WEEK 7

Scheduling: SMP and real-time I/O management

Ch 10 [Stal 05]
(Ch 20 [DDC04], 11.4 [Tane01])
Ch 11.1 – 11.5

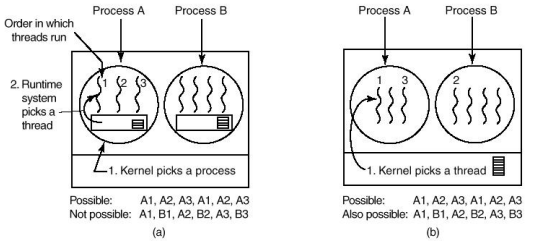
1

SMP scheduling

- Master-slave
  - OS on the master
  - Slaves execute user processes
  - What if master fails?
- SMP (Peer architecture)
  - OS running on any CPU even simultaneously
    - re-entrant code
    - concurrenry and synchronisation
    - fault handling?
  - Each CPU makes scheduling decisions
  - Shared READY-queue vs private ready-queues

2

Thread scheduling



(Fig 2-43 [Tane01])

- **ULT: OS dispatches processes, thread library handles threads -> one process per processor**
- **KLT: OS dispatches threads -> one thread per processor**

3

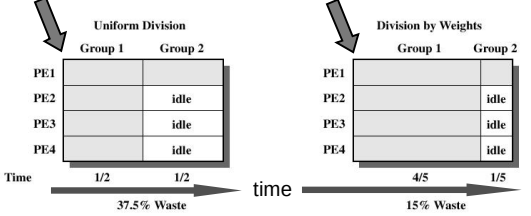
Load sharing

- No processor idle, if there are processes/ threads to execute
- Threads can be run on any CPU
  - Threads of one process scheduled independently
- Global Ready-queue
  - Mutual exclusion
  - Bottle neck?
- Each CPU runs the scheduling routine
  - Any scheduling mechanisms is possible, for example FCFS
- Most commonly used

4

Gang scheduling

- Threads/processes of one gang run at the same time on multiple CPUs - only one scheduling decision needed!
- Reduces synchronisation blocking
- Less process switches



(Fig 10.2 [Stal05])

Dedicated CPU assignment

- Part (or all) of the CPUs reserved for one application
  - Reservation ends only when the process ends
- If a thread *blocked*, CPU is still allocated and idle
- Advantages?
  - Avoid process switching, support priorities
  - Cache utilisation
- Others may suffer
  - Must have one or few CPUs for the others
- Differences with gang scheduling?
  - Here: no scheduling, no process switches
  - Thread always on the same processor

6

Dynamic scheduling

- Somewhere between CPU allocation and gang scheduling
- *"First give what is available, and maybe later a bit more"*
- New process
  - If enough free CPUs, allocate them
  - If not enough available
    - deallocate CPU from another process
    - requesting process must wait or
    - cancel the request
- When CPU is released
  - Allocate to a process waiting for its first CPU
  - Otherwise, allocate on a FCFS basis.

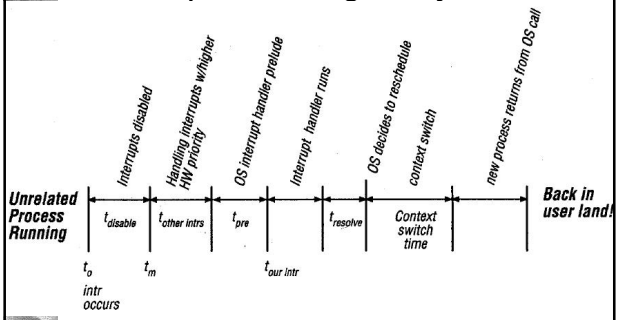
7

Real-time systems

- In addition to the correct results, they must be reached within a given time limit (not too early, not too late)
- Reacts to external events (and time)
- Terms:
  - Hard Real-time vs. Soft Real-time
  - Periodic vs aperiodic
  - Static vs dynamic scheduling
- Requirements:
  - Configurable, deterministic, reliablity, responsiveness,
  - Programmers decide characteristics of the rt processes

8

Interrupt handling delays



lähde?

