ECE 669

Parallel Computer Architecture

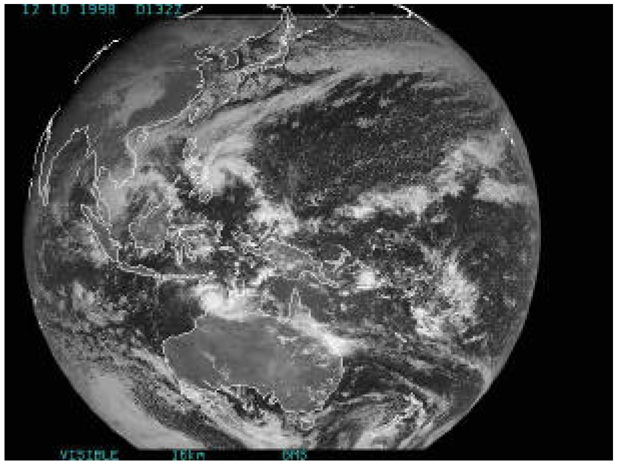
Lecture 4

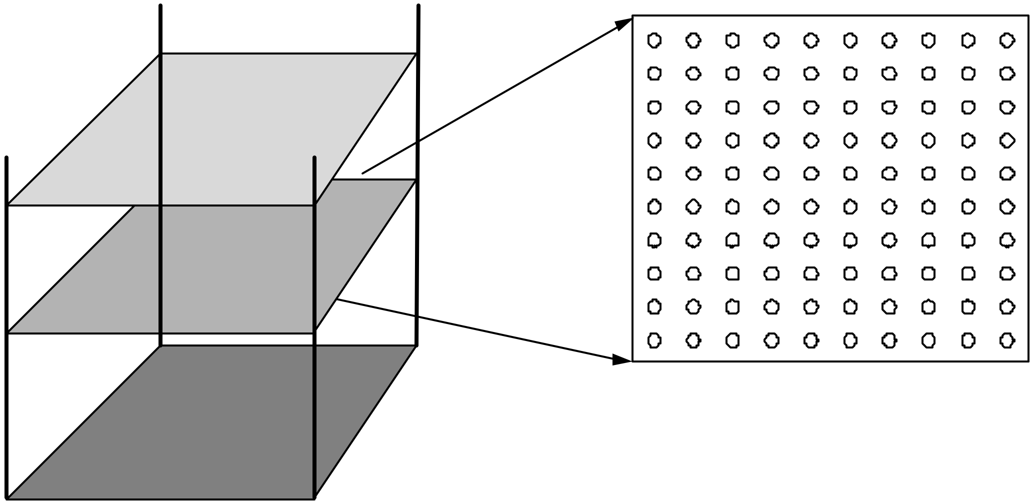
Parallel Applications

Outline

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

Simulating Ocean Currents





(a) Cross sections

(b) Spatial discretization of a cross section

- **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**

