The Lecture Contains:

User Control

- User Control : User has broader control on process characteristics specs in an RTOS
  - Priority
  - Deadlines: Hard or soft.
- Memory Management: paging or swapping
- Name the processes to be resident in memory
- Scheduling policies

Reliability

- Reliability: A processor failure in a non-RT may result in reduced level of service. But in an RT it may be catastrophic : life and death, financial loss, equipment damage.
- Fail-soft Operation: Ability of the system to fail in such a way preserve as much capability and data as possible.
- In the event of a failure, immediate detection and correction is important.
- Notify user processes to rollback.
- Apply compensation.
- Check-pointing and rollback states.

Previous Next

Requirements of RT

- Fast context switch
- Minimal functionality (small size)
- Ability to respond to interrupts quickly (Special interrupts handlers)
- Multitasking with signals and alarms
- Special storage to accumulate data fast
- Preemptive scheduling

Requirements of RT (contd.)

- Priority levels
- Minimizing “interrupt disabled” state
- Short-term scheduler (“omni-potent”)
- Time monitor
- Goal: Complete all hard real-time tasks by dead-line. Complete as many soft real-time tasks as possible by their deadline.

Previous Next

Multi-processor Scheduling

- Load sharing
- Each processor runs one process.
- A single ready queue is maintained (in a shared memory).
- Each processor makes scheduling decisions independently.
- Speed up of an application
- Processor assignment (within threads of the process): Dynamic or fixed
- Scheduling: Gang scheduling or independent scheduling

Multi-core Computing Multi-processor Scheduling

Introduction

- Design issues in system with multiple processors
    - Tightly coupled or loosely coupled.
    - Message passing, shared memory or both.
- Several new issues are introduced into the design of scheduling functions.
- Loosely coupled systems are easier to handle.
    - Not an OS issue but an application issue
    - Several message passing systems are available (such as MPI)
- We will examine these issues and the details of scheduling algorithms for tightly coupled multi-processor systems.

Previous Next

Module 20: Multi-core Computing Multi-processor Scheduling

Lecture 39: Multi-processor Scheduling

Issues With Multi-processor Computations.

- Granularity of computation
  - Fine grain
  - Coarse grain
    - Various shades of coarseness.
- Design issues
  - Assignment of processes to processors
  - Multiprogramming on individual processors
  - Actual dispatching of a process

Granularity

- The main purpose of having multiple processors is to realize ????.
- Applications exhibit parallelism at various levels with varying degree of granularity.
- Fine grain parallelism: Inherent parallelism within a single instruction stream.
  - High data dependency → high frequency of synchronizations required between processes.
  - Dynamic Scheduling/Superscalar architectures exploit this kind of parallelism.

Previous Next

Module 20: Multi-core Computing Multi-processor Scheduling

Lecture 39: Multi-processor Scheduling

Fine Grain Parallelism

- Given a sequence of instructions the issue logic will look at the dependencies between multiple instructions in a window.
    - Dependencies: Data or Control dependency.
    - Dependencies: False or true.
    - Consider add R1, R2 followed by add R3, R4.
    - Consider div R1, R2 followed by div R3, R4
    - Consider load X followed by load Y
- Independent instructions can be executed in parallel if structural resources are available.

Granularity

- Medium Grain Parallelism: Parallelism of an application can be implemented by multiple threads in a single process.
- Usually programmers have to “use” threads in the design.
- Threads are scheduled by user (e.g. pthreads) or by OS (kernel threading).
- Coarse Grain Parallelism: Parallelism in a system by virtue of several concurrent processes
- Need to synchronize using semaphore or other synchronization objects.
- Example: Server side threads for web servers. FTP servers etc.

Previous Next

Granularity

- Very Coarse Grain: When synchronization needs are not high among parallel processes.
  - Processes can even be distributed across network.
    - Example: CORBA standard for distributed system.
- Independent parallelism: Multiple unrelated processes.
  - All independent processes can run in parallel.

Design Issues

- Where does the OS run?
- Mapping processes to processors.
  - Static or dynamic
- Use of multiprogramming on individual processors.
- Actual dispatching of a process.

Previous Next

Where Does The OS Run?

- Master/slave Assignment: Kernel functions always run on a particular processor. Other processors execute user processes.
- Advantage: Resource conflict resolution simplified since single processor has control.
- Single point of failure (Master).
- Peer Assignment: OS executes on all processors. Each processor does its own scheduling from the pool of available processes. Most OSes implement this in an SMP environment.

Mapping Processes to Processors

- Largely an application issue.
- Applications may be written for a particular configuration of the machine.
- Application create threads of computations and channel of communication between these threads assuming a particular machine configuration.
- Applications may be generic in nature
- Create n threads of computations. These threads can execute on machines with 1, <n, n or >n processors.
- OS may assign any number of processors (1 to n) to this application.

Previous Next

Module 20: Multi-core Computing Multi-processor Scheduling

Lecture 39: Multi-processor Scheduling

Multiprogramming at Each Processor

- In a multi-processor system, CPU utilization is not all that important
  - As long as some computation is being carried out, it is fine.
- Application efficiency are more important
  - Turn around time
  - application-related performance metrics.
- Assigning all threads to different processors may not always yield high performance.

Multiprogramming

- A multi-threaded application may require all its threads be assigned to different processors for good performance.
  - If there are n threads and m processors (m > n), m-n processors may not have any thing to execute.
- If threads are dependent on each other and require heavy synchronization, assigning all to same processor may be beneficial
- Process allocation can be Static or dynamic.

Previous Next