

# Disk Scheduling Proposal for an In-Band Bandwidth Virtualization Schema

J. Fernandez, J. Carretero, F. Garcia-Carballeira, A. Calderon and J. D. Garcia  
Computer Architecture Group, Computer Science Department,  
Universidad Carlos III de Madrid, Leganes, Madrid, Spain.  
E-mail: {[jfernand](mailto:jfernand@arcos.inf.uc3m.es),[jcarrete](mailto:jcarrete@arcos.inf.uc3m.es),[fgarcia](mailto:fgarcia@arcos.inf.uc3m.es),[acaldero](mailto:acaldero@arcos.inf.uc3m.es),[jdgarcia](mailto:jdgarcia@arcos.inf.uc3m.es)}@arcos.inf.uc3m.es

## Abstract

*This paper<sup>1</sup> proposes a new disk scheduling algorithm for a storage virtualization schema, decoupling virtual disks and physical disks. It allows the system to virtualize not only the storage capacity, but also the storage bandwidth, following QoS directives. That virtualization can be applied to the applications bandwidth and access time requirements, allowing that each virtual disk could be used for real-time or best-effort applications. The scheduler proposed maintains the QoS of real time requests by rejecting those which deadline will fail. All those efforts are oriented towards the consecution of a network In-Band virtualization schema that not only controls the storage bandwidth, but it will also virtualize a centralized cache in order to obtain a better performance of the whole system.*

**Keywords:** Distributed Architectures; Multimedia Systems; Quality of Service; Scheduling and resource management; Virtualization.

## 1 Introduction

There is a permanent interest on optimizing the I/O performance. The best tool for achieving this objective has been optimizing the scheduling politics of I/O devices. The performance of these techniques depends on which kind of tasks will be performed. However, the requirements and the platforms for real time tasks (as multimedia tasks) and general purpose tasks seemed to be so different [6] that people developed specialized systems. That is because disk scheduling algorithms for general purpose systems tried to reduce the access time, while real time systems tended to satisfy the real-time constraints, normally for cyclical streams.

With the multimedia applications increasing, some authors [10] have proposed the design of a new kind of system, named integrated, that includes

facilities to support heterogeneous multimedia and general purpose information. In an integrated system, the user may request the start of a new I/O request (stream) during run-time. The system must determine, following some admission criteria, whether the stream is schedulable to admit or reject the stream.

In present days there is a trend towards what is call Storage Virtualization. This trend appears because nowadays all the storage elements are in process of being moved out of the hosts using Storage Area Networks (SAN) or Network Attached Storages (NAS). The use of these externals storage elements, that could serve several host al the same time, is the reason behind the need of isolating the hosts from the physical configuration of the storage network that lead to the storage virtualization.

Storage Virtualization refers to the technology that allows the creation of a set of logical storage devices from a single physical storage device. There are several ways of performing this virtualization [12]:

- Virtualization on the host: When the hosts covers all the logic of the virtualization (for example using logical volume managers).
- Virtualization on the storage servers: When all the abstractions are implemented in the storage device.
- Virtualization on the network: There are two possibilities:
  - Out-Band virtualization: When the data path is free from the virtualization logic and all the logic is present on an external element which contact with the hosts separately.
  - In-Band virtualization: When the virtualization logic is on the data path.

In general all the virtualization schemas just virtualize the storage capacity. They may also bundle multiple logical units for higher aggregated I/O rates. But there are other magnitudes that can be virtualized. Probably the most important is the storage bandwidth. Bandwidth virtualization is useful for many tasks but it is critical for those

---

<sup>1</sup> This work has been partially funded by the Project: Técnicas de optimización y fiabilidad para sistemas de Entrada/Salida escalables de altas prestaciones (Comunidad de Madrid-UC3M)

applications which have QoS requirements (like multimedia tasks).

In this paper we present a disk scheduling algorithm for a new In-Band virtualization schema that not only virtualizes storage capacity but also virtualizes storage bandwidth. This work is specially useful for QoS-aware systems (like integrated systems that support real time tasks and general purpose tasks) but not only for them. The work described in this paper is based on the functionality provided by an integrated disk scheduler that is able of scheduling the requests using their origin and their deadline, like the ones proposed in [2] or the one proposed in [13].

