Real-Time Filesystems Guaranteeing Timing Constraints for Disk Accesses in RT-Mach

Anastasio Molano, Kanaka Juvva and Ragunathan Rajkumar
Carnegie Mellon University

Rak-Ki Kim
DMC Lab.

Contents

- Introduction
- Comparison with Related Work
- A Brief Overview of RT-Mach

Introduction

- traditional real-time systems have largely avoided the use of disks.
  - disk accesses can introduce unpredictable latencies
  - disk access requests must also be managed in conjunction with processor scheduling
- we extend the processor reservation model in RT-Mach to include guaranteed and timely access to disk bandwidth.
- the application is guaranteed to obtain its requested share of the disk bandwidth

3

Comparison with Related Work

- we use dynamic priority scheduling, exploit blocking instead of minimizing it and evaluate an implementation within our resource framework.
- Our scheme employs a different period for each real-time activity, and guarantees a "share" of that period to the activity.

4

Comparison with Related Work(cont'd)

- we add real-time filesystem support to RT-Mach in a way that is completely compatible with its resource reservation model.

5

A Brief Overview of RT-Mach

- RT-Mach
  - task scheduling
  - Virtual memory management
  - synchronization mechanisms
  - Interprocess communications
  - real-time disk scheduling
  - Network protocol processing
  - distributed coordination.

6

A Brief Overview of RT-Mach(cont'd)

- RT-Mach adopts the resource reservation model, which allows applications to specify their resource demands independent of the scheduling algorithm actually used in the kernel.

7

Resouce Kernels

- A resource kernel is defined to be one which provides timely, guaranteed and protected access to system resources.
- Resourcecentric approach can be implemented with any of several different resource management schemes

8

Resouce Kernels(cont'd)

- Resource kernel
  - apply a uniform resource model for dynamic sharing of different resource types
  - take resource usage specifications from applications
  - guarantee resource allocations at admission time
  - schedule contending activities on a resource based on a well-defined scheme
  - ensure timeliness by dynamically monitoring and enforcing actual resource usage.

9

Disk Bandwidth Resource Management

- We then improve the algorithm by exploiting "slack" in the reservations to obtain a hybrid of earliest deadline scheduling and a traditional scan algorithm.

10

Important Considerations

- Preemptibility issues
  - The duration of the non-preemption window must ideally be small and perhaps even dynamically adjustable depending on the workload.
- With Preemption
  - a higher priority disk request can preempt a lower priority disk request midway through the processing of its larger request.

11

Important Considerations(cont'd)

- Heterogeneity of the workload
  - many small requests with deadlines, but they are prevented from execution due to larger low priority disk requests.

12

Admission Control

- Our simplest disk head scheduling scheme employs the earliest deadline scheduling algorithm
- An ongoing disk block access must complete before the next highest priority disk block access request can be issued.
- This introduces a blocking factor of a single filesystem block access when Di = Ti.
- When Di < Ti, the required earlier completion time (Ti − Di) is added to the blocking factor.

Admission Control (cont'd)

- : $BU_{max} + \sum_{i=1}^{n} U_i \leq 1.0$
- : $U_i = \frac{t_{IPC-in} + t_{FSB} L_i + 2t_{Seek} + 2t_{Rot} + t_{IPC-out}}{T_i}$
- : $t_{FSB} = t_{fs-overhead} + \frac{\text{size of FSB in bytes}}{\text{disk BW in bytes/sec}}$
- : $BU_{max} = max\left(\frac{B_1}{T_1}, \frac{B_2}{T_2}, \cdots, \frac{B_n}{T_n}\right)$
- : $B_i = t_{FSB} + T_i - D_i$

Admission Control (cont'd)

- A real-time filesystem can allocate disk blocks as contiguously as possible.
- Utilities can be used to choose the block allocation policy within a filesystem and/or relocate file blocks so that they are contiguous.
- With contiguous layout, our current admission control test guarantees the requested bandwidth if the test succeeds.

