What is Parallel Computing?

- A mechanism for speeding up computation
- Multiple processes work together to solve a problem
- Multiple processors allow multiple processes to run at the same time (in parallel)

Example: compute sum of 1M numbers with 4 processors - runs 4 times faster!
Interesting Parallel Programs Need Communication

- Summarize or combine results
- Propagate updates across processors
- Distribute work to processors
- Handle boundary conditions

\[ \Sigma + \Sigma + \Sigma + \Sigma = \text{Result} \]
Two Ways to Communicate

• Pass Messages
  – Explicitly send
  – Explicitly receive

• Share Memory
  – Read and write shared variables
    • Data implicitly passed between processes
  – Explicitly control access to shared variables
    • Prevent inconsistent state
    • Prevent race conditions
Message Passing Systems

- TCP/IP
- UDP/IP
- GM (Myrinet)
- VIA
- PVM
- MPI
  - MPI 1.1
  - MPI 2.0
The MPI Interface

- MPI is an *interface* standard
  - Is *not* a specific implementation
  - Does not specify much about processes
- MPI designed for parallel computing
  - Not very good for general purpose messaging
- Two “levels” of implementation:
  - MPI 1.1: basic level
  - MPI 2.0: more advanced features
An MPI Job

- A Job creates $n$ copies of your program (tasks)
- One process stays on the master node to manage IO
- Tasks can send/receive messages to/from the other tasks
Example - Image Smoothing

- Image data is modified to reduce noise
- Each pixel replaced by the average of the 8 surrounding pixels, and itself
Parallel Smoothing

- Image data divided among tasks
- Each task smooths its portion
Border pixels need overlapping data

- Data is required from the other tasks
- Other tasks require data as well
Tasks exchange border data

- Each task sends to the other tasks
- Each task receives from the other tasks
- When more tasks, each exchanges with 8 adjacent neighbors
for (r = 0; r < n; r++)
    for (c = 0; c < c; c++)
    {
        sum = 0;
        for (rm = -1; rm < 2; rm++)
        {
            if (r == 0 && rm == -1 || r == n-1 && rm == 1)
                continue;
            for (cm = -1; cm < 2; cm++)
            {
                if (c == 0 && cm == -1 || c == n-1 && cm == 1)
                    continue;
                sum += input[r+rm][c+cm];
            }
        }
        output[r][c] = sum / 9;
    }
Smoothing Program
Dividing the Data

-1, -1  -1, 0  -1, +1
0, -1  0, 0  0, +1
+1, -1  +1, 0  +1, +1

Copyright © 2004 Walter B. Ligon III, all rights reserved
Border Cells