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**

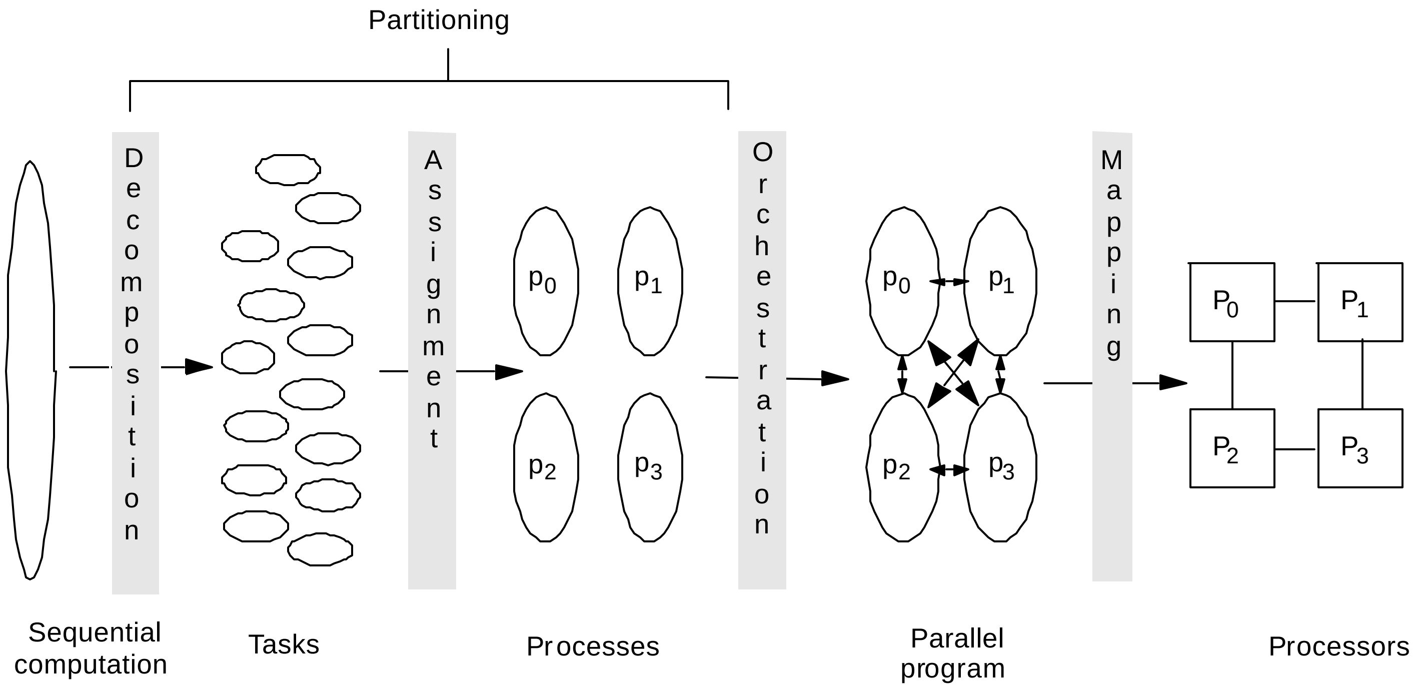
° **Simplification:**

- **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**

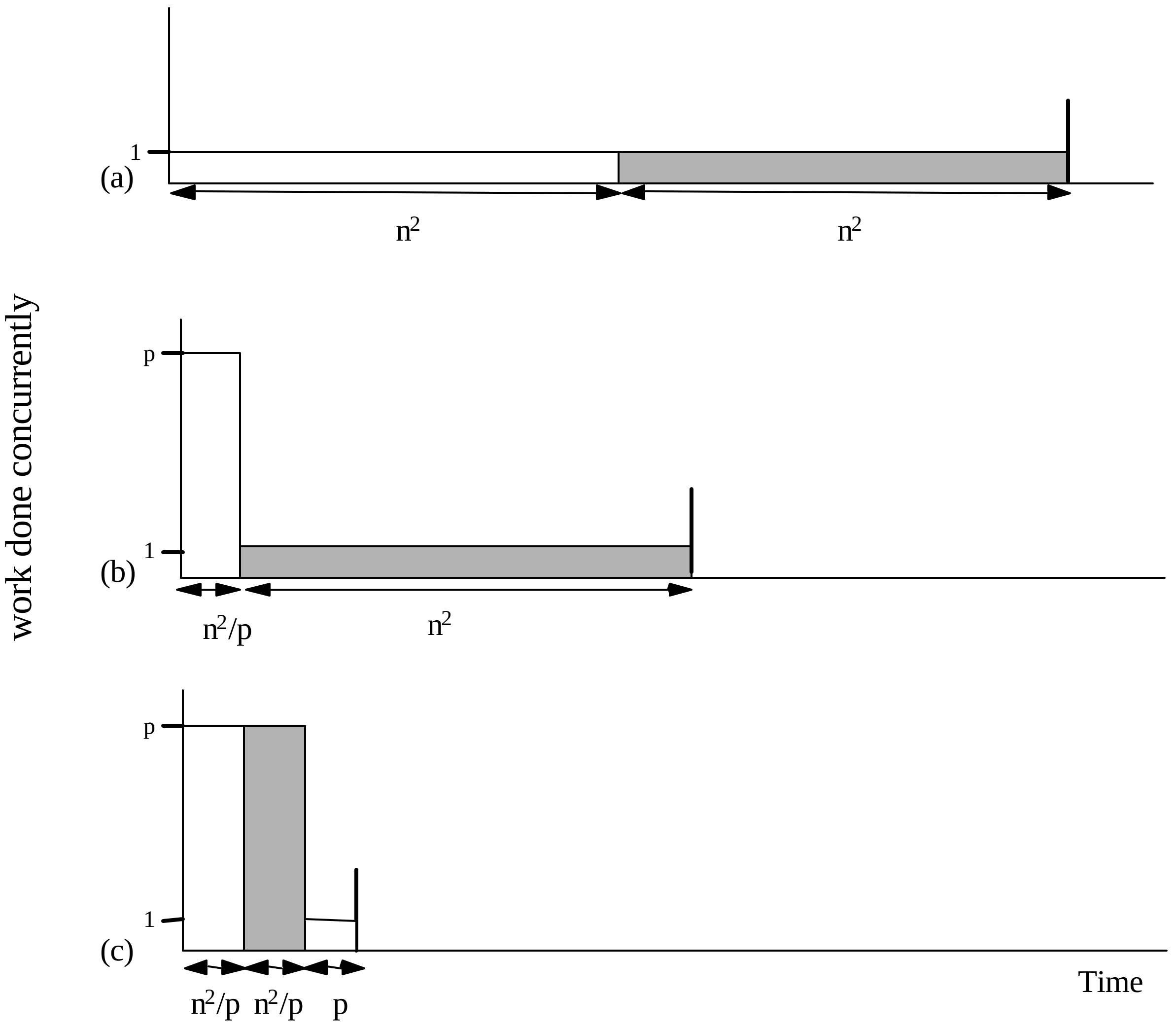
Decomposition

- 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

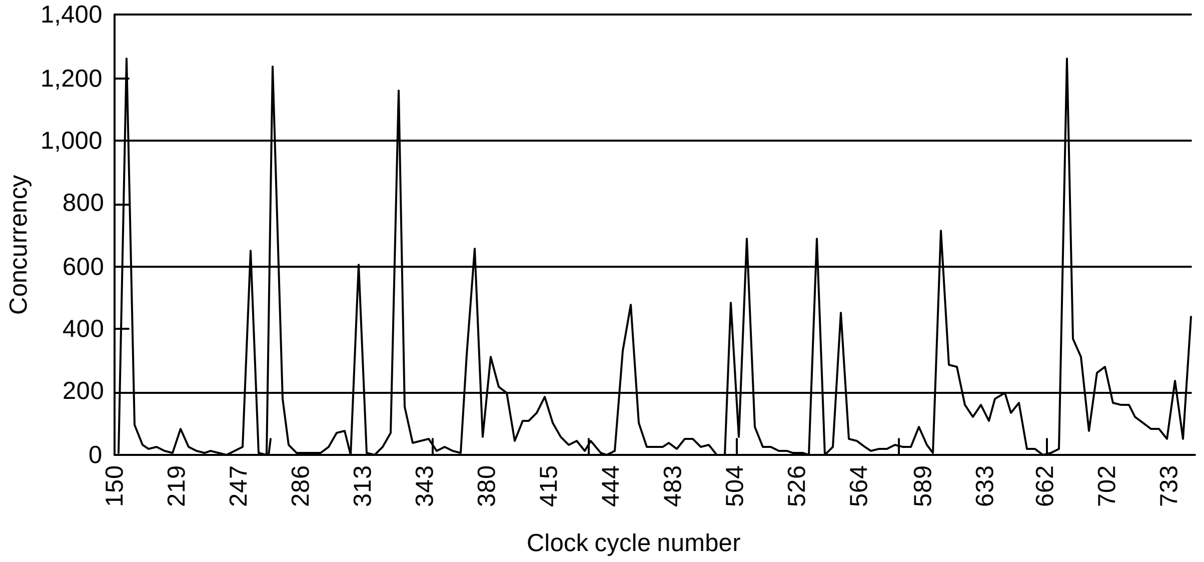
Limited Concurrency: Amdahl's Law

- Most fundamental limitation on parallel speedup
- If fraction *s* of seq execution is inherently serial,
  speedup <= *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 = n^2/p
- Second phase serialized at global variable, so time = n^2
- Speedup <= $\dfrac{2n^2}{\frac{n^2}{p} + n^2}$ 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 $n^2/p + n2/p$ + p, and speedup at best $\dfrac{2n^2}{2n^2 + p^2}$

