

Integrated QOS management for disk I/O *

Ravi Wijayarathne A. L. Narasimha Reddy
Dept. of Comp. Sci. Dept. of Elec. Engg.
Texas A & M University
214 Zachry
College Station, TX 77843-3128
{ravi,reddy}@ee.tamu.edu

Abstract

In this paper, we address the problem of providing different levels of performance guarantees or quality-of-service (QOS) for disk I/O. We classify disk requests into three categories based on the provided level of service. We propose an integrated scheme that provides different levels of performance guarantees in a single system. The proposed method uses admission control and bandwidth allocation to isolate the different categories of requests and appropriate disk scheduling to achieve the desired performance goals. It is shown that the proposed method achieves the performance goals of individual requests while allowing seek optimizations at the disk. In particular, it is shown that the proposed scheme for integrated QOS management at disk provides significantly better performance than SCAN.

1 Introduction

System level support of continuous media has been receiving wide attention. Continuous media impose timing requirements on the retrieval and delivery of data unlike traditional data such as text and images. Timely retrieval and delivery of data requires that the system and network pay attention to notions of time and deadlines. Data retrieval is handled by the I/O system (File system, disk drivers, disks etc.) and the delivery is handled by the network system (network software and the network). In this paper, we will look at the data retrieval problem.

A storage system will have to support requests with different performance requirements based on the application needs. Continuous media applications

may require deterministic performance guarantees i.e., guarantee that a requested block will be available within a specified amount of time continuously during the application's execution. A request from an interactive game or a request to change the sequence of frames in a continuous media application may require that the request have low response time i.e., may require a latency guarantee. A regular file request may only require best-effort service but may require that a certain number of requests be served in a given time i.e., may require a throughput guarantee.

There is a clear need for supporting multiple levels of performance guarantees within the storage system. Several interesting questions need to be addressed when multiple levels of QOS need to be supported in the same system: (a) how to allocate and balance resources for the different QOS levels, (b) how to control and limit the usage of resources to allocated levels, (c) how to schedule different requests to meet the desired performance goals, (d) how do system level parameters and design decisions affect the different types of requests and (e) how to tradeoff performance goals for higher throughput (for example, how much throughput gain can be had with statistical guarantees rather than deterministic guarantees)?

This paper addresses the problem of providing different levels of service for different classes of requests in a single I/O system. This paper makes the following significant contribution: an integrated scheme is presented for providing different levels of performance guarantees to different classes of requests. The proposed solution meets the the QOS goals of different classes of requests while allowing seek optimizations. We will also show that one class of requests cannot disturb the scheduler from meeting the performance goals of another class of requests.

Section 2 discusses our approach for providing different levels of QOS in a single system. Section 3 presents

*This work is supported in part by an NSF Career Award and by a grant from State of Texas Higher Education Board. To appear in IEEE Int. Conf. on Multimedia Computing and Systems, June 1999

a performance evaluation of the proposed scheme based on trace-driven simulations. Section 4 summarizes our results and points out future directions.

2 Performance Guarantees

In this paper, we consider three different categories of requests. *Periodic* requests require service at regular intervals of time. Periodic requests model the behavior of video playback where data is retrieved at regular intervals of time. Periodic requests can be either CBR or VBR. *Interactive* requests require quick response from the I/O system. Interactive requests can be used to model the behavior of change-of-sequence requests in an interactive video playback application or the requests in an interactive video game. These requests arrive at irregular intervals of time. *Aperiodic* requests are regular file requests. Disk scheduling for CBR requests is studied in [1, 2, 3, 4, 5, 6].

