Automated QoS Support for Multimedia Disk Access

Joel C. Wu, Scott Banachowski, and Scott A. Brandt
Computer Science Department, University of California, Santa Cruz

ABSTRACT

This paper describes the AutoQoS mechanism, which improves the timeliness of disk accesses for multimedia applications without requiring any explicit information about their constraints. Multimedia applications typically have periodic time constraints, meaning that they must complete data processing at periodic intervals in order to function correctly. This requirement extends to the disk system, because the application must access data on time in order to meet deadlines. By using Quality of Service algorithms for disk services, an application may receive enough bandwidth and isolation from other disk accesses to read data on time. Nevertheless, past approaches are restrictive because they require that disk bandwidth or deadlines be known and specified in advance. Our system infers from I/O behavior the bandwidth requirement of multimedia streams, and automatically adjusts allocations in order to provide Quality of Service without knowing the constraints or requiring intervention from the application.

Keywords: Storage, QoS, Soft Real-time, and Multimedia Systems

1. INTRODUCTION

Modern commodity systems are expected to run mixed-workloads that, in addition to traditional desktop applications, include tasks with periodic deadlines such as multimedia. Often meeting those deadlines depends on timely disk access, for example, a video that does not fit entirely in volatile memory must be periodically read from disk during playback. For satisfactory multimedia performance, systems must provide real-time access to the disk. Usually this implies *a priori* knowledge of constraints such as deadline or bandwidth requirements. We developed a mechanism called *AutoQoS* to provide disk Quality of Service to multimedia applications without knowledge about constraints. This supports legacy multimedia applications that were developed for best-effort systems. The operating system uses heuristics to infer the needs of applications, and adjusts the allocation of disk bandwidth among them so that multimedia applications receive constant bandwidth in isolation from the other workload. In this paper, we show that the concept of AutoQoS is well-suited to improve the performance of best-effort systems, such as desktop computers and servers, at handling multimedia workloads.

2. RELATED WORK

Existing systems support real-time disk access in several ways, including low-level scheduling approaches. Deadline-based disk schedulers submit requests to the disk such that those with deadlines are served on time while at the same time reducing unnecessary disk seeks, which incur large overhead [1–3]. These algorithms will meet deadlines, but require that the deadline for each request is known by the system; our system does not require this extra capability. Other disk schedulers do similar operations to meet specific bandwidth reservations or guarantees [**?**, 4]. All of these methods require low-level information about disk devices in order to make scheduling decisions. The AutoQoS approach treats the low-level disk scheduler as a black box, and ensures that requests meet their constraints by preventing overload from the non-constrained traffic. Our current implementation uses the default Linux disk scheduler, however that does not preclude using reservation approaches in the future.

Disk schedulers may be augmented with higher-level, policy-based mechanisms to provide different service levels to classes of applications [5–7]. These systems are hierarchical, using different schedulers to prioritize requests from each class, with a global scheduler to combine the requests and submit them to disk. AutoQoS has a similar goal of classifying requests (in our case we have two classes, one with deadline constraints and one without). However, instead of using

Further author information:
Email: {jwu,sbanacho,sbrandt}@cs.ucsc.edu

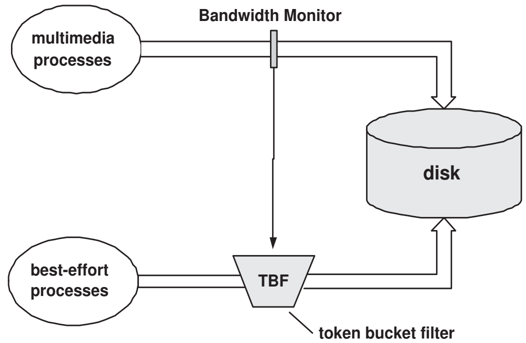


Figure 1. Inferring the stream rate from the multimedia path controls the bandwidth allocation of the competing best-effort path.

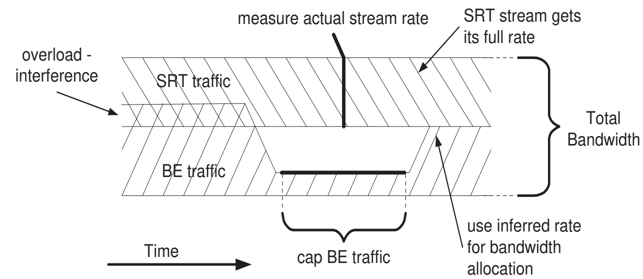


Figure 2. The system limits best-effort traffic in order to expose and measure the real multimedia stream rate.

different schedulers for each class, we limit the bandwidth of the non-critical class during overload. This prevents best-effort workload from interfering with multimedia requests. Unlike the other QoS systems, AutoQoS works strictly on the principle of overload avoidance.