9

RT scheduling

- Timely execution the most important issue
  - No fairness, no minimising the response time
- Os can only guess the actual execution time
  - Programmer gives some estimates, period gives hints
- Quite often: just schedule rt-tasks fast
  - High priority, predetermined schedule plan, others can suffer
- Two main alternative for dynamic scheduling
  - Earliest Deadline First (EDF)
  - Rate Monotonic Scheduling (RM)

10

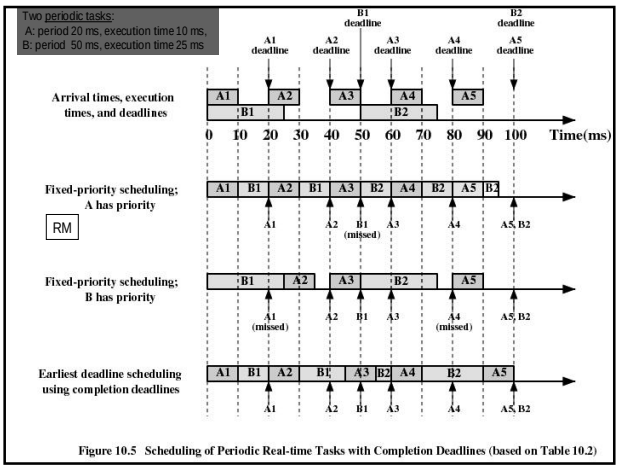


Figure 10.5 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)

Schedulability test

Tbl 10.4 [Stal05]

- Obviously for any schedule, the schedulable utilisation:

$$U_i = C_1/T_1 + C_2/T_2 + .. + C_n/T_n <= 1$$

- Sufficient condition for schedulability using RM scheduling

$$C_1/T_1 + C_2/T_2 + .. + C_n/T_n <= n(2^{1/n} - 1)$$

- N is number of tasks, the upper limit for the value is

$$\ln 2 \sim 0.693 \quad \text{-> for any i } U_i < 0.693$$

- For EDF the sufficient and effective condition is

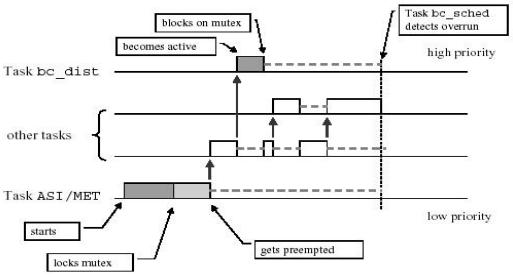
$$U_i = C_1/T_1 + C_2/T_2 + .. + C_n/T_n <= 1$$

What can go wrong ?

- Higher priority task may have to wait !
- … for a shared resource
  - Priority inversion, deadlocks,
- Solution:
  - Non-preemptable (uninterruptible) critical sections
  - Priority inheritance
  - Priority ceiling protocol

13

Priority inversion on Mars Pathfinder (1997)