Our approach to providing QoS guarantees at the disk is shown in Fig. 1. We employ a two-level scheme where bandwidth allocations and resource scheduling are separated. Disk bandwidth is allocated appropriately among the different types of requests. Each class of requests employs an admission controller to limit the disk utilization of that class of requests to their allocated level. The disk scheduler considers the pool of available requests and tries to optimize their schedule while meeting the individual performance goals. In our scheme, the disk scheduler is unaware of the bandwidth allocations and recognizes the QoS goals of individual requests by the request type identifiers. Similar approaches have been independently proposed recently in [7, 8]. Both these schemes employ a two-level scheduling approach as proposed here. The work in [7] shares many of the motivations of our work, but assumes that the scheduler is aware of the admission control issues. Our approach of keeping the scheduler and the admission control policies separate enables the scheduler to be located anywhere, for example at the disk on a remote machine in a distributed system. The scheduler in [8] doesn't support quick response to interactive requests.

The admission controllers employed for each class will depend on the service provided for these requests. The admission controllers for each class of requests controls the number of requests entering the pool of requests and also the order in which the requests enter the pool. These controllers besides enforcing the bandwidth allocations, control the policy for scheduling the requests in that class of service. This can be generalized to larger number of request classes, each with its own admission controller/scheduler. The disk

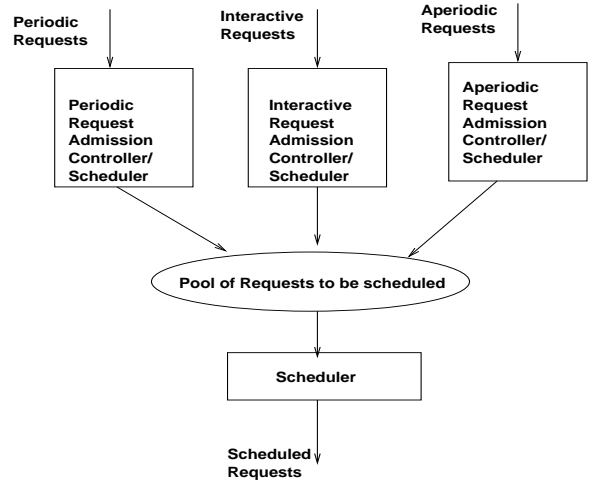


Figure 1: Supporting multiple QoS levels.

level scheduler schedules requests from the request pool to meet the performance criteria of individual requests.

In our system, it is assumed that the requests are identified by their service type at the disk scheduler. The disk scheduler is unaware of the bandwidth allocations and designed such that it is independent of the bandwidth allocations. This is done such that the bandwidth allocation parameters or the admission controllers can be changed without modifying the scheduler.

We first describe the overall functioning of the disk scheduler and then describe how admission control is implemented for each class of requests.

2.1 Scheduling for multiple QoS levels

Since the different classes of requests do not have strict priorities over each other, priority scheduling is not feasible. Periodic requests have to be given priority over others if they are close to missing deadlines. But, if there is sufficient slack time, interactive requests can have higher priority such that they can receive lower latencies. Periodic requests are available ahead of time and interactive and aperiodic requests arrive asynchronously at the disk. If periodic requests are given higher priority and served first, aperiodic and interactive requests will experience long response times until periodic requests are served. Moreover, it may be possible to better optimize seeks if all the available requests are considered at once.

The disk scheduler uses a round based scheme for scheduling the requests from the candidate pool. Each admission controller schedules the requests in

its class and releases them as candidate requests. Each admission controller ensures that its class doesn't take any more time than allocated in a round. The disk scheduler combines the requests and serves them together to meet performance goals of individual requests. If all the requests arrive at the beginning of a round, the disk scheduler will not have to worry about deadlines since the admission controllers enforce the time constraints. However, aperiodic and interactive requests arrive asynchronously. To schedule these requests as they arrive (without waiting for the beginning of next round), the disk scheduler uses the notion of a subperiod and slack times. The disk scheduler considers the available slack time of periodic requests and adjusts the schedule to incorporate any arriving interactive and aperiodic requests each subperiod. The request schedule is not disturbed within a subperiod.

The aperiodic requests are queued into two separate queues. The first queue hold requests based on the minimum throughput guarantee provided to these requests. Scheduling these requests will not violate any time constraints since these requests are within allocated bandwidth. The second queue holds any other requests waiting to be served. The scheduler considers the requests from the second queue after the candidate request pool is exhausted such that these requests can utilize the unused disk bandwidth.