A technique similar to AutoQoS automatically adjusts disk bandwidth for real-rate applications by using feedback from their measured progress [8]. The goal of this proposed system is to match the disk I/O rates to the real-rate needs of applications. Our goal is to address a mixed-workload scenario. Their disk bandwidth allocator is controlled by monitoring the buffer cache, whereas our mechanism monitors the bandwidth above the file-system page cache. Another similar algorithm is a YQF [9], which controls disk bandwidth using a proportional bandwidth sharing algorithm; in contrast our system uses a bandwidth limiting algorithm to control disk bandwidth.

AutoQoS differs from other approaches because it infers all the necessary information through the system, without placing any burden on the application. While there is much research on providing QoS support for disk storage, none were explicitly motivated by the desire to minimize the amount of information required from the applications or users.

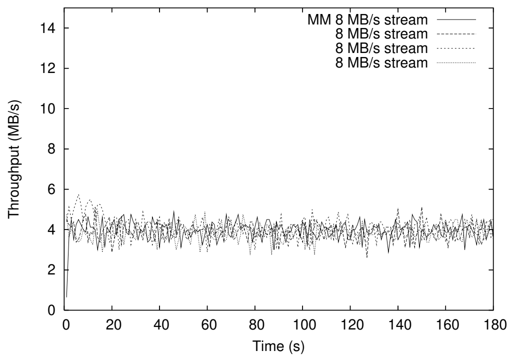
3. AUTOMATED QOS SUPPORT

The principle of the system is that when disk bandwidth is overloaded, multimedia applications should have priority in order to meet their timing constraints. Therefore, our system runs in its default state of taking no action until it recognizes that a multimedia application begins to stream data at a sustained rate, in which case it adjusts allocations so that the media stream receives a reservation of disk bandwidth equal to its observed flow rate.

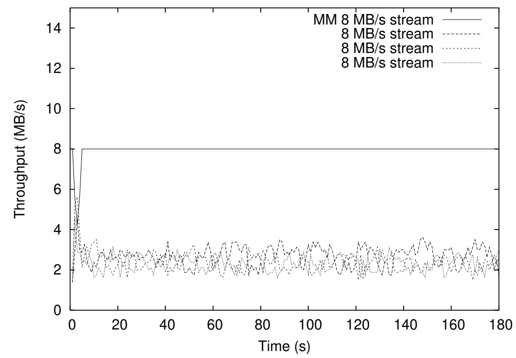
This algorithm is fully automated, and requires no explicit information from applications or users. The information necessary to make bandwidth allocation decisions is inferred by the system, thus supporting legacy multimedia applications without modification. To achieve this we need four abilities: to differentiate disk requests, to shape bandwidth, to determine the disk bandwidth requirement of multimedia processes, and to manage and adapt the allocated bandwidth, all in a fully automated way. Our system automatically differentiates disk requests by labeling those requests that act on multimedia files (recognized by file extension) as belonging to the multimedia class, and the rest as best-effort class. The remainder of this section highlights the other three functions.

3.1. Shaping disk bandwidth

To control the disk bandwidth allocations, we use a mechanism based on the token-bucket filter (TBF) [10]. TBF shapes traffic in networks by controlling the rate and burst size of data transmission. In order to transmit a packet, a token from the bucket must be spent. If there are no tokens, the packet must wait until a token is replenished. The rate of token replenishment governs the maximum rate of transmission, and the size of the bucket determines the maximum number of packets that may be sent at once. In our implementation, each token represents a block of data, so the token rate limits the disk bandwidth, and the bucket size limits the number of outstanding requests. By using a TBF to limit the rate that best-effort processes issue requests, we effectively provide guaranteed bandwidth to the multimedia processes.



(a) Normal Linux behavior



(b) AutoQoS supporting one multimedia stream

Figure 3. Throughput for four 8 MB/s streams, one of the streams is multimedia and is boosted by AutoQoS.

We envision a disk as having two access pipes: one pipe carries the multimedia traffic and the other pipe carries all the best-effort (BE) traffic (see Figure 1). By limiting the rate of BE requests we supply the remainder of the disk bandwidth to multimedia tasks. Our previous work showed that bandwidth limiting is effective in managing multimedia disk allocations [11].

3.2. Inferring stream rate

To achieve the goal of QoS support without requiring any explicit information from applications, the system must identify when disk bandwidth is overloaded and multimedia requests are at risk of not meeting their time constraints. To do this in an automated way, AutoQoS infers the streaming rate of multimedia traffic. We currently ignore write requests, because our multimedia workloads are read intensive, and because write performance depends less on disk response time due to write buffering.