15

Just-In-Time Disk Scheduling

- a scheduling algorithm can exploit the slack available to higher priority tasks to schedule accesses of other disk blocks which are closer to the current head position.
- If slack is stolen, the slack of higher priority reservations is reduced by one.
- If the slack of a high priority reservation goes to zero, it will be serviced independent of its location.

16

"Just-in-Time" Slack-Stealing Algorithm for Disk Bandwidth

- $K_i(t)$: the maximum "slack" in units of time available for each reserved disk data stream
- K_i: the maximum slack available in units of disk blocks.

$$K_i(t) = D_i - T_i \sum_{j=1}^{i} (U_j)$$

$$K_i = \lfloor \frac{K_i(t)}{t_{FSB}} \rfloor$$

The JIT Replenishment Algorithm

- Usage by an unreserved activity:
  - $\forall j,\ j \in \{J\},\ k_j > 0.$
  - $\forall j,\ j \in \{J\},\ k_j \leftarrow (k_j - 1)$
- Stealing by a depleted reservation:
  - $\forall j,\ j \in \{J\},\ k_j > 0.$
  - $\forall j,\ j \in \{J\},\ k_j \leftarrow (k_j - 1)$
- Usage by an unreserved activity:
  - $\forall j, (j \in \{J\}) \wedge (j \leq (i-1)),\ k_j > 0$
  - $\forall j, (j \in \{J\}) \wedge (j \leq (i-1)),\ k_j \leftarrow (k_j - 1).$

Disk Bandwidth Reservation in Real-Time Mach

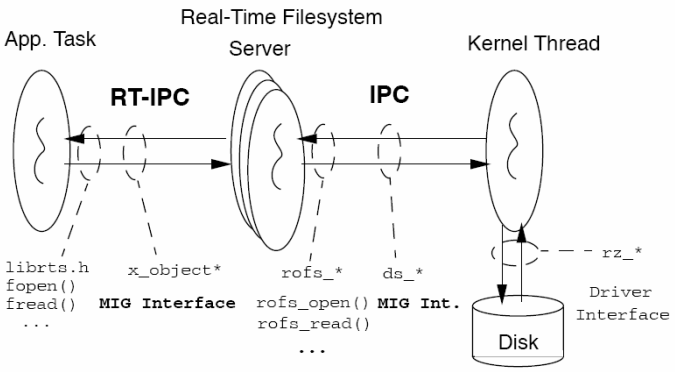
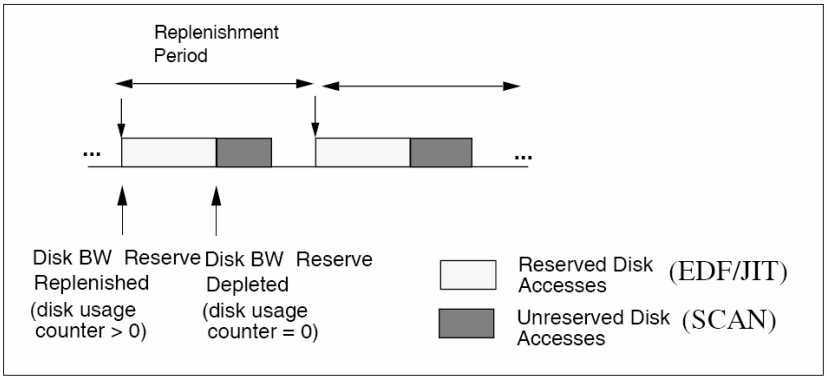


Figure 3-1: Real-Time File System Service Layers

The JIT Replenishment Algorithm(cont'd)

- The file system reserve scheduling mechanism manages all the invoking disk I/O requests.
- The scheduler within the RT filesystem has been implemented as a cooperative scheduler.
- The file system is executed in a loop, sending a command to the disk controller each iteration to read/write a file system block.
- It executes the scheduling algorithm (EDF, EDF/JIT,SCAN) choosing the next request based on the current scheme.