The scheduler merges the periodic requests and aperiodic requests (from queue 1) into a SCAN order at the beginning of a round. These requests are then grouped into a number of subgroups based on their location on the disk surface. The scheduler serves a subgroup of requests at a time. Later arriving aperiodic requests (of queue 1) are, if possible, merged into remaining subgroups. The scheduler considers serving interactive requests only at the beginning of a subgroup i.e., the disk SCAN order is not disturbed within a subgroup. If an interactive request is to be served, the scheduler groups this request into the closest subgroup and serves that group next. When there is no subgroup into which the interactive request can be merged, it is served out of SCAN order and the scheduler then moves to serving the next subgroup. An interactive request hence gets a quick response. Interactive requests are served by a first-come first-serve policy to limit the maximum response time of a single request. A more formal description of the scheduler is given in Fig. 2.

If all the requests arrived at the beginning of the round, guarantees will be met if the individual groups observe the bandwidth allocations. However, aperiodic and interactive requests arrive asynchronously. Since an arriving interactive request disturbs the SCAN schedule, it is assumed to require worst-case seek time

```

While(true)
{
  Combine periodic and aperiodic1 requests
  into SCAN order;
  Break the requests into subgroups;
  slack_time = round_time - service estimate
  for above requests;
  while(not end of round)
  {
    pick first interactive request if any;
    Combine with one of the remaining
    subgroups?
    If (no)
    {
      while (estimated service time <
      slack_time)
        service waiting interactive requests;
      Serve the closest subgroup;
    }
    else serve the merged subgroup;
    Adjust slack_time;
    while (slack_time > 0)
    {
      Combine aperiodic1 requests into
      existing subgroups;
      Adjust slack_time;
    }
    if (all periodic requests served)
    {
      Continue serving interactive &
      aperiodic requests until
      end of round;
    }
  }
}

```

Figure 2: Semi-formal description of the scheduler.

such that servicing this request won't violate the bandwidth allocations of other requests. The slack times of periodic requests are considered while scheduling these late arriving requests so as to not violate the deterministic guarantees for periodic requests. The disk scheduler keeps track of the least slack time for a periodic request to be served in this round. If the estimated service time for an arriving request (interactive or aperiodic) is below this slack time, then it is served within this round. If the estimate is above the slack time, the request is considered only after all the periodic requests are served within this round.

2.2 Admission Controllers

We used a novel admission controller for periodic requests. In this scheme [9], the I/O system keeps track of the worst-case time committed for service in each round at each of its disks in the form of a *load trace*. Before a stream is admitted, its demand trace is combined with the load trace of the appropriate disks to see if the load on any one of the disks exceeds the capacity (committed time greater than the length of the round). The load trace of a system consists of load traces of all the disks over sufficient period of time. A stream is admitted if its demand can be accommodated by the system. It is possible that the stream cannot be supported in the round the request arrives. The *stream scheduling* policy will look for a round in which this stream can be scheduled. The admission controller assumes that a stream will wait for a maximum amount of time, given by *latency target*, for admittance.

It has been shown that this approach allows statistical multiplexing of resources while providing deterministic guarantees. Our earlier evaluations have shown that this approach can yield significantly better throughput than peak-rate based allocations [9, 10].

Latency guarantees are provided by the disk scheduler as explained earlier. When requests arrive randomly, a burst of requests can possibly disturb the guarantees provided to the periodic requests. To avoid this possibility, interactive requests are controlled by a leaky-bucket controller which controls the burstiness by allowing only a certain maximum number of interactive requests served in a given time window.

