ECE 669
Parallel Computer Architecture
Lecture 4
Parallel Applications
Outline

- Motivating Problems (application case studies)
- Classifying problems
- Parallelizing applications
- Examining tradeoffs
- Understanding communication costs
  - Remember: software and communication!
Simulating Ocean Currents

- **Model as two-dimensional grids**
  - Discretize in space and time
  - finer spatial and temporal resolution => greater accuracy
- **Many different computations per time step**
  - set up and solve equations
  - Concurrency across and within grid computations
- **Static and regular**

(a) Cross sections

(b) Spatial discretization of a cross section
Creating a Parallel Program

° **Pieces of the job:**
  • Identify work that can be done in parallel
    - work includes computation, data access and I/O
  • Partition work and perhaps data among processes
  • Manage data access, communication and synchronization

° **Simplification:**
  • How to represent big problem using simple computation and communication

° **Identifying the limiting factor**
  • Later: balancing resources
4 Steps in Creating a Parallel Program

- **Decomposition** of computation in tasks
- **Assignment** of tasks to processes
- **Orchestration** of data access, comm, synch.
- **Mapping** processes to processors
Decomposition

° Identify concurrency and decide level at which to exploit it

° Break up computation into tasks to be divided among processors
  • Tasks may become available dynamically
  • No. of available tasks may vary with time

° Goal: Enough tasks to keep processors busy, but not too many
  • Number of tasks available at a time is upper bound on achievable speedup
Limited Concurrency: Amdahl’s Law

° Most fundamental limitation on parallel speedup
° If fraction $s$ of seq execution is inherently serial,
  speedup $\leq \frac{1}{s}$
° Example: 2-phase calculation
  • sweep over $n$-by-$n$ grid and do some independent computation
  • sweep again and add each value to global sum
° Time for first phase = $\frac{n^2}{p}$
° Second phase serialized at global variable, so time = $n^2$
° Speedup $\leq \frac{2n^2}{\frac{n^2}{p} + n^2}$ or at most 2
° Trick: divide second phase into two
  • accumulate into private sum during sweep
  • add per-process private sum into global sum
° Parallel time is $\frac{n^2}{p} + \frac{n^2}{p} + p$, and speedup at best $\frac{2n^2}{2n^2 + p^2}$
Understanding Amdahl’s Law

(a)

(b)

(c)
Concurrent Profiles

- Area under curve is total work done, or time with 1 processor
- Horizontal extent is lower bound on time (infinite processors)

- Speedup is the ratio: \( \frac{\sum_{k=1}^{\infty} f_k k}{\sum_{k=1}^{\infty} f_k \left[ \frac{k}{p} \right]} \), base case: \( \frac{1}{s + \frac{1-s}{p}} \)
- Amdahl’s law applies to any overhead, not just limited concurrency
Applications

° Classes of problems
  • Continuum
  • Particle
  • Graph, Combinatorial

° Goal: Demystifying

° Differential equations ---> Parallel Program
Particle Problems

° Simulate the interactions of many particles evolving over time

° Computing forces is expensive
  • Locality
  • Methods take advantage of force law: \( G \frac{m_1 m_2}{r^2} \)

• Many time-steps, plenty of concurrency across stars within one
Graph problems

- Traveling salesman
- Network flow
- Dynamic programming

° Searching, sorting, lists,
° Generally unstructured
Continuous systems

- **Hyperbolic**
  \[
  \frac{\partial^2 A}{C^2 \partial T^2} = \nabla^2 A + B
  \]

- **Parabolic**
  \[
  \frac{\partial A}{\partial T} = \nabla^2 A + B
  \]

- **Elliptic**
  \[
  0 = \nabla^2 A + B
  \]
  - **Laplace:** $B$ is zero
  - **Poisson:** $B$ is non-zero

- **Examples:**
  - Heat diffusion
  - Electrostatic potential
  - Electromagnetic waves
Numerical solutions

Let’s do finite difference first

Eg. \( \frac{\partial A}{\partial T} = \frac{\partial^2 A}{\partial x^2} \)

Solve

- Discretize
- Form system of equations
- Solve ->
  - Direct methods
  - Indirect methods
  - Iterative
Discretize

- **Time**
  - Where
    \[
    \frac{\partial A}{\partial T} = \frac{A^{n+1} - A^n}{\Delta t}
    \]
    \[
    \Delta t = \frac{1}{T \text{ steps}}
    \]

- **Space**
  - **1st**
    \[
    \frac{\partial A}{\partial x} = \frac{A_{i+1} - A_i}{\Delta x}
    \]
    \[
    \Delta x = \frac{1}{X \text{ grid points}}
    \]
  - **2nd**
    \[
    \frac{\partial}{\partial x} \left( \frac{\partial A}{\partial x} \right) = \frac{(A_{i+1} - A_i) - (A_i - A_{i-1})}{\Delta x^2}
    \]
    \[
    \frac{\partial^2 A}{\partial x^2} = \frac{A_{i+1} - 2A_i + A_{i-1}}{\Delta x^2}
    \]

  - Can use other discretizations
    - Backward
    - Leap frog
### 1D Case

\[
\frac{\partial A}{\partial T} = \frac{\partial^2 A}{\partial x^2} + B
\]

\[
\frac{A_i^{n+1} - A_i^n}{\Delta t} = \frac{1}{\Delta x^2} \left[ A_{i+1}^n - 2 A_i^n + A_{i-1}^n \right] + B_i
\]