The layout of the paper is as follows. Section 2 presents some related works. Section 3 describes the architecture proposed. Section 4 shows the disk scheduling algorithm proposed in this paper. Section 5 shows the evaluation of our solution compared with previous solutions. Finally, section 6 shows some concluding remarks and future work.

## 2 State of the art

Storage management has attracted great research attention over the years. On recent years there were a great interest on storage virtualization, but mainly related to storage capacity. In this environment the trend is to use Out-Band schemas, but when virtualizing the storage bandwidth the In-Band schema gets the attention because the logic must be on the data path in order to scheduling the request as needed.

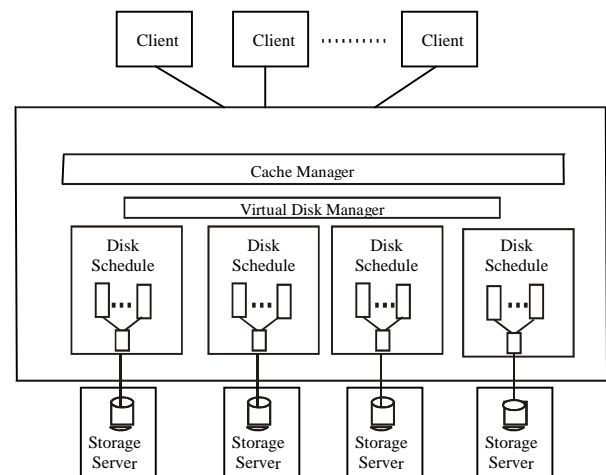
Over the years there has also been great interest on disk scheduling algorithm to achieve QoS. Other scheduling algorithms, like the Cello algorithm [11] or the one proposed for Wijayaratne and Reddy [14] or Bruno et al.[1], tried also to provide QoS. The last one describes a QoS-aware disk scheduler to implement the weighted fair queuing (WFQ) algorithm [8], which optimizes disk resource utilization by servicing one batch of high-priority requests at a time. Compared to these solutions the one proposed on the paper can achieve better results because the integration of several techniques like prior request rejection or better disk resource utilization because it is not limited for a fixed batch.

There are also solutions proposed for storage virtualization where the bandwidth is one magnitude to be virtualized. The Stonehenge System [5] provides a Multi-Dimensional storage virtualization schema that not only virtualized the capacity but the bandwidth as well. This solution is based on the Virtual Clock algorithm [15] (which has a similar

orientation that the one applied on the proposed algorithm). But this solution, while it is quite interesting, has some minor faults like those derived from using two queues for the same requests: one for achieving the deadlines and the other for optimizing the performance performing requests in advance. This schema in average would perform well at virtualizing the bandwidth but on specific instants the reserved bandwidth could not be achieved by the virtualized disks.

## 3 Architecture proposed

The architecture proposed (see Figure 1) is a typical storage virtualization architecture based on the network In-Band schema. Where the client hosts connect with the storage servers using a storage manager that covers the data path and performs all the virtualization logic.



**Fig. 1 Architecture proposal for the storage manager.**

The storage manager is the logic of the whole architecture. It will implement the virtual disks and it will attach them with the physical ones using a scheduler for each storage server. A Virtual Disk Manager will distribute the requests depending on the server storing the related data. All the bandwidth virtualization is done with the schedulers of each storage server. This manager will have also a cache manager for a better performance. The cache architecture is an on-going work in order to obtain a cache virtualization for each virtual disk.

## 4 Disk scheduling algorithm

The disk scheduling algorithm proposed in this paper used an architecture based on a two-level scheduler. That architecture decouples virtual disks from the physical disk. The first level of the architecture is composed with queues, each of one for one virtual

disk. The second level has a single queue for the physical disk. Each virtual disk queue could have assigned a different bandwidth and could have a different internal scheduling algorithm.

#### 4.1 Managing the virtual disk queues

There are two kinds of virtual disk queues:

- Real time queues.
- Best-effort queues.

The best effort queues are just regular scheduling queues with nothing more significant than the scheduling algorithm chosen for it. The requests assigned for these queues are those without time requirements (no deadlines). The scheduling algorithms more appropriated would be those based only on the disk geometry (The CSCAN algorithm is probably the best choice).