Aperiodic requests are provided bandwidth guarantees by restricting the periodic and interactive requests to certain fraction of the available bandwidth. The admission controller for periodic and interactive requests enforce the bandwidth allocations. Aperiodic requests utilize the remaining I/O bandwidth. If periodic and interactive requests cannot utilize the allocated bandwidths, aperiodic requests are allowed to utilize the available bandwidth to improve the response times for aperiodic requests. Bandwidth guarantees are provided to aperiodic requests by ensuring that certain minimum number of requests are scheduled every round.

3 Performance Evaluation

3.1 Simulations

We evaluated a number of the above issues through trace-driven simulations. A system with 8 disks is simulated. Each disk is assumed to have the character-

Table 1. Disk characteristics.

Parameter	Value
Zero Seek time	0.60 ms
Avg. Seek time	8.0 ms
Max. Seek time	17.0 ms
Min. Transfer rate	11.5 MB/s
Max. Transfer rate	17.5 MB/s
Ave. latency	4.17 ms
Spindle speed	7200 RPM
Num. cylinders	3711

istics of a Seagate Barracuda drive [11]. The disk drive characteristics are shown in Table 1. In simulations, it is assumed that the first block of each movie stream is stored on a random disk.

Interactive requests and aperiodic requests are modeled by Poisson arrival. Periodic requests are based on real traces of VBR movies obtained from University of Wuerzburg [12]. Periodic request load is varied by requesting more streams to be scheduled. Interactive requests always ask for 64KB of data and aperiodic requests are uniformly distributed over (4kB, 128KB). The burstiness of interactive requests is controlled at each disk by a leaky bucket controller that allowed a maximum number of interactive requests per second based on their bandwidth allocation.

If the stream requests are assumed to arrive randomly over time, more streams can be admitted. However, to study the worst-case scenario, we assumed that all the requests arrive at once. The simulator tries to schedule as many streams as possible until a stream cannot be scheduled. The number of streams scheduled is the stream throughput. Four different video streams are considered in our study as explained below.

3.2 Results

Fig. 3 shows the average disk utilization when periodic requests are allocated 50% bandwidth, aperiodic and interactive requests are each allocated 25% bandwidth. The number of periodic requests streams was maintained at the maximum that the system can support. Aperiodic request rate is varied while maintaining the interactive request rate at 50 requests/sec (request rates are measured over the entire system of 8 disks). It is observed that the average utilization of the periodic streams stays below 50%. Because of variations in demand over time, more periodic streams could not be admitted. The utilizations of periodic and interactive requests are unaffected by the aperiodic request rate. It is also observed that as we increase the aperiodic

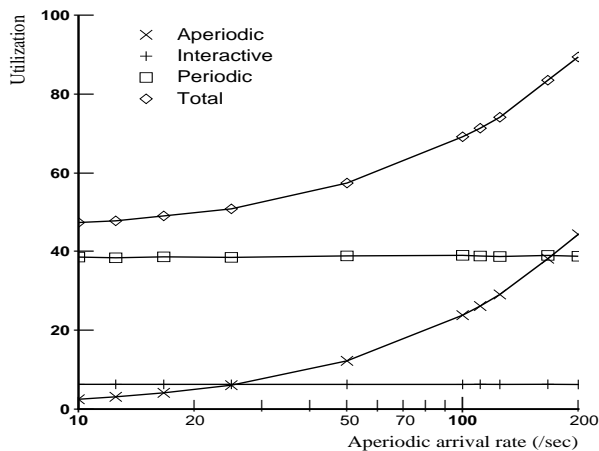


Figure 3: Average disk utilizations across request categories.

request rate, aperiodic requests take up more and more bandwidth and eventually utilize more than the allocated 25% bandwidth. When periodic and interactive requests don't make use of the allocated bandwidth, aperiodic requests make use of any available bandwidth (25% is the minimum available) and hence can achieve more than the allocated 25% utilization of the disk. This shows that disks are not left idle when aperiodic requests are waiting to be served.

