Announcements

- Project #6 is due Tuesday at 6:00 PM
- Office Hours This Week
  - Tu 11-12
  - W 10-12 (starts at review session in classroom)
  - Th 11-12
- Course Evaluations
  - Please fill them out!
Display and Window Management

- **The screen is a resource in a workstation system**
  - multiple processes desire to access the device and control it
  - OS needs to provide abstractions to permit the interaction

- **Services**
  - protection
  - windows
  - multiplex keyboard and mouse
  - configuration and placement

- **Issues**
  - how to get good performance and remain device independent
  - how much policy to dictate to users
My Research Interests

- **Parallel Computing**
  - There are limits to how fast one processor can run
  - solution: use more than one processor

- **Issues in parallel computing design**
  - do the processors share memory?
    - is the memory “uniform”?
    - how do processors cache memory?
  - if not how do they communicate?
    - message passing
    - what is the latency of message passing
Parallel Processing

- What happens in parallel?
- Several different processing steps
  - pipeline
  - simple example: `grep foo | sort > out`
  - called: *multiple instruction multiple data* (MIMD)
- The same operation
  - every processor runs the same instruction (or no-instruction)
  - called: *single instruction multiple data* (SIMD)
  - good for image processing
- The same program
  - every processor runs the same program, but not “lock step”
  - called: *single program multiple data* (SPMD)
  - most common model
Issues in effective Parallel Computation

- **Getting enough parallelism**
  - Limited by what is left serial
  - Even 10% serial limited to a speedup of 10x even with infinite numbers of processors

- **Load balancing**
  - every processor should to have some work to do.

- **Latency hiding/avoidance**
  - getting data from other processors (or other disks) is slow
  - need to either:
    - hide the latency
      - processes can “pre-fetch” data before they need it
      - block and do something else while waiting
    - avoid the latency
      - use local memory (or cache)
      - use local disk (or file buffer cache)

- **Limit communication bandwidth**
  - use local data
  - use “near” data (i.e. neighbors)
My Research:

- Given a parallel program and a machine
- Try to answer performance related questions
  - Why is the programming running so slowly?
  - How do I fix it?
- Issues:
  - how to measure a program without changing it?
  - how do you find (and then present) the performance problem, not tons of statistics?
- Techniques:
  - dynamic data collection
  - automated search
  - analysis of process interactions
Introduction

- **Software today**
  - makes extensive use of libraries and re-usable components
  - Libraries used by an application may not be tuned to the application’s need

- **Fast software development/distribution with built-in (default) configurations**
  - Applications may not run well in all environments
  - There may be no single configuration good for all environments
Large Scale Computing

- **Today (11/2009)**
  - 5 systems with more than 128k processors
  - 32 systems >= 16k processors
  - World’s fastest computer (jaguar at ornl)
    - 224,000 AMD cores
    - 62 Terrabytes of RAM