Introduction to parallel computing

Matrix Libraries and Other Miscellaneous Topics

Zhiao Shi (Modifications by Will French)
Advanced Computing Center for Research & Education
Vanderbilt University
Today

- Overview of linear algebra/libraries
- Parallel packages/toolkits in Python, Matlab
Why are matrix algebra libraries important?

• Matrix algebra shows up in a number of applications:
  • Solving systems of equations
  • Finite element analysis
  • Quantum physics and chemistry
  • Game theory
  • Machine learning and data mining
• C/Fortran matrix libraries are often used by interpreted languages like Matlab, Python, and R.
Defining “dense matrix”

- **Sparse** matrix: most of the elements are zero.
- **Structured** matrix: follows pattern that allows computational complexity to be reduced.

\[
\begin{bmatrix}
    a_0 & a_{-1} & a_{-2} & \ldots & \ldots & a_{-n+1} \\
    a_1 & a_0 & a_{-1} & \ddots & & \\
    a_2 & a_1 & \ddots & \ddots & \ddots & \\
    & \ddots & \ddots & \ddots & \ddots & \ddots \\
    a_{n-1} & & \ldots & a_1 & a_0 & a_{-1} \\
    a_n & \ldots & \ldots & a_2 & a_1 & a_0
\end{bmatrix}
\]

Toeplitz matrix

- **Dense** matrix: neither sparse nor structured.
Motivations for dense linear algebra

• Dense algorithms are easy to understand.
• Some applications yield large dense matrices.
  • LINPACK benchmark to rank the world’s fastest computer uses dense matrix to solve

\[ \mathbf{A}\mathbf{x} = \mathbf{b} \]

• Large sparse matrix algorithms often yield smaller dense problems.
• Due to their regular structure, parallel computations involving matrices and vectors readily lend themselves to data-decomposition.
Linear algebra libraries

• Because some tasks are quite common across many science and engineering applications, groups of researchers have put a lot of effort into writing scientific libraries: collections of routines for performing these commonly-used tasks (e.g., linear algebra solvers).

• The people who write these libraries know a lot more about these things than we do.

• So, a good strategy is to use their libraries, rather than trying to write our own.
Libraries for dense linear algebra

• BLAS
• ATLAS
• LAPACK
• ScaLAPACK
Memory hierarchy: put these memory locations in order from fastest to slowest access speed.

- Main memory (RAM)
- L1 Cache
- L2 Cache
- Hard disk
- Registers
BLAS

- http://www.netlib.org/blas
- Standardized API for subroutines to perform basic linear algebra operations such as vector and matrix multiplication.
- First published in 1979 and are used to build larger packages such as LAPACK
- Some hardware vendors provides highly optimized implementations of BLAS interface, e.g. Intel.
- 3 levels of BLAS:
  - BLAS1: Vector-vector operations
  - BLAS2: Matrix-vector operations
  - BLAS3: Matrix-Matrix Operations
What does FLOP stand for in a HPC context?

Hint: Top500.org lists the 500 (publicly known) fastest supercomputers in the world by FLOPs.
BLAS

- **BLAS1 (1970s)**
  - vector operations: dot product \((y = \alpha x + y)\) etc.
  - \(m = 2n, f = 2n, q \sim 1\) or less.
  - \(m\) - memory references, \(f\) - floating point operations (flops), \(q\): flops/memory reference.

- **BLAS2 (mid 1980s)**
  - matrix-vector operations: matrix vector multiply, etc.
  - \(m = n^2, f = 2n^2, q \sim 2\) faster than BLAS1

- **BLAS3 (late 1980s)**
  - matrix-matrix operations: matrix-matrix multiply etc.
  - \(m \geq 4n^2, f = O(n^3)\), \(q\) is as large as \(n\).
  - Potentially much faster than BLAS2.
  - Good algorithms use BLAS3 when possible.
Other implementations

• ACML
  • AMD, Atholon/Opteron, Linux/Windows

• ATLAS
  • Open source, C/Fortran77

• ESSL
  • IBM, PowerPC, AIX/Linux

• GotoBLAS
  • Univ. of Texas Austin

• Intel MKL
  • Intel Math Kernel Library, Linux/Windows

• uBLAS
  • C++ template, part of Boost library
ATLAS

• The Automatically Tuned Linear Algebra Software package (ATLAS) is a self-tuned version of BLAS (it also includes a few LAPACK routines).

• ATLAS is substantially faster than the generic version of BLAS.

• Optimization techniques:
  • Parameterization: e.g. block size
  • Multiple implementation
  • Code generation
How are the following organized in a computer? What goes where?