Or

\[
A_i^{n+1} = \frac{\Delta t}{\Delta x^2} \left[ A_{i+1}^n - 2 A_i^n + A_{i-1}^n \right] + B_i \Delta t + A_i^n
\]

\[
\begin{bmatrix}
A_{x}^{n+1} \\
A_i^{n+1} \\
A_{2i}^{n+1} \\
A_i^{n+1}
\end{bmatrix} =
\begin{bmatrix}
\frac{\Delta t}{\Delta x^2} & \frac{-2\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & 0 \\
0 & \frac{\Delta t}{\Delta x^2} + 1 & \frac{\Delta t}{\Delta x^2} & -2 \frac{\Delta t}{\Delta x^2} + 1 \\
0 & \frac{\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} + 1 & \frac{\Delta t}{\Delta x^2} \\
0 & \frac{\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} + 1
\end{bmatrix}
\begin{bmatrix}
A_x^n \\
A_i^n \\
A_2^n \\
A_i^n
\end{bmatrix} + \begin{bmatrix}
B \\
B \\
B \\
B
\end{bmatrix}
\]
Poisson's

For

\[
\frac{\partial^2 A}{\partial x^2} + B = 0
\]

\[
\forall i \quad 0 = \frac{1}{\Delta x^2} \left[ A_{i+1} - 2A_i + A_{i-1} \right] + B_i
\]

Or

\[
\begin{bmatrix}
\frac{1}{\Delta x^2} & -2 & \frac{1}{\Delta x^2} \\
0 & \frac{1}{\Delta x^2} & 0 \\
0 & 0 & \frac{1}{\Delta x^2}
\end{bmatrix}
\begin{bmatrix}
A_0 \\
A_1 \\
\vdots \\
A_i \\
\vdots \\
B_i
\end{bmatrix}
= 
\begin{bmatrix}
B_0 \\
B_1 \\
\vdots \\
B_i
\end{bmatrix}
\]

\[
A \times x = b
\]
2-D case

\[ \frac{\partial A}{\partial T} = \frac{\partial^2 A}{\partial x^2} + \frac{\partial^2 A}{\partial y^2} + B \]

\( \Delta S \)

\[ \frac{A_{i,j}^{n+1} - A_{i,j}^n}{\Delta t} = \frac{1}{\Delta s^2} \left[ A_{i+1,j}^n + A_{i-1,j}^n + A_{i,j+1}^n + A_{i,j-1}^n - 4A_{i,j}^n \right] + B_{i,j} \]

\[ A_{i,j}^{n+1} = \frac{\Delta t}{\Delta s^2} \left[ A_{i+1,j}^n + A_{i-1,j}^n + A_{i,j+1}^n + A_{i,j-1}^n - 4A_{i,j}^n \right] + B_{i,j} \Delta t + A_{i,j}^n \]

\[ \begin{bmatrix} A_{11} & A_{12} & \cdots \\
A_{21} & A_{22} & \cdots \\
\end{bmatrix} \]

\[ \begin{bmatrix} A_{i,j}^{n+1} \end{bmatrix} = \begin{bmatrix} ? \end{bmatrix} \begin{bmatrix} A_{i,j}^n \end{bmatrix} + \begin{bmatrix} B_{i,j} \end{bmatrix} \]

° What is the form of this matrix?
Current status

- We saw how to set up a system of equations
- How to solve them
- Poisson: Basic idea

Iterative
Direct
Jacobi, ...
Multigrid...

\[ 0 = \frac{1}{\Delta s^2} \left[ A_{i+1,j} + A_{i-1,j} + A_{i,j+1} + A_{i,j-1} - 4 A_{i,j} \right] + B_{i,j} \]

Or

\[ A_{i,j} = \frac{A_{i+1,j} + A_{i-1,j} + A_{i,j+1} + A_{i,j-1}}{4} + C_{i,j} \]

- In iterative methods

\[ A_{i,j}^{k+1} = \frac{A_{i+1,j}^k + A_{i-1,j}^k + A_{i,j+1}^k + A_{i,j-1}^k}{4} = C_{i,j} \]

- Iterate till no difference
- The ultimate parallel method
In Matrix notation  \( Ax = b \)

- Set up a system of equations.
- Now, solve
  - Direct:
  - Iterative: Solve \( Ax=b \) directly

\[
\begin{align*}
Ax &= b \\
    &= -Ax + b \\
Mx &= Mx - Ax + b \\
Mx &= (M - A)x + b \\
Mx_{k+1} &= (M - A)x_k + b \\
\end{align*}
\]

Solve iteratively

- Direct methods
  - Gaussian elim.
  - Recursive dbl.
  - Semi-direct - CG
- Iterative
  - Jacobi
  - MG
Machine model

- Data is distributed among memories (ignore initial I/O costs)
- Communication over network-explicit
- Processor can compute only on data in local memory. To effect communication, processor sends data to other node (writes into other memory).
Summary

- Many types of parallel applications
  - Attempt to specify as classes (graph, particle, continuum)

- We examine continuum problems as a series of finite differences

- Partition in space and time

- Distribute computation to processors

- Understand processing and communication tradeoffs