Fig. 4 shows the average and maximum response times of aperiodic and interactive requests as a function of the aperiodic request rate. The number of streams is kept at the maximum allowed by the system and the interactive arrival rate is kept at 50 requests/sec. Interactive response times are not considerably affected by the aperiodic arrival rate and the maximum interactive response time stays relatively independent of the aperiodic arrival rate. It is also observed that the interactive requests achieve considerably better response times than aperiodic requests (260ms maximum interactive response time compared to 1600ms for aperiodic requests both at 50 reqs/sec). Both average and maximum response times are better for interactive requests than for aperiodic requests even at lower aperiodic arrival rates. We observed that the maximum interactive response times are only dependent on the burstiness of arrival of interactive requests and the bandwidth allocated to them. Zero percentage of periodic requests missed deadlines as aperiodic request rate is varied.

Fig. 5 shows the response times of aperiodic requests and interactive requests (both at 25 requests/sec) as the number of requested streams in the system is varied

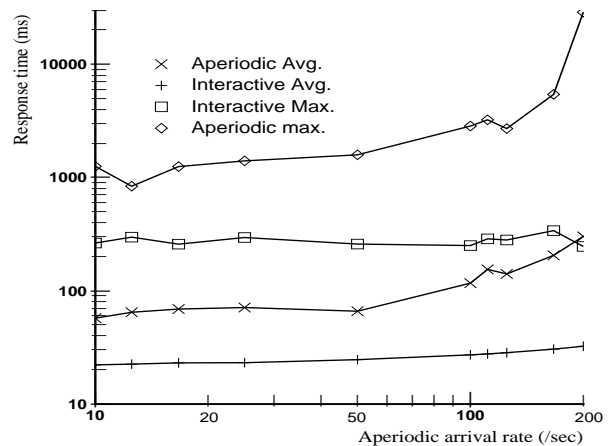


Figure 4: Impact of aperiodic arrival rate on response times.

from 5 to 100. With the considered allocation of bandwidths, the system could support a maximum of 33 streams. Hence, even when more number of streams are requested, the system admits only 33 streams. This shows that the periodic request rate is contained to allow aperiodic requests and interactive requests to achieve their performance goals. We observe that the maximum response times of interactive requests are not considerably impacted by the number of requested streams in the system.

Fig. 6 shows a comparison of the proposed scheduling algorithm and a variant of SCAN scheduling algorithm used in most of the current disks [13]. The figure shows the average and maximum response times of interactive requests as the aperiodic request rate is varied. The proposed method achieves better average and maximum response times compared to SCAN. As the aperiodic arrival rate is increased, both the maximum and average response times of interactive requests get impacted with SCAN scheduling policy. The proposed method isolates the request categories and maintains the performance of the interactive requests independent of the aperiodic arrival rate.

4 Conclusions and Future work

In this paper, we addressed the problem of providing different performance guarantees in a disk system. The proposed approach uses admission controllers and an appropriate scheduler to achieve the desired performance goals. We showed that through proper bandwidth allocation and scheduling, it is possible to design a system such that one type of requests

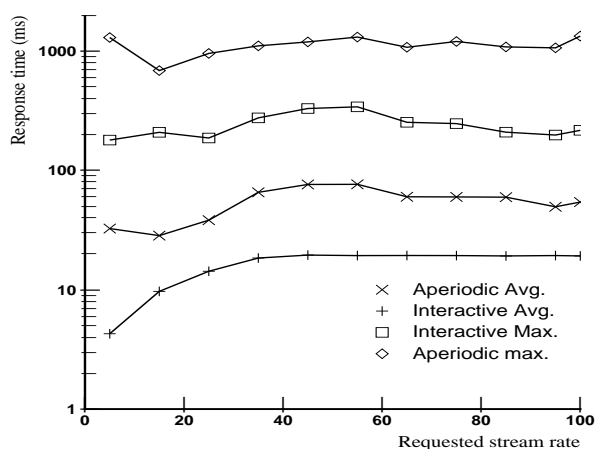


Figure 5: Impact of requested stream rate on response times.