The real time queues are more complex than the best effort queues. The requests assigned for these queues are those with time requirements. Those requirements come in the form of deadlines for each request. These queues have two mechanisms for controlling the time requirements. The first is the scheduling algorithm chosen. This algorithm must be able to cope with the request deadlines and the disk geometry. There are several algorithms in the state of the art (like the SCAN-EDF algorithm that select the requests based on their deadline and when the deadline are equals uses the SCAN algorithm for the selection). The second is to assign an effective starting instant (SI) for each request. This SI value is calculated when the request has been placed on the queue by the scheduling algorithm. The meaning of the SI value is as follows: The request must be served before the effective starting instant in order that the disk could serve on time this request and all the following requests of the queue. The SI value can be calculated as following:

$$SI = \text{Min} (DL, SI_{\text{follow}}) - ST$$

Where DL is the deadline of the request and  $SI_{\text{follow}}$  is the effective starting instant of the following request on the queue. The minimum of both values would represent the moment the request must be finished so this request and all the requests that follow this one on the queue could accomplish their deadlines. The effective starting instant of this request would then be obtained adding the finish instant with the awaited service time of one request (ST). This ST value would be calculated on average. The effective starting instant is used to decide when a request must be served but it is also useful to know when a request can not be served. When a new request is inserted on the queue, the scheduler has to

recalculate all the SI values for the requests that go before the new one. If the new starting instant of the first request is an earlier one than the actual instant, then the requests of the queue can not be served on time. The solution is to reject the new request and return the whole queue to its previous state (see Figure 2).

- S1:** Place the request on the queue using the selected algorithm (SCAN-EDF, ...).
- S2:** Calculate the effective starting instant (SI) for this request. It must ensure the SI of the following request ( $SI_{\text{follow}}$ ).  

$$SI = \text{Min} (DL, SI_{\text{follow}}) - ST$$
- S3:** Recalculate SI values for all the previous requests on the queue: (They must allow to serve the new request on time).
- S4:** IF ( $SI_{\text{FirstRequest}} < \text{ActualTime}$ )
  - Reject the new request.
  - Undo the queue modifications.

**Fig. 2 Insertion algorithm for real time queues.**

#### 4.2 Managing the physical disk queue

The queue of the second level of the scheduler is the one directly attached with the disk. This queue receives the requests from the virtual disk queues and select which one would be served first. The physical queue would receive a set of requests all at once and would send all of them to the disk. When the queue is empty, a new set of requests is selected to be inserted on the queue.

The selection of the requests set is done using an iterative process. The scheduler selects one request and tries to insert it. The request selection is done by obtaining the first request of each virtual disk queue alternatively. That alternation is not exact, instead it depends of the configuration of the scheduler. Each virtual queue has assigned a percentage of requests to be served. The sum of all the virtual queues percentages must be 100%. The alternation must follow these percentages so the final set of requests selected will accomplished these percentages. Those preconfigured percentages defined the disk bandwidth percentage assigned for each virtual disk queue and they show the minimum bandwidth that would receive each virtual disk.

The selected request would be inserted on the physical disk queue, but if the request have an effective starting instant then this insertion would only happen if the SI is already valid. So if the request SI is lower than the actual time then this request is automatically rejected from the scheduler. If not, the request is send to the physical queue in order to be inserted.

The physical queue can accept or reject the request. If the request is rejected from the physical queue

then this request would be returned to its virtual queue and the selection process stops. The physical queue will then start sending the stored request to the disk. If the request is accepted then the scheduler will select another request to be inserted only if the number of request inserted is less that the maximum number of requests that the physical queue can accept. This value is previously defined using a temporal metric. This metric, called Round Time (RT), is part of the physical disk queue configuration. This value is the maximum elapsed time in which the physical queue requests must be served. The maximum number of request that can be inserted on the physical queue at once is obtained dividing this round time between the service time of one request (see Figure 3).

- S1:** Obtain the first request of the current virtual disk queue. (Virtual queues are alternated upon the predefined percentages)
- S2:** IF ( $SI < ActualTime$ )
- Reject the request
  - Go to S1
- S3:** Insert the request on the physical queue.
- IF request is rejected => Finish.
- S4:** IF ( $ServiceTime (reqs) < RoundTime$ )
- Go to S1

**Fig. 3 Selection algorithm for the physical queue.**

The process of inserting requests on the physical queue is similar to inserting requests on the real-time virtual disk queues. The scheduling algorithm used on the physical queue is always the SCAN-EDF algorithm in order to be prepared for request with deadlines. Those requests that do not have deadlines (best-effort requests) would use a virtual deadline. This deadline would be obtained from the round time. That is because all the requests inserted on the physical queue should be served before the round time is completed. So the virtual deadline of those best-effort requests inserted on the queue would be equal to the current time plus the round time. In order to be coherent, those real-time requests which deadline would be great than the final instant of the round time would also have the same virtual deadline than the best-effort requests (the rest of the requests would have a virtual deadline equal to the real deadline).