Understanding Amdahl's Law



Concurrency 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:** $\dfrac{\sum_{k=1}^{\infty} f_k k}{\sum_{k=1}^{\infty} f_k \left\lceil \frac{k}{p} \right\rceil}$**, base case:** $\dfrac{1}{s + \frac{1-s}{p}}$
- **Amdahl's law applies to any overhead, not just limited concurrency**

Applications

- Classes of problems
  - Continuum
  - Particle
  - Graph, Combinatorial

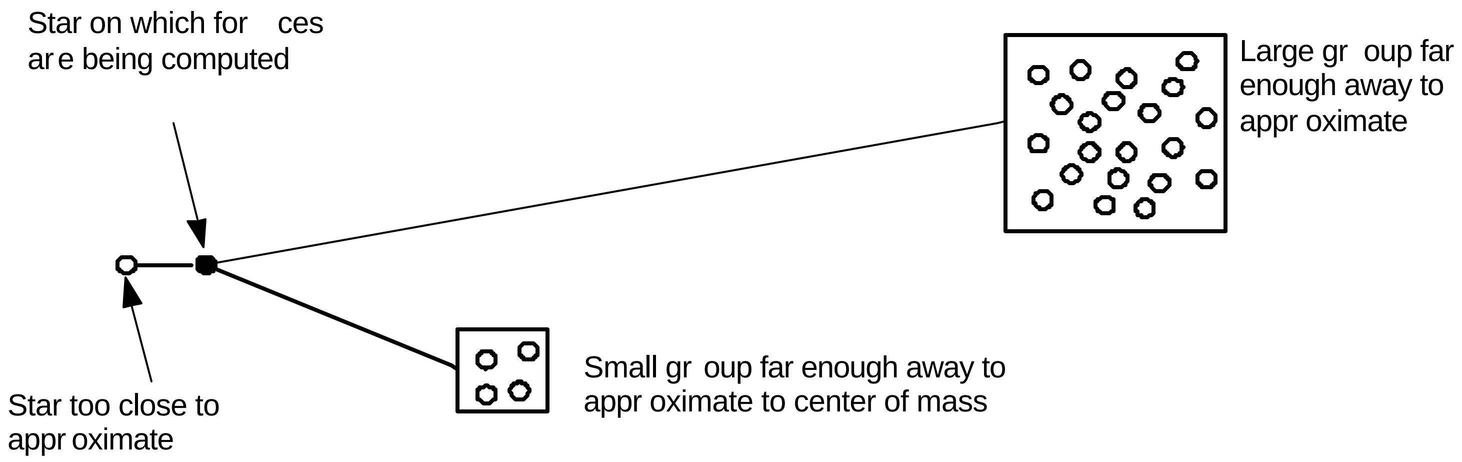
- Goal: Demystifying
- Differential equations ---> Parallel Program

Particle Problems

° **Simulate the interactions of many particles evolving over time**

° **Computing forces is expensive**

- **Locality**
- **Methods take advantage of force law:** $G\frac{m_1 m_2}{r^2}$



Star on which for ces ar e being computed
Large gr oup far enough away to appr oximate
Small gr oup far enough away to appr oximate to center of mass
Star too close to appr oximate

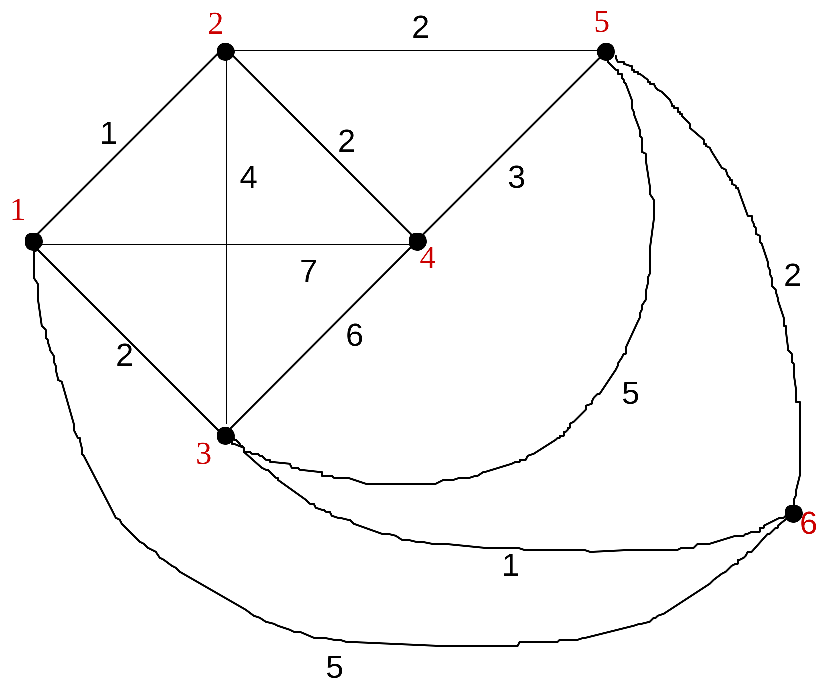
•Many time-steps, plenty of concurrency across stars within one

