Lecture 25
Linear Algebra Software

MIT 18.335J / 6.337J
Introduction to Numerical Methods

Per-Olof Persson (persson@mit.edu)
December 12, 2007
BLAS

- Basic Linear Algebra Subroutines (BLAS)
  - Standardized interface for simple vector and matrix operations
  - Manufacturers provide optimized implementations for their machines

- History:
  - BLAS1 (1970s) – Vector operations: $\alpha = x^T y$, $y = \alpha x + y$
  - BLAS2 (mid 1980s) – Matrix-vector operations: $y = Ax + y$
  - BLAS3 (late 1980s) – Matrix-matrix operations: $C = AB + C$

- Efficient cache-aware implementations give almost peak performance for BLAS3 operations

- High level algorithms (Gaussian elimination, etc) use BLAS but no other machine dependent code
  - Performance and portability
Memory Hierarchy and High Level BLAS

- Modern computers use a memory hierarchy
- From fast/expensive to cheap/slow: Registers, L1 cache, L2 cache, local memory, remote memory, secondary memory
- Fast algorithms perform many operations on each memory block to minimize memory access (cache reuse)
- Only BLAS3 has potential for very high performance

<table>
<thead>
<tr>
<th>BLAS</th>
<th>Memory Refs</th>
<th>Flops</th>
<th>Flops / Memory Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1 ((y = \alpha x + y))</td>
<td>3n</td>
<td>2n</td>
<td>2/3</td>
</tr>
<tr>
<td>Level 2 ((y = Ax + y))</td>
<td>(n^2)</td>
<td>(2n^2)</td>
<td>2</td>
</tr>
<tr>
<td>Level 3 ((C = AB + C'))</td>
<td>(4n^2)</td>
<td>(2n^3)</td>
<td>(n/2)</td>
</tr>
</tbody>
</table>
BLAS Performance

- For high performance write algorithms in terms of BLAS3 operations
BLAS Implementations

- **Vendor provided:**
  - Intel Math Kernel Library (MKL), AMD Core Math Library (ACML)
  - Sun Performance Library
  - SGI Scientific Computing Software Library

- **Automatically Tuned Linear Algebra Software (ATLAS)**
  - Analyzes hardware to produce BLAS libraries for any platform
  - Used in MATLAB, precompiled libraries freely available
  - Sometimes outperforms vendor libraries

- **GOTO BLAS (mainly for Intel processors)**
  - Manually optimized assembly code, currently the fastest implementation
Calling BLAS from C

- BLAS standard based on Fortran 77:
  - All memory must be preallocated
  - All variables are passed by reference

- Example: Double precision matrix-matrix multiply \((C = \alpha AB + \beta C)\):
  \[
  \text{dgemm}(&\text{transa}, &\text{transb}, &\text{m}, &\text{n}, &\text{k}, &\alpha, \text{A}, &\text{lda}, \\
  &\text{B}, &\text{ldb}, &\beta, \text{C}, &\text{ldc});
  \]

  - \text{transa}, etc: Matrix transpose \(’T’\) or not \(’N’\)
  - \text{lda}, etc: Leading dimensions of matrices
  - Some platforms/compilers do not require the trailing underscore
  - In C++, declare functions with \texttt{extern "C"}

- See also C BLAS interface in ATLAS
LAPACK

- Standard library for dense/banded linear algebra
  - Linear systems: $Ax = b$
  - Least squares problems: $\min_x \|Ax - b\|_2$
  - Eigenvalue problems: $Ax = \lambda x, Ax = \lambda Bx$
  - Singular value decomposition (SVD): $A = U\Sigma V^T$

- Algorithms use BLAS3 as much as possible

- Used by MATLAB (since version 6)

- LAPACK Search Engine useful for finding routines
LAPACK Performance

- Matrix-matrix multiply and LU factorization as function of matrix size
- About 80% of peak performance for LU factorization of large matrices
Sparse Solver Packages

- **UMFpack** (Unsymmetric MultiFrontal method)
  - Used in MATLAB (since version 7.1), no parallel version

- **PARADISO**
  - Serial and shared memory, used in Intel MKL

- **SuperLU**
  - Versions for serial and parallel computers (shared/distributed)
  - “Static pivoting” for distributed machines (increase small pivots, iterative refinement for accuracy)

- **MUMPS** (MUltifrontal Massively Parallel sparse direct Solver)
  - Versions for serial and parallel computers (distributed)