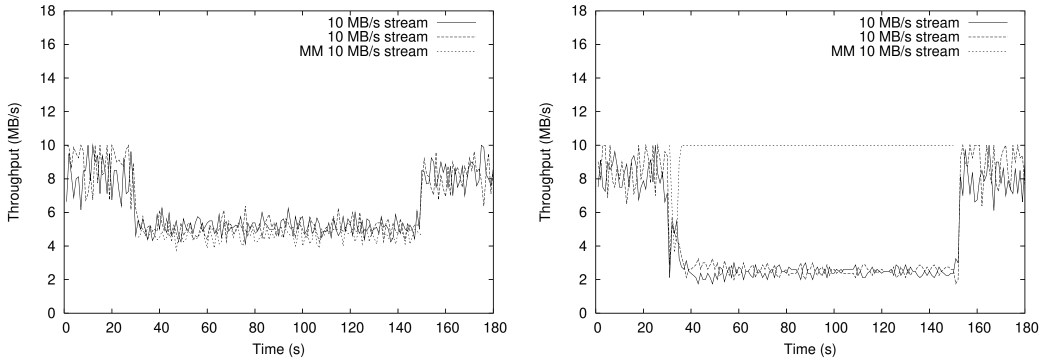
To infer a multimedia application's bandwidth, whenever a process initiates access to a multimedia file, AutoQoS reduces the best-effort bandwidth to a predetermined fraction of the entire disk bandwidth. This allows the multimedia application to use a reserved portion of the disk bandwidth (possibly shared with other multimedia applications). By reducing the best-effort bandwidth, we temporarily over-provision bandwidth to multimedia streams, so that they may proceed at full rate. During this time, we monitor the actual consumed bandwidth, and once determined, increase the best-effort bandwidth to take up the portion of disk bandwidth unused by the multimedia applications (see Figure 2).

3.3. Managing the bandwidth

Once the bandwidth of a multimedia stream is determined, the limit on best-effort bandwidth is controlled using a heuristic. The heuristic maintains the limit at the point where best-effort workload does not interfere with multimedia, while attempting to maximize the total bandwidth utilization. The system constantly monitors the rate of multimedia traffic, and by comparing it to the previously inferred rate, adjusts the best-effort limit accordingly: if the multimedia traffic decreases then the best-effort limit drops, and if the multimedia traffic saturates its bandwidth, the best-effort limit increases. This also re-calibrates the system in the case of dynamic changes to workload.

4. RESULTS

We implemented the automated QoS mechanism for the Linux 2.6 kernel block device driver. Our test application simulates multimedia file access by making periodic streaming reads of disk data. In our experiments, several test applications run concurrently to load the disk, and different scenarios are created by labeling each test application either multimedia or best-effort. Performance is measured by observing the throughput of each stream. The goal of experiments was to test whether the automated QoS mechanism will support multimedia streams as we hypothesized. Our test system is a 1.5 GHz



(a) Normal Linux behavior

(b) AutoQoS supporting one multimedia stream

Figure 4. Three 10 MB/s streams. The multimedia stream enters at time 30 and terminates at time 150 and is boosted by AutoQoS.

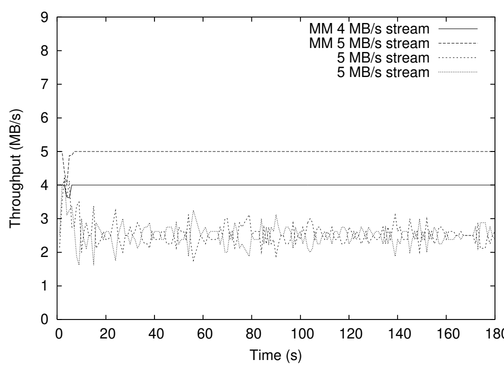


Figure 5. Support for more than one multimedia stream.

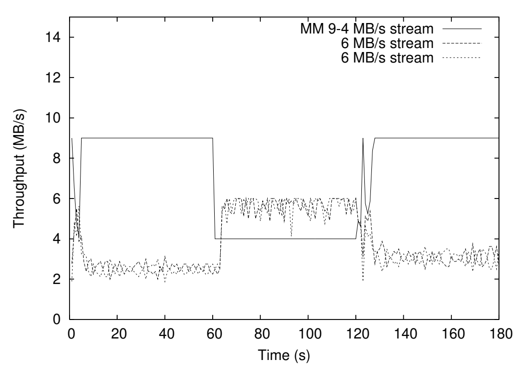


Figure 6. Adapting the best-effort limit to changing multimedia stream rates.

P4 with 512MB of RAM. The disk is a Seagate ST340810A IDE drive formatted with ext2 file system. The bandwidth of the disk averages 27.6 MB/s for a large sequential read.