Graph problems

- Traveling salesman
- Network flow
- Dynamic programming

° **Searching, sorting, lists,**

° **Generally unstructured**



Continuous systems

- **Hyperbolic** $$\frac{\P^2 A}{C^2 \P T^2} = \nabla^2 A + B$$
- **Parabolic** $$\frac{\P A}{C \P T} = \nabla^2 A + B$$
- **Elliptic** $$0 = \nabla^2 A + B$$

  Laplace: B is zero
  Poisson: B is non-zero

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

Numerical solutions

finite	difference	methods
finite	element	methods
	⋮	

Result in system of equations

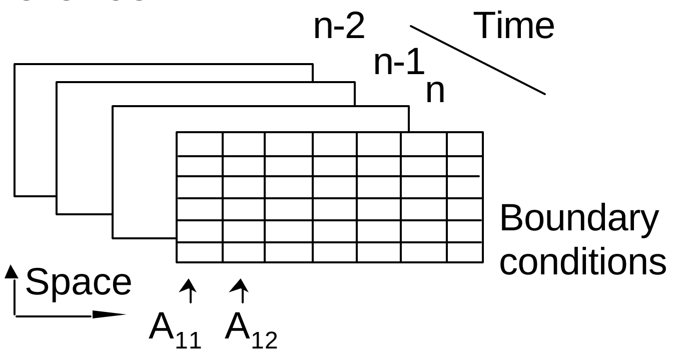
- Let’s do finite difference first

$$\text{Eg.} \quad \frac{\P A}{\P T} = \frac{\P^{2} A}{\P x^{2}}$$

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

Discretize

- **Time** $$\frac{\P A}{\P T} = \frac{A^{n+1} - A^{n}}{\Delta t}$$ **Forward difference**
  - **Where** $$\Delta t = \frac{1}{T \text{ steps}}$$



- **Space**
- **1st** $$\frac{\P A}{\P x} = \frac{A_{i+1} - A_{i}}{\Delta x}$$
  - **Where** $$\Delta x = \frac{1}{X \text{ grid points}}$$
- **2nd** $$\frac{\P}{\P x}\left(\frac{\P A}{\P x}\right) = \frac{(A_{i+1} - A_{i}) - (A_{i} - A_{i-1})}{\Delta x^{2}}$$

$$\frac{\P^{2} A}{\P x^{2}} = \frac{A_{i+1} - 2A_{i} + A_{i-1}}{\Delta x^{2}}$$

  - **Can use other discretizations**
    - **Backward**
    - **Leap frog**

1D Case

$$\frac{\P A}{\P T} = \frac{\P^{2} A}{\P x^{2}} + B$$

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

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

$$\begin{bmatrix} A_x^{n+1} \\ A_i^{n+1} \\ \vdots \\ A_2^{n+1} \\ A_i^{n+1} \end{bmatrix} = \begin{bmatrix} \ddots & \ddots & \ddots & & 0 \\ & \frac{\Delta t}{\Delta x^{2}} & \frac{-2\Delta t}{\Delta x^{2}} + 1 & \frac{\Delta t}{\Delta x^{2}} & \\ 0 & & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} A_x^{n} \\ A_i \\ \vdots \\ A_2^{n} \\ A_i^{n} \end{bmatrix} + \begin{bmatrix} \vdots \\ B \\ \vdots \end{bmatrix}$$

Poisson’s

For

$$\frac{\P^2 A}{\P x^2} + B = 0$$

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

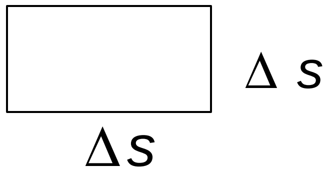
$$\begin{bmatrix} \ddots & \ddots & \ddots & & \mathbf{0} \\ & \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & \\ \mathbf{0} & & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ Ai \\ \vdots \end{bmatrix} = \begin{bmatrix} B_0 \\ B_1 \\ \vdots \\ Bi \\ \vdots \end{bmatrix}$$

