CS 5220

Dense LA

David Bindel

2024-10-22

Logistics

Special talk!

CS colloquium / Salton lecture

Thursday, Oct 22, 11:45-12:45

Gates G01

Speaker: Jack Dongarra

Title: An Overview of High Performance Computing and Responsibly Reckless Algorithms

Please come out!

Participation

Trying a few things:

- More small-group discussion
- Poll Everywhere
- CMS quizzes

Open to suggestions! Help me make this time most useful to you.

Projects 1 and 2

- Project 1 is graded
  - NB re ChatGPT - doesn't seem to have helped
- Project 2 extended to 10/24
  - Wrapping earlier is good!
  - *Do* spend time on the write-up
  - P2 peer evals are due 10/25

Homework 3

- See hw3 prompt
- Due Oct 29

Project 3: Shallow water

- FA2020 version
- Linearized SWE

Final project

Final project basics

- Scope
  - Analogous to other projects
  - Groups of 1-5
  - *Must* be related to performance
- Platform
  - Does *not* need to be on Perlmutter
  - Or even parallel
  - Or even involve new code!

Final project ideas

From my own work:

- Parallel optimization (variable evaluation time)
- Smart GPU memory in a GPU-accelerated time stepper
- Blocked relaxation methods for kernel systems
- Bayesian optimization and auto-tuning
- Mixed-precision linear algebra operations

Final project ideas

- From this class
  - Previous semester projects
  - Current projects on other platforms
- PDE solves, dynamic programming (shortest path), etc
- Tree operations
- Things from your research

Final project ideas

"But I want to parallelize ML workloads!"

- Knock yourself out
  - I'm interested in training constitutive models...
- Be aware of what course compute resources we have
- Compare to a fair baseline

Final project timeline

- Proposal due 10/31
- Final outline 11/26 (just before Thanksgiving)
- Project presentations 12/3-4
- Final deliverable due 12/19 at 12 PM

Dense LA

Where we are

- This week: *dense* linear algebra
- Next week: *sparse* linear algebra

Nutshell NLA

- Linear systems: $Ax = b$
- Least squares: minimize $\|Ax - b\|_2^2$
- Eigenvalues: $Ax = \lambda x$

Nutshell NLA

- Basic paradigm: matrix factorization
  - $A = LU$, $A = LL^T$
  - $A = QR$
  - $A = V\Lambda V^{-1}$, $A = QTQ^T$
  - $A = U\Sigma V^T$
- Factorization $\equiv$ switch to basis that makes problem easy

Dense and sparse

Two flavors: dense and sparse

Dense

Common structures, no complicated indexing

- General dense (all entries nonzero)
- Banded (zero below/above some diagonal)
- Symmetric/Hermitian
- Standard, robust algorithms (LAPACK)

Sparse

Stuff not stored in dense form!

- Maybe few nonzeros
  - e.g. compressed sparse row formats
- May be implicit (e.g. via finite differencing)
- May be “dense”, but with compact repn (e.g. via FFT)
- Mostly iterations; varied and subtle
- Build on dense ideas

History: BLAS 1 (1973–1977)

15 ops (mostly) on vectors

- BLAS 1 == $O(n^1)$ ops on $O(n^1)$ data
- Up to four versions of each: S/D/C/Z
- Example: DAXPY
  - Double precision (real)
  - Computes $Ax + y$

BLAS 1 goals

- Raise level of programming abstraction
- Robust implementation (e.g. avoid over/underflow)
- Portable interface, efficient machine-specific implementation
- Used in LINPACK (and EISPACK?)

History: BLAS 2 (1984–1986)

25 ops (mostly) on matrix/vector pairs

- BLAS2 == $O(n^2)$ ops on $O(n^2)$ data
- Different data types and matrix types
- Example: DGEMV
  - Double precision
  - GEneral matrix
  - Matrix-Vector product

BLAS 2 goals

- BLAS1 insufficient
- BLAS2 for better vectorization (when vector machines roamed)

History: BLAS 3 (1987–1988)

9 ops (mostly) on matrix/matrix

- BLAS3 == $O(n^3)$ ops on $O(n^2)$ data
- Different data types and matrix types
- Example: DGEMM
  - Double precision
  - GEneral matrix
  - Matrix-Matrix product

BLAS 3 goals

Efficient cache utilization!

LU for 2×2:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ c/a & 1 \end{bmatrix} \begin{bmatrix} a & b \\ 0 & d - bc/a \end{bmatrix}$$

Why BLAS?

Block elimination

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ 0 & D - CA^{-1}B \end{bmatrix}$$

Why BLAS?

Block LU

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{12} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$$
$$= \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{12}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{bmatrix}$$

Why BLAS?

Think of A as $k \times k$, k moderate:

```
[L11,U11] = small_lu(A);    % Small block LU
U12 = L11\B;                % Triangular solve
L12 = C/U11;                % "
S   = D-L21*U12;            % Rank k update
[L22,U22] = lu(S);          % Finish factoring
```

Three level-3 BLAS calls!

- Two triangular solves
- One rank-k update

LAPACK (1989-present)

- Supercedes earlier LINPACK and EISPACK
- High performance through BLAS
  - Parallel to the extent BLAS are parallel (on SMP)
  - Linear systems, least squares – near 100% BLAS 3
  - Eigenproblems, SVD — only about 50% BLAS 3
- Careful error bounds on everything
- Lots of variants for different structures

ScaLAPACK (1995-present)

- MPI implementations
- Only a small subset of LAPACK functionality

PLASMA (2008-present)

Parallel LA Software for Multicore Architectures

- Target: Shared memory multiprocessors
- Stacks on LAPACK/BLAS interfaces
- Tile algorithms, tile data layout, dynamic scheduling
- Other algorithmic ideas, too (randomization, etc)

MAGMA (2008-present)

Matrix Algebra for GPU and Multicore Architectures

- Target: CUDA, OpenCL, Xeon Phi
- Still stacks (e.g. on CUDA BLAS)
- Again: tile algorithms + data, dynamic scheduling
- Mixed precision algorithms (+ iterative refinement)

And beyond!

SLATE???

Much is housed at UTK ICL