• Motherboard
• Hardware thread (hyperthreading)
• Core
• Processor
Goto BLAS

• Another implementation of BLAS, developed by Kazushige Goto (UT Austin).
• This version is unusual, in addition to optimizing for cache, it also optimizes for the Translation Lookaside Buffer (TLB), which is a special cache that contains the page table entries which maps virtual addresses onto physical addresses (often ignored by library developers).
• Optimizing for the TLB (minimize TLB misses) would be more effective than optimizing for cache.
LAPACK

• LAPACK (Linear Algebra PACKage) solves dense or special-case sparse systems of equations depending on matrix properties such as:
  • Precision: single, double
  • Data type: real, complex
  • Shape: diagonal, bidiagonal, tridiagonal, banded, triangular, trapezoidal, Hessenberg, general dense
  • Properties: orthogonal, positive definite, Hermetian (complex), symmetric, general

• LAPACK is built on top of BLAS, which means it can benefit from ATLAS or Goto BLAS.
LAPACK: a Library and an API

• It is also an Application Programming Interface (API): a definition of a set of routines, their arguments, and their behaviors. Anyone can write an implementation of LAPACK.

• LAPACK is a good choice for non-parallelized solving, because its popularity has convinced many supercomputer vendors to write their own, highly tuned versions.
LAPACK portability

• LAPACK calls BLAS as much as possible, using BLAS as building blocks.
• The efficiency of LAPACK depends on the efficient implementations of BLAS (provided by computer vendors or others for their machines).
• BLAS forms a low-level interface between LAPACK and machine architectures.
• Above this level, almost all of the LAPACK is truly portable.
As a rule of thumb, what is the maximum number of threads or processes you would want to use in a parallel program?

A. As many as the operating system allows
B. Half as many as the operating system allows
C. A number equal to the number of CPU cores on the machine
D. Twice the number of CPU cores on a machine
An example of using LAPACK

• We want to solve a system of linear equations with DGESV routine.

\[
\begin{bmatrix}
3.1 & 1.3 & -5.7 \\
1.0 & -6.9 & 5.8 \\
3.4 & 7.2 & -8.8 \\
\end{bmatrix} \times \begin{bmatrix}
-1.3 \\
-0.1 \\
1.8 \\
\end{bmatrix}
\]

• LAPACK routine:

```fortran
SUBROUTINE DGESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
```

# include <stdio.h>
#define size 3

int main()
{
  int i, j , c1, c2, pivot[size], ok;
  double A[size][size] = {{ 3.1, 1.3, -5.7}, { 1.0, -6.9, 5.8}, { 3.4, 7.2, -8.8}};
  double b[size] = {-1.3,-0.1,1.8};
  double AA[size*size];

  for (i=0; i<size; i++)
    for(j=0; j<size; j++)
      AA[j+size*i]=A[j][i];

  c1=size;
  c2=1;

  dgesv_(&c1, &c2, AA, &c1, pivot, b, &c1, &ok);

  for (j=0; j<size; j++)
    printf("%e\n", b[j]);
}
ScaLAPACK

- ScaLAPACK is the distributed parallel version of LAPACK. It contains only a subset of the LAPACK routines.
ScaLAPACK

• “Local” for serial libraries.
  • LAPACK contains and is built on BLAS.
  • Vendor provides optimized BLAS for their architectures.
  • ScaLAPACK uses blocking algorithms to keep and reuse data in the lowest levels of the memory hierarchy.
ScaLAPACK

• “Global” for parallel libraries.
  • ScaLAPACK contains and is built on PBLAS.
  • Used for MPP systems to perform parallel tasks.
  • Built on BLACS in order to transfer local data from one processor to others.
• The BLACS library is built on top of a message passing library.
  • Need to install the version of BLACS that matches the message passing library present on a given machine.
  • Most BLACS interfaces to MPI, but older libraries (such as PVM still supported).
ScaLAPACK routines rely on:

- BLAS routines for optimized calculations for local arrays on single processors (the local arrays containing pieces of the global array actually being worked on in parallel).

- BLACS routines for optimized communication of data between local arrays when needed.

- Only the BLAS and BLACS libraries depend on the characteristics of the real computer being used (cache/memory sizes, interconnection topology). Thus the ScaLAPACK user is isolated from having to know about program for a given machines specifications.
What is the preferred way to compute the exponent of an array of data (element by element) in Python, Matlab, or R?

A. Within a loop
B. Without a loop, using functions that operate on arrays of data
C. All of the above
D. None of the above
Vectorization

• Vectorization often refers to avoiding loops in interpreted languages like Python, Matlab, or R.
  • A single call to an underlying C library operating on arrays of data avoids interpretation overhead.

• Vectorization may also refer to dedicated vector processing units (VPUs) built into the chip for performing SIMD operations.
  • Intel compilers often auto-vectorize (compile with -vec-report) loops into SIMD instructions to be performed on the VPU.
Python Parallel Packages

• Python has a lot of parallel packages available, mostly for doing single-computer computations to take advantage of multi-core processors
  • https://wiki.python.org/moin/ParallelProcessing

• There are also packages available for doing processing across multiple processes/computers.
  • mpi4pi
  • pyMPI
Matlab Parallel Packages

• Matlab offers licenses for parallel processing.
  • Parallel Processing Toolbox.
    • Parallel for loops.
    • GPU offloading.
    • Limited to single-node analysis.
• Distributed Computing Cluster.
  • Capable of running Matlab across multiple machines.
  • Expensive.
What is the best indicator of processor speed?

- Clock speed
- Number of CPU cores
- Number of hardware threads
- Cache size and speed
- Other factors
- All of the above
Next time

• Computing on graphics processing units!