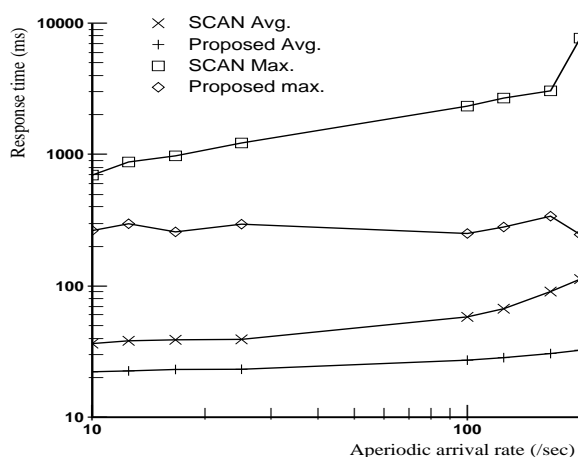


Figure 6: Comparison with SCAN.

do not impact the performance of another type of requests. The proposed scheduling policy allows seek optimizations while achieving the performance goals of individual requests. The proposed scheduling policy is *work conserving*, i.e., did not let the disk be idle when there are requests to be served. The clear separation of the scheduler from the admission controller will allow this scheme to be useful in a distributed system. Our results showed that the proposed approach performs significantly better than SCAN in meeting the stated performance goals of the different classes of requests.

In the work presented here, we used static bandwidth allocations to achieve performance goals. We are currently investigating issues in dynamic allocation and adaptive performance guarantees.

References

- [1] A. L. N. Reddy and J. Wyllie. I/O issues in a multimedia system. *IEEE Computer*, Mar. 1994.
- [2] P. S. Yu, M. S. Chen, and D. D. Kandlur. Grouped sweeping scheduling for dasd-based multimedia storage management. *Multimedia Systems*, 1:99–109, 1993.
- [3] D. J. Gemmell and S. Christodoulakis. Principles of delay-sensitive multimedia storage and retrieval. *ACM Trans. on Info. Systems*, pages 51–90, 1992.
- [4] C. Martin, P. Narayanan, B. Ozden, R. Rastogi, and A. Silberschatz. The Fellini multimedia storage server. in *Multimedia Information Storage and Management*, Ed: S.Chung, KluwerPublishers, 1996.
- [5] A. Molano, K. Juvva, and R. Rajkumar. Guaranteeing timing constraints for disk accesses in rt-mach. *Proc. of Real time systems Symposium*, Dec. 1997.
- [6] T. Niranjan, T. Chiueh, and G. A. Schloss. Implementation and evaluation of a multimedia file system. *Proc. of IEEE Conf. on Multimedia Computing and Systems*, pages 269–276, June 1996.
- [7] P.J.Shenoy and H. M. Vin. Cello: A disk scheduling framework for next generation operating systems. *Proc. of ACM SIGMETRICS*, June 1998.
- [8] M. M. Budhikot, X. J. Chen, D. Wu, and G. M. Parulkar. Enhancements to 4.4 BSD UNIX for efficient networked multimedia in project MARS. *Proc. of IEEE Multimedia Computing and Systems Conf.*, pages 326–337, June 1998.
- [9] A. L. Narasimha Reddy and R. Wijayarathne. Techniques for improving the throughput of VBR streams. *Proc. of ACM/SPIE Conf. on Multimedia Computing and Networking*, Jan. 1999.
- [10] A. L. Narasimha Reddy and R. Wijayarathne. On providing deterministic guarantees for VBR streams. *Tech. rep. TAMU-ECE-9701, Texas A&M University*, Apr. 1997.
- [11] Seagate Corp. Disk drive product info. <http://www.seagate.com/>, 1997.
- [12] O. Rose. Mpeg trace data sets. <ftp://info3.informatik.uni-wuerzburg.de>, 1995.
- [13] B. L. Worthington, G. R. Ganger, and Y. N. Patt. Scheduling algorithms for modern disk drives. *Proc. of ACM SIGMETRICS*, pages 241–251, May 1994.