The JIT Replenishment Algorithm(cont'd)



Performance Evaluation

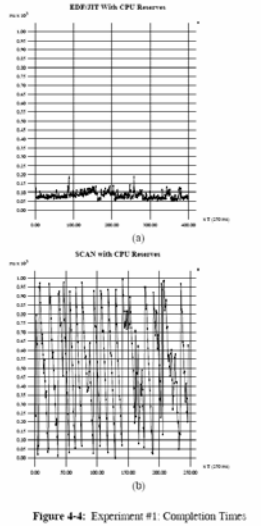


Figure 4-4: Experiment #1: Completion Times

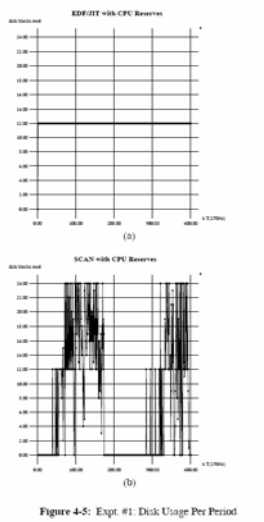


Figure 4-5: Expt. #1: Disk Usage Per Period

Performance Evaluation(cont'd)

Thread and the number ofdisk blocks read	EDF with CPU Reserves	Scan with CPU Reserves	EDF without CPU Reserves	Scan without CPU Reserves
RT-Thread (reserved)	4788	2975	4800	2980
Thread 1	2100	2267	2100	2681
Thread 2	478	770	510	822
Thread 3	2355	4410	2190	4410
Thread 4	2088	2061	2107	2106
Thread 5	1824	1482	1862	1482
Thread 6	1172	600	1169	650
Looping Thread	2340	3428	2743	3428
Total (blocks read)	**17,945**	**17,993**	**17,481**	**18,559**
% of Scan	96.69	96.95	94.19	100
% degradation relative to Scan	**3.31**	**3.05**	**5.81**	**0**

Table 4-1: Expt. #1: Total Disk Usage of each Thread

Performance Evaluation(cont'd)

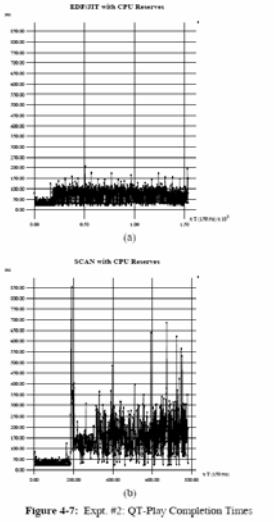


Figure 4-7: Expt. #2: QT-Play Completion Times

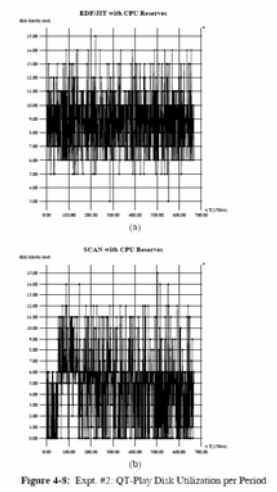


Figure 4-8: Expt. #2: QT-Play Disk Utilization per Period

Performance Evaluation(cont'd)

Test	Scan (total disk blocks read)	EDF/JIT (total disk blocks read)	EDF/JIT Performance Penalty (%) (Scan - EDF) / Scan
1	14951	12521	16.25%
2	13468	11029	18.1%
3	13704	9797	28.5%
4	14879	12601	15.31%
5	15115	13827	8.52%

Table 4-2: Total Throughput Comparison Over Multiple QuickTime Files with Different Disk Locations

25

Concluding Remarks

- We presented a "just-in-time" disk scheduling algorithm that attempts to meet timing deadlines while trying to keep system throughput.
- We have designed and implemented a real-time filesystem in RT-Mach using its resource reservation model.

26