Scheduling Soft Real-Time Jobs Over a Dual Disk System

Chetan Vaity (02329901) Shweta Agrawal (02329013)

September 24, 2003

1 Introduction

Systems like space shuttle, are hard real time systems, they require specialized real-time hardware that guarantee timely processing of jobs. However, there are other *soft*-real time systems for which meeting every single job deadline causes poor utilization of system resources, is unnecessary or even impossible. For example, a disk I/O system should be designed in such a way that hot spot access be done promptly to avoid long queues and improve response time. However some misses may be tolerable.

To build a soft real time system, one may or may not be able to use specialized real-time hardware, such as a disk that schedules I/O requests according to deadlines. Sophisticated disk controllers that employ algorithms such as Elevator algorithm schedule disk requests according to disk block position instead of request deadlines. If we only have a single non-real time standard hardware component, where applications have no control over low level scheduling algorithm, there is not much that we can do. However, if we have redundant components of such systems, for example, a disk system with mirrored disks, it may be possible to control how jobs are dispatched to replicated components to achieve real-time goals. In this simulation project, we study and compare two schemes for dispatching jobs (read requests) to the mirrored disks in a dual disk system to achieve soft-real time deadlines.

2 System description

For simulating a dual disk system, we make the following assumptions:

1. I/O requests are being generated according to a Poisson distribution at a rate of λ requests per second.
2. Only read requests are considered (*write* has to be processed by both the disks).
3. As a request is generated, it is dispatched, according to the dispatcher strategy adopted to one of the disks.

4. The disks mirror each other and are equally capable of satisfying I/O requests.

5. The disk controllers internally perform I/O request buffering and scheduling using Elevator algorithm, thus the queue at each of the disk controller is not simple FCFS but processed using elevator algorithm.

6. Each disk request can be represented by a track number (Rotational optimization is not considered). We assume that the track numbers are uniformly distributed in the range [1, MaxTrack]. where MaxTrack is the highest track number of the disk.

7. The following formula is used to compute the access time $Access(n)$ for an I/O request n tracks away from current head position: $Access(n) = (DiskFactor\sqrt{n} + DiskConstant) msec.$ *DiskFactor* is the seek time scaling factor and *DiskConstant* measures the rotational latency plus the transfer time of an average request.

8. Associated with each request is its *slack*. Slack is defined as t_d - t_a, where t_a is the arrival time of the job and t_d is the time at which job must be completed (deadline). Job slacks are assigned according a slack density function $S(x)$ where $S(x)$ is uniformly distributed over $[S_{min}, S_{max}]$.

9. Each request has an associated deadline (time by which the request must be completed), which is calculated as: $Deadline = Arrival\_time + Slack\_time$

10. The requests are independent of each other.

3 Dispatcher Strategies

Two schemes for dispatching requests will be studied and compared:

1. **Balance**: Dispatch a job to either server (disk controller) with equal probability.

2. **Chop**: One server, S_T is reserved for tight slack job. It handles jobs whose slacks are at the lower p quartiles of the slack distribution. Server S_L handles the rest of the load. (p only in the range $[0, 0.5]$). In this way, the jobs with more stringent time constraints are routed to one of the servers, while the other server handles the less-time critical jobs. Hence, the load on the server carrying high priority jobs will be kept lighter so that they meet their time constraints.

4 Performance measure:

The primary performance measure is the percentage of jobs that missed their deadlines. The smaller the number, the better the system performs. A second measure can be the average response time of a read request.