Once the current request has an associated virtual deadline, it can be inserted on the physical queue following the SCAN-EDF algorithm. As in the real time virtual queues, all the requests would have assigned an effective starting instant (SI). This SI value would be different form the one used in previous queues. The SI calculated for the physical queue have two differences from the SI calculated in previous queues:

- The requests use the virtual deadline instead of the real one.
- The service time will be used only for the real time requests. These real time requests will use the maximum service time possible to ensure that they could be served. The best effort requests will use a service time equal to zero. The reason is to avoid that the best effort requests would deny the insertion of a real time request.

As with the real time virtual queues, when a new request is inserted on the queue, the scheduler has to recalculate all the SI values for the previous requests. When the modifications are finished If the starting instant of the first request is earlier than the actual instant, then the requests of the queue can not be served on time. The solution is to reject the new request and return the whole queue to its previous state (see Figure 4).

- S1:** Assign each request a virtual deadline.  
 $VDL = \text{Min} (DL, RT + ActualTime)$
- S2:** Place the request on the queue using the SCAN-EDF algorithm.
- S3:** Calculate the effective starting instant (SI) for this request. It must ensure the SI of the following request ( $SI_{follow}$ ) (the service time wold be 0 for best effort requests).
- $$SI = \text{Min} (VDL, SI_{follow}) - ST$$
- S4:** Recalculate SI values for all the previous requests on the queue: (They must allow to serve the new request on time).
- S5:** IF ( $SI_{FirstRequest} < ActualTime$ )
- Reject the new request.
  - Undo the aueue modifications.

**Fig. 4 Insertion algorithm for physical disk queue.**

Once all of the selected requests are inserted on the physical queue, the scheduler has to send them to the disk. All this requests must be served before the round time is over. This should happen just because the way the requests set is chosen, but there may be interferences or other things that could delay the system. When the system detects that a request cannot be served on time, it must reject the request so it could not interfere with other requests. The rule is that the system only sends a request to the disk when its effective starting instant is greater than the current time and it is lower than the SI of the following request plus the service time of the actual request (that would be 0 if it is a best effort request or the maximum service time if it is a real time). The request would be rejected otherwise (See Figure 5).

- S1:** Get the first request of the queue.  
**S2:** Get the SI of the following request ( $SI_{follow}$ ).  
**S3:** IF ( $SI > ActualTime$ ) AND ( $SI_{follow} > ActualTime + ST$ )
- Serve the request
- ELSE**
- Reject the request
- S4:** IF There are request to send
- Go to S1.

**Fig. 5 Algorithm for sending request to disk.**

## 5 Evaluations

To test the algorithm proposed before, a simulation program has been implemented using the PARSEC simulation environment [7] and the DISKSIM disk simulator [4]. DISKSIM can simulate several disk systems, but for this paper a Quantum Atlas 10K disk model TM09100W, with a 9.1 GB size and 10025 RPM, has been chosen.

To compare the results of the integrated disk scheduling algorithm (ID) proposed in this paper with other disk scheduling algorithms, we have also simulated the CSCAN algorithm that is the standard for scheduling best effort request in operating systems and the SCAN-EDF algorithm, a well known scheduling algorithm for real time requests.

We have made two kinds of tests to measure the system behavior:

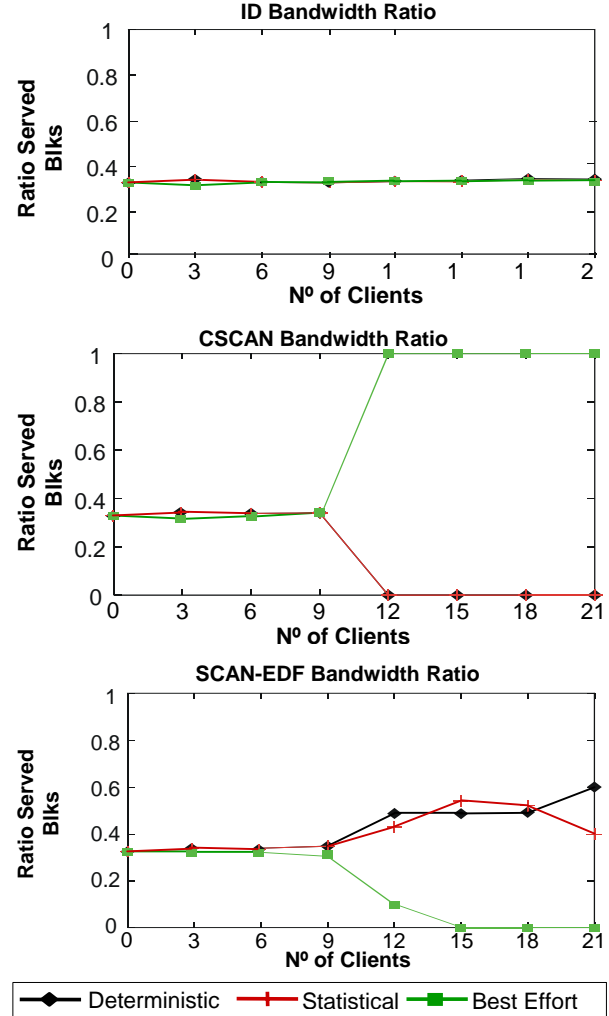
- Ability of each algorithm to maintain the selected QoS for each virtual disk.
- Raw performance of our algorithm, with QoS, compared to existing solutions without QoS.

The workload selected for those tests was a mixed combination of general purpose requests and multimedia stream-based requests. There are three virtual disks defined: one for general purpose requests and two for serving two different sets of multimedia streams: deterministic streams and statistical streams. The three disks were configured to have 33% of the total bandwidth each. The client hosts are distributed equally between the three virtual disks. Each client hosts request a bandwidth of 320 KB/sec (20 disk blocks) from the same file.

The files can be stored on one disk or in four disks with two different distribution schemas: Each file stored on a single disk or each file distributed using a round robin algorithm and with a slice of 16 KB. To obtain the result values those requests whose deadlines have failed are not included into the obtained bandwidth.

## 5.1 QoS evaluation for virtualized disk

Figure 6 shows the bandwidth percentage obtained for each virtual disk when a single physical disk is used.

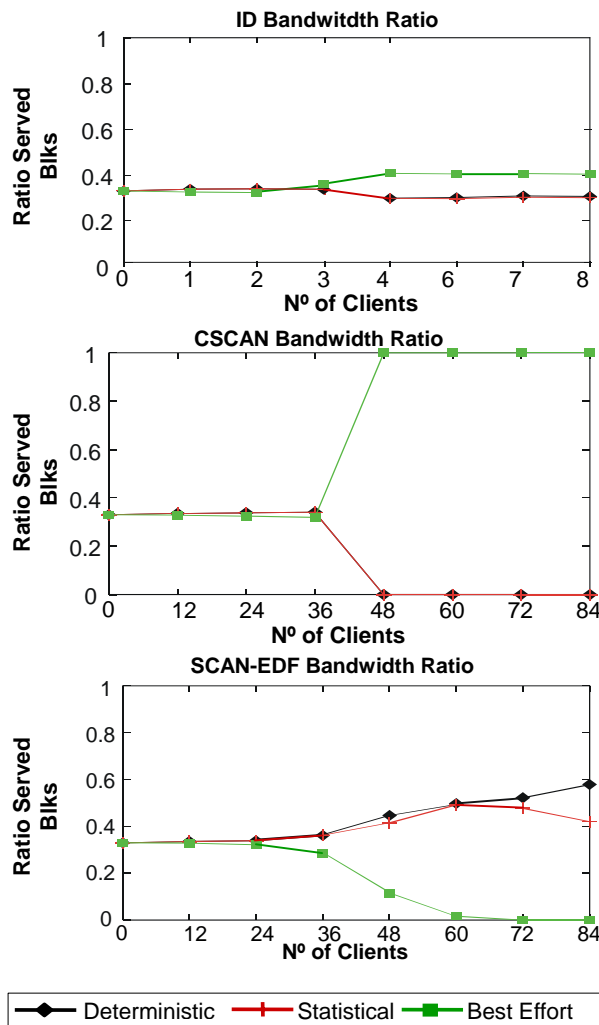


**Fig. 6 Bandwidth percentage of each virtual disk**

Our algorithm is the only one that maintains the QoS for the bandwidth percentage defined for each virtual disk. The CSCAN algorithm just serves the best-effort requests and does nothing for the real-time requests. Meanwhile the SCAN-EDF algorithm shares the bandwidth between the real time virtual disks and ignores the best-effort requests.

