Conjugate gradients, sparse matrix-vector multiplication, graphs, and meshes

Thanks to Aydin Buluc, Umit Catalyurek, Alan Edelman, and Kathy Yelick for some of these slides.
The middleware of scientific computing

Continuous physical modeling

Linear algebra

Computers

Ax = b
Example: The Temperature Problem

- A cabin in the snow
- Wall temperature is 0°, except for a radiator at 100°
- What is the temperature in the interior?
Example: *The Temperature Problem*

- A cabin in the snow (a square region 😊)
- Wall temperature is 0°, except for a radiator at 100°
- What is the temperature in the interior?
The physics: Poisson’s equation

\[ \nabla^2 u(x, y) \equiv \frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = f(x, y) \]

for \((x, y) \in \mathbb{R} = \{ (x, y) \mid a < x < b, \ c < y < d \}\), and

\[ u(x, y) = g(x, y) \]

for \((x, y)\) on the boundary of \(\mathbb{R}\).
Many Physical Models Use Stencil Computations

- PDE models of heat, fluids, structures, ...
- Weather, airplanes, bridges, bones, ...
- Game of Life
- many, many others
Model Problem: Solving Poisson’s equation for temperature

- Discrete approximation to Poisson’s equation:
  \[ t(i) = \frac{1}{4} \left( t(i-k) + t(i-1) + t(i+1) + t(i+k) \right) \]

- Intuitively:
  Temperature at a point is the average of the temperatures at surrounding points

\[ k = n^{1/2} \]
Examples of stencils

5-point stencil in 2D
(temperature problem)

7-point stencil in 3D
(3D temperature problem)

9-point stencil in 2D
(game of Life)

25-point stencil in 3D
(seismic modeling)

… and many more
Parallelizing Stencil Computations

- **Parallelism** is simple
  - Grid is a *regular* data structure
  - Even decomposition across processors gives *load balance*

- **Spatial locality** limits communication cost
  - Communicate only boundary values from neighboring patches

- **Communication volume**
  - \( v = \text{total # of boundary cells between patches} \)
Two-dimensional block decomposition

- $n$ mesh cells, $p$ processors
- Each processor has a patch of $n/p$ cells
- Block row (or block col) layout: $v = 2 \times p \times \sqrt{n}$
- 2-dimensional block layout: $v = 4 \times \sqrt{p} \times \sqrt{n}$
Detailed complexity measures for data movement I: Latency/Bandwidth Model

Moving data between processors by message-passing

- **Machine parameters:**
  - $\alpha$ or $t_{\text{startup}}$ latency (message startup time in seconds)
  - $\beta$ or $t_{\text{data}}$ inverse bandwidth (in seconds per word)
  - between nodes of Triton, $\alpha \sim 2.2 \times 10^{-6}$ and $\beta \sim 6.4 \times 10^{-9}$

- Time to send & recv or bcast a message of $w$ words: $\alpha + w \beta$

- $t_{\text{comm}}$ total communication time

- $t_{\text{comp}}$ total computation time

- Total parallel time: $t_p = t_{\text{comp}} + t_{\text{comm}}$
Ghost Nodes in Stencil Computations

Comm cost = \( \alpha \times (\# \text{messages}) + \beta \times (\text{total size of messages}) \)

- Keep a ghost copy of neighbors’ boundary nodes
- Communicate every second iteration, not every iteration
- Reduces \#messages, not total size of messages
- Costs extra memory and computation
- Can also use more than one layer of ghost nodes
Parallelism in Regular meshes

- Computing a Stencil on a regular mesh
  - need to communicate mesh points near boundary to neighboring processors.
    - Often done with ghost regions
  - Surface-to-volume ratio keeps communication down, but
    - Still may be problematic in practice

Implemented using “ghost” regions.
Adds memory overhead
Model Problem: Solving Poisson’s equation for temperature

- Discrete approximation to Poisson’s equation:

\[ t(i) = \frac{1}{4} \left( t(i-k) + t(i-1) + t(i+1) + t(i+k) \right) \]

- Intuitively:

Temperature at a point is the average of the temperatures at surrounding points
Model Problem: Solving Poisson’s equation for temperature

- For each $i$ from 1 to $n$, except on the boundaries:
  $$- t(i-k) - t(i-1) + 4*t(i) - t(i+1) - t(i+k) = 0$$

- $n$ equations in $n$ unknowns: $A*t = b$
- Each row of $A$ has at most 5 nonzeros
- In three dimensions, $k = n^{1/3}$ and each row has at most 7 nzs
A Stencil Computation Solves a System of Linear Equations