Figure 3(a) shows the observed read bandwidth when running four 8 MB/s streams simultaneously, one of which is labeled as a multimedia (MM) stream, on a Linux system without QoS support. All four streams receive only about half of the bandwidth they request, because the total load exceeds the disk bandwidth capacity. The observed aggregate stream rate is approximately 16 MB/s, significantly less than when only a single sequential file is read; this is due to extra seeks between reads from the different streams. Figure 3(b) shows the same workload when using AutoQoS. The multimedia stream receives its desired 8 MB/s bandwidth. The other three competing best-effort streams are limited and each of them receives about 2.5 MB/s. The aggregate throughput for these two runs are approximately the same: 15.86 MB/s for normal Linux and 15.40 MB/s with AutoQoS.

Figure 4 shows the effect of a multimedia stream entering and exiting the system with competing best-effort traffic. The workload consists of three 10 MB/s streams, one of which is multimedia. Figure 4(a) shows default Linux. Two streams start at time zero, and when the third stream enters at time 30, the overload reduces the bandwidth of each stream to 5MB/s. Figure 4(b) shows that with AutoQoS, the multimedia stream receives its desired rate of 10 MB/s, because the bandwidth of the two competing streams is limited to make room for it. When the multimedia stream terminates at time 150, the two competing streams regain their original bandwidth. The overall throughput for running with and without AutoQoS are the same (15.52 MB/s).

Figure 5 shows AutoQoS supporting multiple multimedia streams. Here a 4 MB/s and a 5 MB/s multimedia stream compete with two 4 MB/s best-effort streams, and the multimedia streams are able to receive the bandwidth they need.

Figure 6 shows the ability of AutoQoS to adapt to changes in the rate of the multimedia stream. In this run, the multimedia stream initially starts at a rate of 9 MB/s, changing to 4 MB/s at time 60, and then back to 9 MB/s at time 120. There are two 6 MB/s competing best-effort streams. When the MM rate drops at time 60, the best-effort rates take up the slack, and when the MM rate increases at time 120, the best-effort rates return after a short interval of adjustments.

5. CONCLUSION

QoS support for disk is important for multimedia systems. Existing work focuses on providing guarantees based on advanced reservation or specification of disk request deadlines. We developed a QoS mechanism that supports multimedia streams in a mixed-workload environment without requiring any explicit information from the application. We explained how this mechanism infers and sustains the disk transfer rates required by multimedia applications when the disk bandwidth is overloaded. The benefit of this approach is the absence of application complexities associated with utilizing existing QoS mechanisms—in this system QoS support is fully automated. Legacy applications based on best-effort systems, without knowledge of QoS mechanisms, immediately benefit by receiving better performance.

ACKNOWLEDGMENTS

This research was funded in part by a DOE High-Performance Computer Science Fellowship, Intel, and the Lawrence Livermore, Los Alamos, and Sandia National Labs. Thanks to Caixue Lin for suggestions.

REFERENCES

1. A. L. Reddy and J. Wyllie, "Disk scheduling in a multimedia I/O system," in *Proceedings of ACM Conference on Multimedia*, pp. 225–233, ACM Press, 1993.
2. R.-I. Chang, W.-K. Shih, and R.-C. Chang, "Multimedia real-time disk scheduling by hybrid local/global seek-optimizing approaches," in *Proceedings of Seventh International Conference on Parallel and Distributed Systems*, pp. 323–330, July 2000.
3. R. Abbott and H. Garcia-Molina, "Scheduling I/O requests with deadlines: A performance evaluation," in *Proceedings of the IEEE Real-Time Systems Symposium (RTSS '90)*, pp. 113–124, December 1990.
4. L. Reuther and M. Pohlack, "Rotational-position-aware real-time disk scheduling using a dynamic active subset (DAS)," in *Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS 2003)*, IEEE, December 2003.
5. P. Shenoy and H. Vin, "Cello: A disk scheduling framework for next generation operating systems," in *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp. 44–55, ACM Press, 1998.
6. Z. Dimitrijevic and R. Rangaswami, "Quality of service support for real-time storage systems," in *Proceedings of the International IPSI-2003 Conference*, October 2003.
7. R. Wijayaratne and A. L. Reddy, "Integrated QOS management for disk I/O," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 487–492, June 1999.
8. D. Revel, D. McNamee, C. Pu, D. Steere, and J. Walpole, "Feedback based dynamic proportion allocation for disk I/O," Tech. Rep. CSE-99-001, Oregon Graduate Institute of Science and Technology, December 1998.
9. J. L. Bruno, J. C. Brustoloni, E. Gabber, B. Özden, and A. Silberschatz, "Disk scheduling with quality of service guarantees," in *Proceedings of the 1999 IEEE International Conference on Multimedia Computing and Systems (ICMCS '99)*, pp. 400–405, 1999.
10. D. Clark, S. Shenker, and L. Zhang, "Supporting realtime applications in an integrated services packet network: Architecture and mechanisms," in *Proceedings of the ACM SIGCOMM*, 1992.
11. J. C. Wu and S. A. Brandt, "Storage access support for soft real-time applications," in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '04)*, (Toronto, Canada), May 2004.