Or

$$\mathbf{A} \quad \mathbf{x} \quad = \mathbf{b}$$

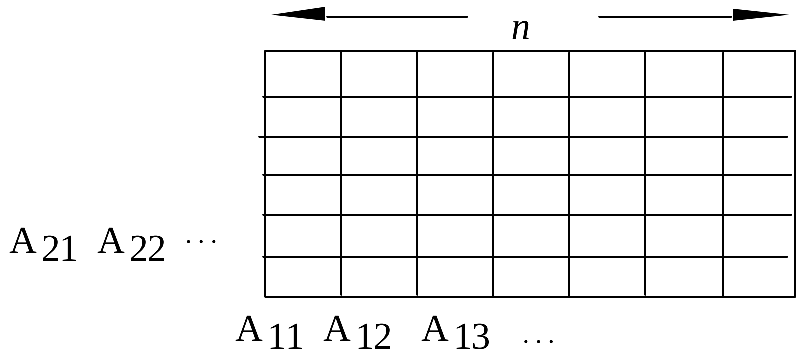
2-D case

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



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

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



$$\left[A_{i,j}^{n+1}\right] = \left[\ ?\ \right]\left[A_{i,j}^{n}\right] + \left[B_{i,j}\right]$$

- What is the form of this matrix?

Current status

- We saw how to set up a system of equations
- How to solve them

Iterative
Direct

Jacobi, ...
Multigrid...
:

- Poisson: Basic idea

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

Or

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

0 for Laplace

- In iterative methods

$$A^{k+1}_{i,j} = \frac{A^k_{i+1,j} + A^k_{i-1,j} + A^k_{i,j+1} + A^k_{i,j-1}}{4} = C_{i,j}$$

  - Iterate till no difference
  - The ultimate parallel method

In Matrix notation $Ax = b$

- Set up a system of equations.
- Now, solve
- Direct:
- Iterative:

Direct methods
Semi-direct - CG
Iterative

Gaussian elim.
Recursive dbl.
Jacobi
MG

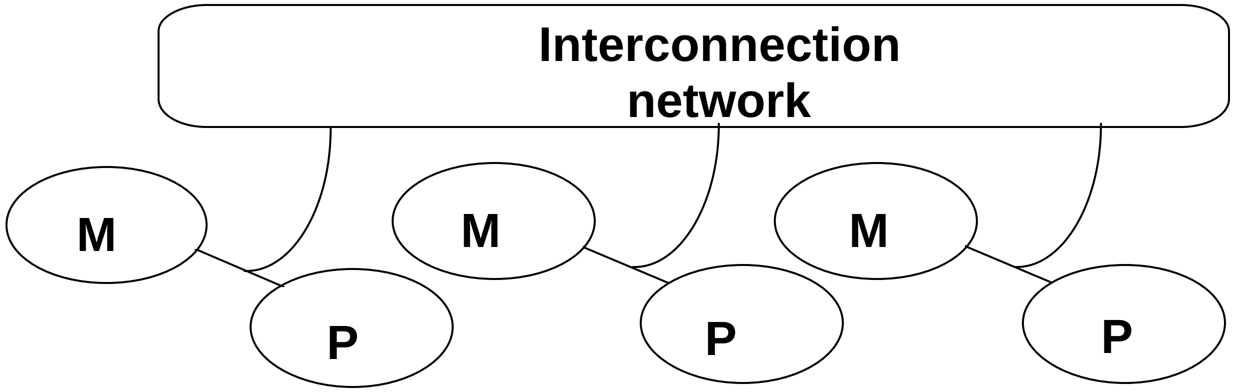
Solve Ax=b directly   LU

$$
\begin{aligned}
A\mathbf{x} &= b \\
&= -A\mathbf{x}+b \\
M\mathbf{x} &= M\mathbf{x} - A\mathbf{x} + b \\
M\mathbf{x} &= (M - A)\,\mathbf{x} + b
\end{aligned}
$$

$$M\mathbf{x}_{k+1} = (M - A)\,\mathbf{x}_k + b$$

Solve iteratively

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