- Solve $Ax = b$ for $x$
- Matrix $A$, right-hand side vector $b$, unknown vector $x$
- $A$ is *sparse*: most of the entries are 0
Conjugate gradient iteration to solve $A^*x=b$

$x_0 = 0, \quad r_0 = b, \quad d_0 = r_0 \quad \text{(these are all vectors)}$

for $k = 1, 2, 3, \ldots$

$$\alpha_k = (r_{k-1}^T r_{k-1}) / (d_{k-1}^T A d_{k-1})$$  step length

$$x_k = x_{k-1} + \alpha_k d_{k-1}$$  approximate solution

$$r_k = r_{k-1} - \alpha_k A d_{k-1}$$  residual $= b - A x_k$

$$\beta_k = (r_k^T r_k) / (r_{k-1}^T r_{k-1})$$  improvement

$$d_k = r_k + \beta_k d_{k-1}$$  search direction

- One matrix-vector multiplication per iteration
- Two vector dot products per iteration
- Four $n$-vectors of working storage
Vector and matrix primitives for CG

- **DAXPY:** \( v = \alpha v + \beta w \)  
  - (vectors \( v, w \); scalars \( \alpha, \beta \))
  - Broadcast the scalars \( \alpha \) and \( \beta \), then independent * and +
  - comm volume = 2p, span = log n

- **DDOT:** \( \alpha = v^T w = \sum_j v[j]w[j] \)  
  - (vectors \( v, w \); scalar \( \alpha \))
  - Independent *, then + reduction
  - comm volume = p, span = log n

- **Matvec:** \( v = A^w \)  
  - (matrix A, vectors \( v, w \))
  - The hard part
  - But all you need is a subroutine to compute \( v \) from \( w \)
  - Sometimes you don’t need to store A (e.g. temperature problem)
  - Usually you do need to store A, but it’s sparse ...
**Broadcast and reduction**

- **Broadcast** of 1 value to \( p \) processors in \( \log p \) time

  ![Broadcast Diagram]

- **Reduction** of \( p \) values to 1 in \( \log p \) time

  ![Add-reduction Diagram]

- Takes advantage of associativity in +, *, min, max, etc.

\[ \alpha \]

\[ 1 \quad 3 \quad 1 \quad 0 \quad 4 \quad -6 \quad 3 \quad 2 \]
Where’s the data (temperature problem)?

- The matrix $A$: Nowhere!!
- The vectors $x$, $b$, $r$, $d$:
  - Each vector is one value per stencil point
  - Divide stencil points among processors, $n/p$ points each
- How do you divide up the $\sqrt{n}$ by $\sqrt{n}$ region of points?
  - Block row (or block col) layout: $v = 2 \times p \times \sqrt{n}$
  - 2-dimensional block layout: $v = 4 \times \sqrt{p} \times \sqrt{n}$
How do you partition the $\sqrt{n}$ by $\sqrt{n}$ stencil points?

- First version: number the grid by rows
- Leads to a block row decomposition of the region
- $v = 2 \times p \times \sqrt{n}$

In this example assumes a square boundary
How do you partition the $\sqrt{n}$ by $\sqrt{n}$ stencil points?

- Second version: 2D block decomposition
- Numbering is a little more complicated
- $v = 4 \times \sqrt{p} \times \sqrt{n}$
Where’s the data (temperature problem)?

- The matrix A: Nowhere!!

- The vectors x, b, r, d:
  - Each vector is one value per stencil point
  - Divide stencil points among processors, n/p points each

- How do you divide up the $\sqrt{n}$ by $\sqrt{n}$ region of points?

- Block row (or block col) layout: $v = 2 \times p \times \sqrt{n}$

- 2-dimensional block layout: $v = 4 \times \sqrt{p} \times \sqrt{n}$
The Landscape of $Ax = b$ Algorithms

Gaussian elimination

Iterative

Pivoting LU

GMRES, BiCGSTAB, ...

Cholesky

Conjugate gradient

Any matrix

Symmetric positive definite matrix

More Robust

More General

More Robust

Less Storage
Conjugate gradient in general

- CG can be used to solve *any* system $Ax = b$, if …
Conjugate gradient in general

- CG can be used to solve any system $Ax = b$, if …
- The matrix $A$ is symmetric ($a_{ij} = a_{ji}$) …
- … and positive definite (all eigenvalues > 0).
Conjugate gradient in general

• CG can be used to solve any system $Ax = b$, if …
• The matrix $A$ is symmetric ($a_{ij} = a_{ji}$) …
• … and positive definite (all eigenvalues > 0).

