product

3) \( \| x \|_2 = \sqrt{\sum x_i^2} \)

4) find largest entry \( \| x \|_1 \) in \( x \)
Motivation:
- easier programming
- readability
- robustness
- C avoids over/underflow
- portable + efficient

Can't minimize comm

Comp. Intensity = \( q = \frac{f}{w} = \frac{2n}{2n} = 1 \)

det product

BLAS-2 library (mid 1980s)
- standard library of 25 ops
  - on pairs of matrices + vectors

1) \( y = \alpha y + \beta A x \)
   - \( A \) matrix
   - \( x, y \) vectors
   - \( \alpha, \beta \) scalars

   "GEMV"

   lots of variations
   - \( A \) symmetric
   - triangular...
   - \( A^T \)
   - or \( A^T \)

2) \( A = A + \alpha x y^T \)
   - rank-one update
   "GER" : 2 inner-most loops
   of GE
(3) Solve $Tx = b$, $T$ triangular

Motivation: similar to BLAS1
+ more opportunities for optimization on vector computers

Not much improvement on $g$

\[ GEMV: g = \frac{f}{w} = \frac{2n^2}{n^2 + n + 1} \approx 2 \]

BLAS-3 library (late 1980s)

9 operations on pairs of matrices

1) $C = \beta C + \alpha AB$ \quad $A, B, C$ matrices, $\alpha, \beta$ scalars
   "GEMM"

2) $C = \beta C + \alpha AA^T$ \quad $A^{n \times k}$
   "SYRK"

3) Solve $TX = B$, $T$ triangular

For GEMM $g = \frac{f}{w} = \frac{2n^3}{3n^2 + n^2} = \frac{n}{2}$

But usual 3 nested loops for matmul no help, actual $g = 2$
Hint: BLAS-k does $O(n^k)$ operation on data of dimension $n$

BLAS3 led community to rebuild all linear algebra to use GEMM etc as much as possible. LAPACK, ScalAPACK.

Goal: Prove comm lower bound for matmul for

```
  CPU
  cache
      size M
      slow, main mem
```

Easy case: if $3n^2 \leq M \Rightarrow$
read all data into cache, do all work, write C back to DRAM

Hard case: $3n^2 > M$

Then (Hong, Kung 1981): To multiply $C = A \cdot B$ using usual $2n^3$ flops in any (correct) order, #words moved $= \Omega \left( \frac{n^3}{M} \right)$
More modern proof based on
Iony, Tiskin, Toledo (2009)

Extends to rectangular, sparse matrices

\[ \Omega \left( \frac{\# \text{flops}}{UM} \right) \]

Proof sketch (ignore constants)

Suppose we fill cache with M words,
do as many flops as possible,
store results back to main mem
Upper bound \# flops possible by G
\[ \Rightarrow \text{Doing } G \text{ flops costs } 2M \text{ words moved} \]
\[ \Rightarrow \text{Since we do } 2n^3 \text{ flops, need to repeat } 2n^3/G \text{ times} \]
\[ \Rightarrow \text{\# words moved } \frac{2n^3}{G} \cdot 2M \]

Need G (or an upper bound)

use geometric model to get G
represent algebra \( n \times n \times n \) lattice

with axes \( i, j, k \)

\[
(c(i, j, k)) = C(i, j) + A(i, k)B(k, j)
\]

Bound \( |V| \)

using \( |V_A|, |V_B|, |V_C| \)

because \( |V_A| + |V_B| + |V_C| \leq M \)

Intuition

\[
|V| = x \cdot y \cdot z
\]

\[
|V_A| = x \cdot y
\]

\[
|V_B| = x \cdot z
\]

\[
|V_C| = y \cdot z
\]

\[
\sqrt{|V_A| \cdot |V_B| \cdot |V_C|} = x \cdot y \cdot z = |V|
\]

Thm (Loomis and Whitney 1949)

\[
\sqrt{|V_A| \cdot |V_B| \cdot |V_C|} \geq |V|
\]
$G = 1 |V| = \sum_{M \cdot M \cdot M} = M^{3/2}$

# words moved $\geq \frac{2n^3}{G} \cdot 2M = \frac{2n^3}{M^{3/2}} \cdot 2M = \Omega \left( \frac{n^3}{M} \right)$

If careful with constants:

# words moved $\geq \frac{2n^3}{UM}$

attainable!

Goal: Optimal MatMul:

What shape of $V$ has projections $V_A, V_B, V_C$ all of size $M$,

$|V| = M^{3/2}$?

Cube: $V$: $M^{1/2} \times M^{1/2} \times M^{1/2}$ cube

Algorithm: break $A, B, C$ into square submatrices that all fit in cache, as large as possible.

Read each triple of submatrices into cache, multiply, put answer back into main mem.
\[ A_{n \times n} = \begin{bmatrix} b & \cdots & b \\ b & \ddots & b \\ b & & \ddots \end{bmatrix} \]

\[ A[i,j] \text{ b\times b submatrix} \]

for \( i = 1 \) to \( n/b \)
  for \( j = 1 \) to \( n/b \)
    read \( C[i,j] \) into cache... \( b^2 \) words
    for \( k = 1 \) to \( n/b \)
      read \( A[i,k] \) \( B[k,j] \) into cache... \( 2b^2 \) words
      \( C[i,j] = C[i,j] + A[i,k] \cdot B[k,j] \)
      \( b \times b \) matmul, all in cache, 3 more nested loops
    end for
  end for
end for

Write \( C[i,j] \) to main mem... \( b^2 \) words

Total words moved =
\[ 2 \cdot \frac{n}{b} \cdot \frac{n}{b} \cdot b^2 + (\frac{n}{b})^3 \cdot 2b^2 \]
\[ = 2n^2 + \frac{2n^3}{b} \]
where \( b = \sqrt[3]{\frac{M}{b}} \)
\[ = O\left(\frac{b^3}{\sqrt[3]{M}}\right) \]

What about rectangular case?
What if some dimension < \( b \)?
**Ex:** Ax

General case for $m \times k \times n$ matmul:

$$T \geq \max \left( \frac{\min \left( \frac{k \cdot n}{m} \right)}{1 + \text{size of input}} \right)$$

Idea of using Loomis-Whitney enough for linear algebra, extends to any algorithm that looks like:

- nested loops
- any number of arrays
- "any" subscripts, e.g. $i$, $i+j$, $i-2j+k$,..

Get a lower bound and optimal alg:

$$\text{# words moved} = \Omega \left( \frac{\#\text{loop iterations}}{m} \right)$$

$e$ depends on details of alg, uses generalization of Loomis Whitney called Hölder-Brascamp-Lieb Inequality.