Figures 7 and 8 show the bandwidth percentage obtained for each virtual disk with two distributions: file per disk and striped file, respectively. The results are similar than the ones obtained for one disk. The proposed algorithm is the only one that maintains the bandwidth percentage defined for each virtual disk. The CSCAN algorithm just serves the best-effort request and does nothing for the real-time request. Meanwhile the SCAN-EDF algorithm shares the

bandwidth between the real time virtual disks and ignores the best-effort requests.



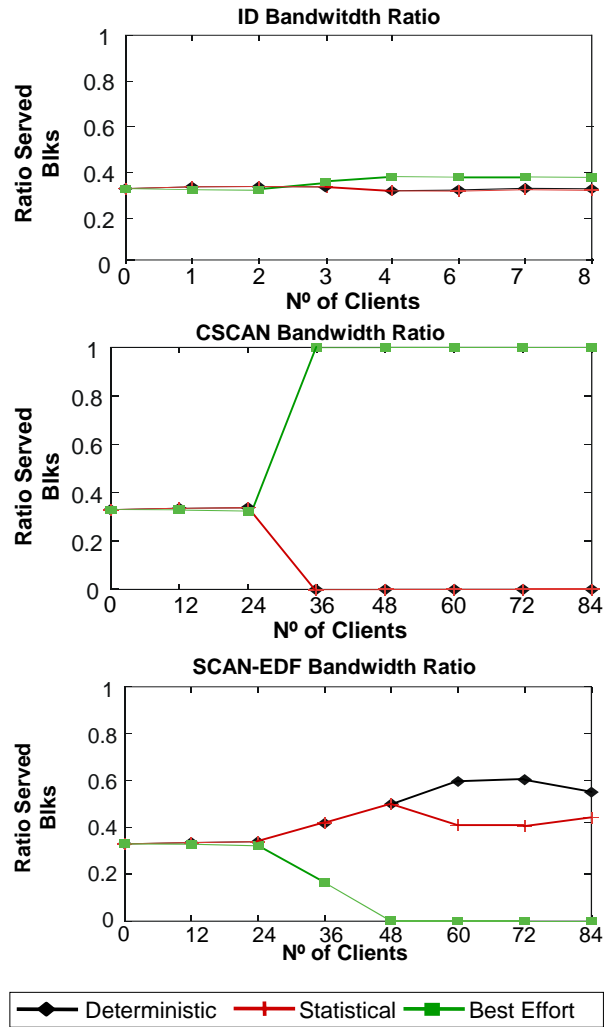
**Fig. 7 Bandwidth percentage of each virtual disk: Four Disks, each file on a disk.**

Major differences with Figure 6 are that our algorithm shows a little deviation from the configured bandwidth compared with the one disk test. That deviation is greater when we store each file on a single disk that when we distribute each file onto the four disks. The reason of this deviation should be further studied but, in spite of this, the results are good enough and clearly better than the existing algorithms.

## 5.2 Raw Performance Results.

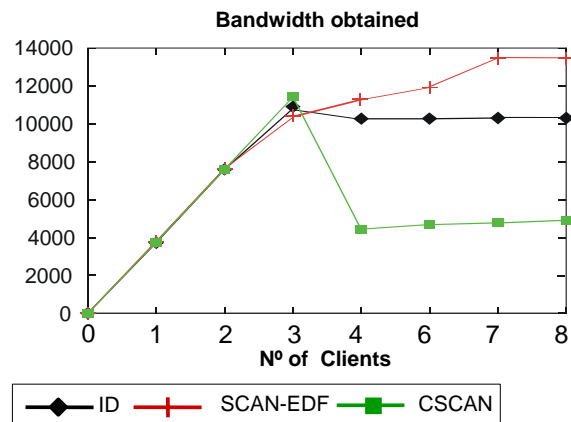
This evaluation was performed using 4 physical disks to host the virtual disks.

