CS 3331 Numerical Methods

Introduction to BLAS/LAPACK

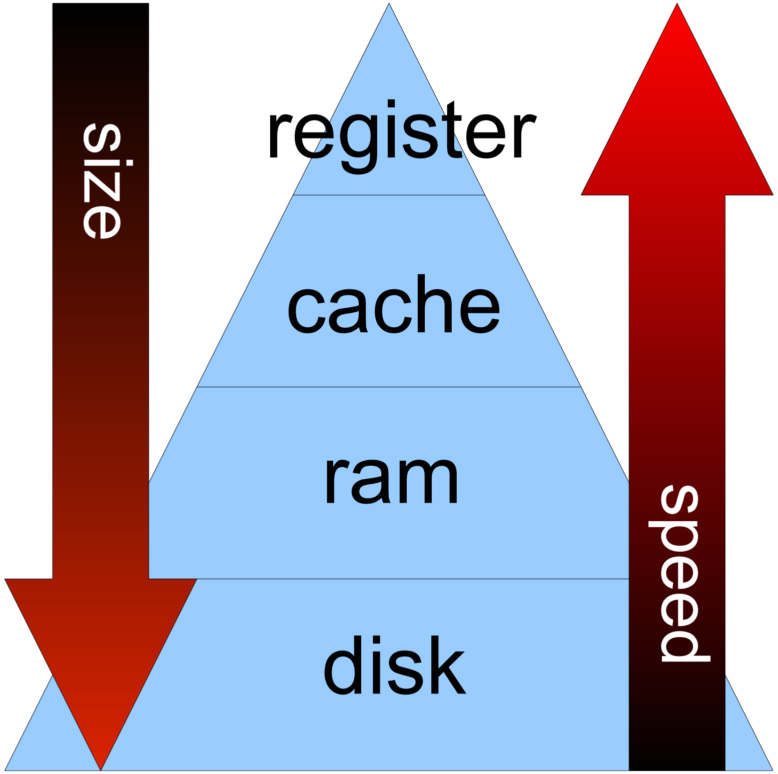
Cherung Lee

Outline

- Block algorithms
  - Memory hierarchy
  - Matrix-matrix multiplication
- BLAS/LAPACK

Block Algorithms

Memory hierarchy*



- Running time
  - Flops*(time per flop)
  - Words moved/bandwidth
  - Messages*latency

- Trend (improvement/year)
  - Time per flop : 59%.
  - Memory BW : 23%
  - Memory latency : 5.5%

*Jim Demmel's talk at MMDS2008

Ratio of flops to memory access

- Let f be the number of flops, m be number of memory access. Then $q = f/m$ is the ratio of flops to memory access.

- Let t_{comp} be the time per flops, t_{mem} be the time per memory access. The running time is

$$f \cdot t_{\text{comp}} + m \cdot t_{\text{mem}} = f \cdot t_{\text{comp}} \left(1 + \frac{m \cdot t_{\text{mem}}}{f \cdot t_{\text{comp}}}\right) = f \cdot t_{\text{comp}} \left(1 + \frac{t_{\text{mem}}}{q \cdot t_{\text{comp}}}\right)$$

Operation	f	m	q
$\mathbf{y} = \alpha \mathbf{x} + \mathbf{y}$	$2n$	$3n + 1$	$2/3$
$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{y}$	$2n^2$	$n^2 + 3n$	2
$\mathbf{C} = \mathbf{A}\mathbf{B} + \mathbf{C}$	$2n^3$	$4n^2$	$n/2$

Matrix-matrix multiplication*

- Suppose there are fast and slow memory. The size of fast memory is $M(\approx 2n)$, and the size of slow memory is $> 3n^2$.
- There loop algorithm for $\mathbf{C} = \mathbf{AB}$

```
for i = 1:n              % read row i of A into fast memory
    for j = 1:n          % read column j of B into fast memory
        for k = 1:n      % read C(i,j) into fast memory
            C(i,j)=C(i,j) + A(i,k)*B(k,j)
        end
    end
end
```

*Jim Demmel, Applied numerical linear algebra, SIAM 1997

- Memory access counts.
  - Read $\mathbf{B}$ n times: n^3.
  - Read $\mathbf{A}$ 1 time: n^2.
  - Read and write $\mathbf{C}$ 2 times: $2n^2$.
- Total memory access is $n^3 + 3n^2$
- The ratio $q = 2n^3/(n^3 + 3n^2) \approx 2$.
  - In the different order of theoretical value $n/2$.

Block matrix-matrix multiplication

- Partition $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ into $N \times N$ blocks. Each block submatrix is of size n/N. And suppose $M \geq 3(n/N)^2$.
- Denote $\mathbf{A}[I, J]$ the I, J block submatrix of $\mathbf{A}$. Same to $\mathbf{B}$, $\mathbf{C}$.

```
for I = 1:N            %
    for J = 1:N        % read C[I,J] into fast memory
        for K = 1:N    % read A[I,K] and B[K,J] into fast memory
            C[I,J]=C[I,J] + A[I,K]*B[K,J]
        end
    end
end
```

- Memory access counts.
  - Read $\mathbf{B}$ N times: Nn^2.
  - Read $\mathbf{A}$ N time: Nn^2.
  - Read and write $\mathbf{C}$ 2 times: $2n^2$.
- Total memory access is $(2N+2)n^2 \approx 2Nn^2$.
- The optimal N is $n\sqrt{3/M}$, where M is the size of fast memory. (Let $M = 3(n/N)^2$, $N = n\sqrt{3/M}$.)
- The ratio $q \approx 2n^3/(2Nn^3) \approx \sqrt{M/3} \approx n/N$.

BLAS/LAPACK

BLAS/LAPACK

- BLAS: **B**asic **L**inear **A**lgebra **S**ubprograms
- LAPACK: **L**inear **A**lgebra **PACK**age
- The engine of Matlab, Octave and many other high performance software. (Dense and band matrices only.)
- Written in Fortran 77, but also have interfaces to C, (C++), Java, Fortran 90.
- Similar functions for real and complex matrices, in both single and double precision.

BLAS

- Level 1: vector operations
  - ex: $\mathbf{y} = a\mathbf{x} + \mathbf{y}$
- Level 2: matrix-vector operations
  - ex: $\mathbf{y} = a\mathbf{A}\mathbf{x} + b\mathbf{y}$
- Level 3: matrix-matrix operations
  - ex: $\mathbf{C} = a\mathbf{A}\mathbf{B} + b\mathbf{C}$

LAPACK

- Solving linear equations,
- Least-squares solutions of linear systems of equations,
- Eigenvalue problems, and singular value problems.
- The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur)
- Related computations such as reordering of the Schur factorizations and estimating condition numbers.

Availability

- Official site: *http://www.netlib.org/blas/* and

  http://www.netlib.org/lapack/

- Optimized version: Goto Blas, Altas (Automatically Tuned Linear Algebra Software),

- Commercial packages: Intel MKL, AMD ACML, IBM ESSL, HP MLIB, NVIDIA CUDA, Apple Accelerate ...

- Parallel version are also available.

- Libraries for sparse matrix computation are another story.