•http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html

14

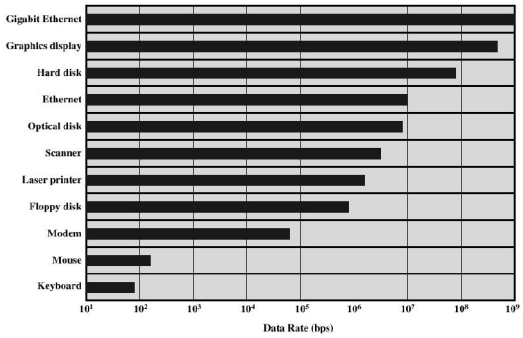
LUENTO 14

I/O MANAGEMENT

Stallings, Luku

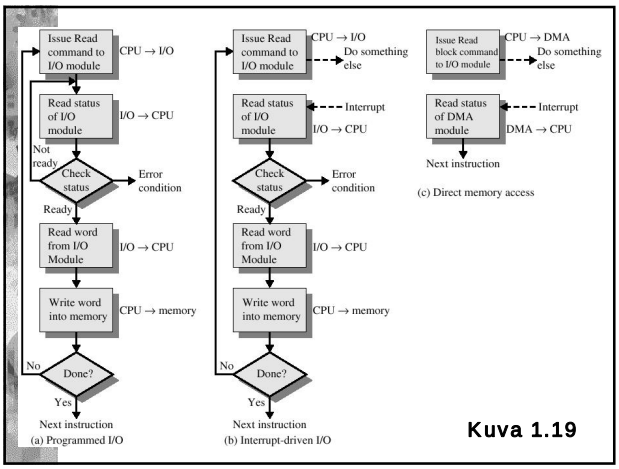
15

Theoretical speeds Kuva 11.1



NOTE: Logarithmic scale

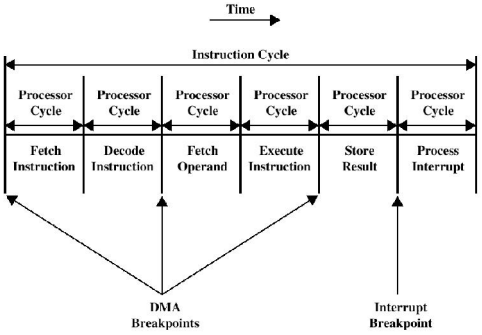
16



Kuva 1.19

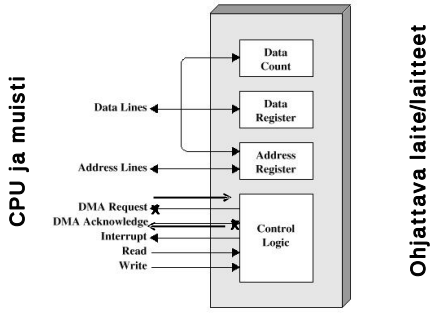
DMA & Bus usage

Kuva 11.3 [Stal01]

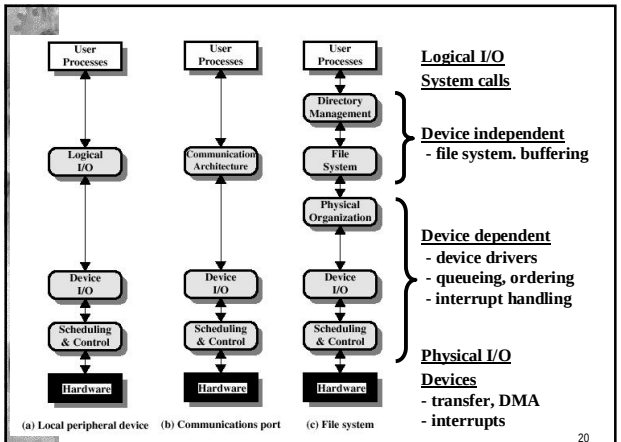


18

DMA-controller Kuva 11.2



19



20

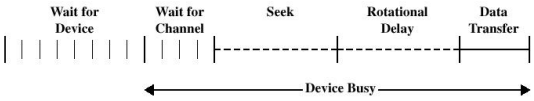
Kuva 11.4

Buffering

- No buffering
  - Copied directly to the process's memory area, that cannot be swapped out
  - Process need to know the size
- Single buffer
  - Device controller copies to OS memory and OS copies to the process memory -> process can be fully swapped out
  - Read ahead, delayed write
- Doubel buffer
  - Device uses the other buffer and the process other, once done switch
  - Generic model: Producer – buffer – consumer
- Ring buffer
  - Covers variation of speeds between producer and consumer

21

Disk access delays & scheduling



(Fig 11.6 [Stal05])

- Random? FIFO? PRI? LIFO?
  - Not good, no usage of current position of the arm
- Considers the position
  - SSTF
  - SCAN
  - C-SCAN
  - N-step-SCAN ja FSCAN

22

Disk scheduling

Table 11.2 Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	Average seek length	27.5	Average seek length	27.8	Average seek length	35.8

23

Logical sectors

- Sequential sector numbers do not mean that the sectors will be sequentially on the physical disk
  - There are some extra sectors for fault detection and avoidance
  - faulty sector 123? Use a replacement sector 9876 with the loogical sector number remaining still 123
  - Not visible outside the disk and its controller
    - device driver does not know about them!
    - Thus the optimisation done by the driver (or kernel) may be based on faulty information
- On some disks the controller does the optimisation (reordering of the requests)
  - Knows the logical sectors and their physical location
  - The driver should not optimise in this case
- On some disks the driver can ask about the actual physical location

24