Advanced Operating Systems
Lecture notes

Dr. Dongho Kim
Dr. Clifford Neuman
University of Southern California
Information Sciences Institute

CSci555: Advanced Operating Systems

Lecture 12 - November 12, 2004
Scheduling, Real-Time, Fault Tolerance
(slides by Dr. Neuman)

Dr. Dongho Kim
University of Southern California
Information Sciences Institute

Administrative

- Mid-term and assignment grades sent to students this week
  - Contact us if you did not receive email with your grades
- Final exam is Thursday December 11, 11AM
- Research paper due Friday December 5
  - Accepted w/o penalty until December 12
- Please send suggested topics for December 5 lecture to csci555@usc.edu

Scheduling and Real-Time systems

- Scheduling
  - Allocation of resources at a particular point in time to jobs needing those resources, usually according to a defined policy.
- Focus
  - We will focus primarily on the scheduling of processing resources, though similar concepts apply the the scheduling of other resources including network bandwidth, memory, and special devices.

Parallel Computing - General Issues

- Speedup - the final measure of success
  - Parallelism vs Concurrency
    - Actual vs possible by application
  - Granularity
    - Size of the concurrent tasks
    - Reconfigurability
  - Number of processors
  - Communication cost
  - Preemption v. non-preemption
  - Co-scheduling
    - Some things better scheduled together

Shared Memory Multi-Processing

- Includes use of distributed shared memory, and shared memory multi-processors
- Processors usually tightly coupled to memory, often on a shared bus. Programs communicated through shared memory locations.
- For SMPs cache consistency is the important issue. In DSM it is memory coherence.
  - One level higher in the storage hierarchy
- Examples
  - Sequent, Encore Multimax, DEC Firefly, Stanford DASH

Where is the best place for scheduling

- Application is in best position to know its own specific scheduling requirements
  - Which threads run best simultaneously
  - Which are on Critical path
  - But Kernel must make sure all play fairly
- MACH Scheduling
  - Lets process provide hints to discourage running
  - Possible to hand off processor to another thread
    - _Makes easier for Kernel to select next thread
    - _Allow interleaving of concurrent threads
  - Leaves low level scheduling in Kernel
  - Based on higher level info from application space

Scheduler activations

- User level scheduling of threads
  - Application maintains scheduling queue
- Kernel allocates threads to tasks
  - Makes upcall to scheduling code in application when thread is blocked for I/O or preempted
  - Only user level involved if blocked for critical section
- User level will block on kernel calls
  - Kernel returns control to application scheduler

Distributed-Memory Multi-Processing

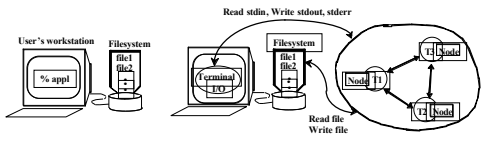
- Processors coupled to only part of the memory
  - Direct access only to their own memory
- Processors interconnected in mesh or network
  - Multiple hops may be necessary
- May support multiple threads per task
- Typical characteristics
  - Higher communication costs
  - Large number of processors
  - Coarser granularity of tasks
- Message passing for communication

Prospero Resource Manager

Prospero Resource Manager - 3 entities

- One or more system managers
  - *Each manages subset of resources*
  - *Allocates resources to jobs as needed*
- A job manager associated with each job
  - *Identifies resource requirements of the job*
  - *Acquires resources from one or more system managers*
  - *Allocates resources to the job's tasks*
- A Node manager on each node
  - *Mediates access to the nodes resources*

The Prospero Resource Manager



A) User invokes an application program on his workstation.

b) The program begins executing on a set of nodes. Tasks perform terminal and file I/O on the user's workstation.

Advantages of the PRM

- Scalability
  - *System manager does not require detailed job information*
  - *Multiple system managers*
- Job manager selected for application
  - *Knows more about job's needs than the system manager*
  - *Alternate job managers useful for debugging, performance tuning*
- Abstraction
  - *Job manager provides a single resource allocator for the job's tasks*
  - *Single system model*

Real time Systems

- Issues are scheduling and interrupts
  - Must complete task by a particular deadline
  - Examples:
    - _Accepting input from real time sensors
    - _Process control applications
    - _Responding to environmental events
- How does one support real time systems
  - If short deadline, often use a dedicated system
  - Give real time tasks absolute priority
  - Do not support virtual memory
    - _Use early binding

Real time Scheduling

- To initiate, must specify
  - Deadline
  - Estimate/upper-bound on resources
- System accepts or rejects
  - If accepted, agrees that it can meet the deadline
  - Places job in calendar, blocking out the resources it will need and planning when the resources will be allocated
- Some systems support priorities
  - But this can violate the RT assumption for already accepted jobs

Fault-Tolerant systems

- Failure probabilities
  - Hierarchical, based on lower level probabilities
  - Failure Trees
  - Add probabilities where any failure affects you
    - Really (1 - ((1 - lambda)(1 -lambda) (1 - lambda)))
  - Multiply probabilities if all must break
    - _Since numbers are small, this reduces failure rate
  - Both failure and repair rate are important

Making systems fault tolerant

- Involves masking failure at higher layers
  - Redundancy
  - Error correcting codes
  - Error detection
- Techniques
  - In hardware
  - Groups of servers or processors execute in parallel and provide hot backups
- Space Shuttle Computer Systems exampls
- RAID example

Types of failures

- Fail stop
  - Signals exception, or detectably does not work
- Returns wrong results
  - Must decide which component failed
- Byzantine
  - Reports difficult results to different participants
  - Intentional attacks may take this form

Recovery

- Repair of modules must be considered
  - Repair time estimates
- Reconfiguration
  - Allows one to run with diminished capacity
  - Improves fault tolerance (from catastrophic failure)

OS Support for Databases

- Example of OS used for particular applications
- End-to-end argument for applications
  - Much of the common services in OS's are optimized for general applications.
  - For DBMS applications, the DBMS might be in a better position to provide the services
    - Caching, Consistency, failure protection