• Symmetric positive definite matrices occur a lot in scientific computing & data analysis!
Conjugate gradient in general

• CG can be used to solve any system $Ax = b$, if …
• The matrix $A$ is symmetric ($a_{ij} = a_{ji}$) …
• … and positive definite (all eigenvalues $> 0$).

• Symmetric positive definite matrices occur a lot in scientific computing & data analysis!

• But usually the matrix isn’t just a stencil.
• Now we do need to store the matrix $A$. Where’s the data?
Conjugate gradient in general

- CG can be used to solve any system $Ax = b$, if …
- The matrix $A$ is symmetric ($a_{ij} = a_{ji}$) …
- … and positive definite (all eigenvalues $> 0$).

- Symmetric positive definite matrices occur a lot in scientific computing & data analysis!

- But usually the matrix isn’t just a stencil.
- Now we do need to store the matrix $A$. Where’s the data?

- The key is to use graph data structures and algorithms.
Vector and matrix primitives for CG

- **DAXPY**: \( v = \alpha \cdot v + \beta \cdot w \)  
  (vectors \( v, w \); scalars \( \alpha, \beta \))
  - Broadcast the scalars \( \alpha \) and \( \beta \), then independent * and +
  - comm volume = 2p, span = log n

- **DDOT**: \( \alpha = v^T \cdot w = \sum_j v[j] \cdot w[j] \)  
  (vectors \( v, w \); scalar \( \alpha \))
  - Independent *, then + reduction
  - comm volume = p, span = log n

- **Matvec**: \( v = A \cdot w \)  
  (matrix \( A \), vectors \( v, w \))
  - The hard part
  - But all you need is a subroutine to compute \( v \) from \( w \)
  - Sometimes you don’t need to store \( A \) (e.g. temperature problem)
  - Usually you do need to store \( A \), but it’s sparse ...
Graphs and Sparse Matrices

- Sparse matrix is a representation of a (sparse) graph.

\[
\begin{array}{cccccc}
1 & 1 & & & & 1 \\
2 & 1 & 1 & 1 & & \\
3 & & 1 & 1 & 1 & \\
4 & & & 1 & 1 & \\
5 & & & & 1 & 1 \\
6 & & & & & 1
\end{array}
\]

- Matrix entries are edge weights.
- Number of nonzeros per row is the vertex degree.
- Edges represent data dependencies in matrix-vector multiplication.
**Parallel Dense Matrix-Vector Product (Review)**

- **y = A*x**, where A is a dense matrix

- **Layout:**
  - 1D by rows

- **Algorithm:**
  - Foreach processor j
  - Broadcast X(j)
  - Compute A(p)*x(j)

- **A(i) is the n by n/p block row that processor Pi owns**

- **Algorithm uses the formula**
  \[ Y(i) = A(i)*X = \sum_j A(i)*X(j) \]
Parallel sparse matrix-vector product

- Lay out matrix and vectors by rows
- \( y(i) = \text{sum}(A(i,j) \times x(j)) \)
- Only compute terms with \( A(i,j) \neq 0 \)

**Algorithm**
- Each processor \( i \):
  - Broadcast \( x(i) \)
  - Compute \( y(i) = \text{A}(i,:) \times x \)

**Optimizations**
- Only send each proc the parts of \( x \) it needs, to reduce comm
- Reorder matrix for better locality by graph partitioning
- Worry about balancing number of nonzeros / processor, if rows have very different nonzero counts
Data structure for sparse matrix $A$ (stored by rows)

- **Full matrix:**
  - 2-dimensional array of real or complex numbers
  - $(nrows \times ncols)$ memory

- **Sparse matrix:**
  - compressed row storage
  - about $(2 \times nzs + nrows)$ memory
Distributed-memory sparse matrix data structure

Each processor stores:
- # of local nonzeros
- range of local rows
- nonzeros in CSR form
Irregular mesh: NASA Airfoil in 2D
Composite Mesh from a Mechanical Structure
Converting the Mesh to a Matrix
Adaptive Mesh Refinement (AMR)

- Adaptive mesh around an explosion
- Refinement done by calculating errors
Shock waves in a gas dynamics using AMR (Adaptive Mesh Refinement)
See: http://www.llnl.gov/CASC/SAMRAI/
Irregular mesh: Tapered Tube (Multigrid)

Example of Prometheus meshes

Figure 6: Sample input grid and coarse grids
Scientific computation and data analysis

- Continuous physical modeling
  - Linear algebra
    - Computers
Scientific computation and data analysis

- Continuous physical modeling
  - Linear algebra
  - Computers

- Discrete structure analysis
  - Graph theory
  - Computers
Scientific computation and data analysis

- Continuous physical modeling
- Discrete structure analysis

Linear algebra & graph theory

Computers