For the first distribution schema, where each file is stored on a single disk, Figure 9 shows the total bandwidth obtained which the three algorithms tested. The conditions are the same as when using one disk.



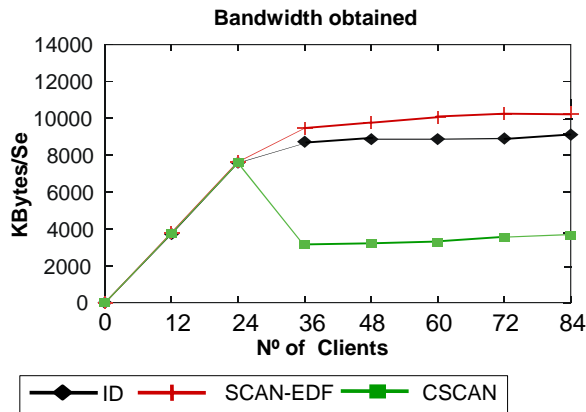
**Fig. 8 Bandwidth percentage of each virtual disk: Four Disks, files distributed onto the disks.**

The results show that the bandwidth obtained by the proposed algorithm is not the best (the SCAN-EDF gets more bandwidth). The reason is that our ID algorithm is the only one that maintains the required QoS bandwidth for each virtual disk.



**Fig. 9 Total bandwidth for four disks test: Each file on a disk.**

For the second distribution, where each file is distributed on the four disks, Figure 10 shows the total bandwidth obtained which the three algorithms tested. That distribution shows similar results as before, but this time the difference between SCAN-EDF and the proposed algorithm is lower, mainly due to a better data distribution, that results in nearer data for our algorithm.



**Fig. 10 Total bandwidth for four disks test: Files distributed onto the disks.**

## 6 Conclusions

This paper presents a disk scheduling algorithm proposed for new storage virtualization schema. This scheduler is designed in order to virtualize the storage bandwidth and to provide QoS to the applications. This algorithm uses a two level scheduler where the first-level queues are dedicated for the virtual disks and the second-level queue is dedicated to the physical storage component. This algorithm is configured to maintain a defined bandwidth of each virtual disk. But it can also be configured to serve requests of different nature (so they could be best-effort or real-time requests) and it also can configure the specific scheduling algorithm used for each virtual disk.

The evaluation performed shows that the proposed algorithm maintains the configured bandwidth of each virtual disk and at the same time it achieves the deadlines of real-time requests. The total bandwidth is less than the one obtained for a typical SCAN-EDF scheduler but is a good tradeoff in order to achieve the virtualization correctly.

Future work is going on to include a virtual cache manager at the storage manager level. Similar efforts can be found on [9] or [3]. The objective is to configure each virtual disk with the corresponding caching algorithm and, if it is possible, to manage the size of each virtual cache in order to obtain a configured percentage of cache bandwidth for each virtual disk.

## 7 References

- [1] J.L. Bruno, J.C. Brutoloni, E. Gabber, B. Orden, A. Silberschatz. Disk scheduling with quality of service guarantees. In Proc. of IEEE Int. conf. On Multimedia Computing and Systems (ICMCS'99), Florence, Italy, June 1999.
- [2] J. Carretero, J. Fernandez, F. Garcia and A. Choudhary. A hierarchical disk scheduler for multimedia systems. *Future Generation Computer Systems*, (Vol. 19, N° 1):23-35, 2003.
- [3] J. Fernandez, F. García, J. Carretero, J.M. Perez and A. Calderon. A new cache management algorithm for multimedia storage systems. In Proc. of the 18 ACM Symposium on Applied Computing. (SAC'03) Melbourne, Florida, USA, March 2003.
- [4] G. Ganger, B. Worthington and Y. Patt. The disksim simulation environment version 2.0 reference manual. Technical report, University on Michigan, Ann Arbor, 1999.
- [5] Lan Huang, Gang Peng, and Tzi-cker Chiueh. Multidimensional Storage Virtualization. Proceedings of SIGMETRICS 2004/Performance. 2004
- [6] D. Makaroff, G Neufeld and N. Hutchinson. Design of a variable bit-rate continuous media server for and ATM network. In Proc. of the IST/SPIE Multimedia Computing and Networking (San Jose, CA, USA), January 1996.
- [7] R. A. Meyer and R. Bagrodia. PARSEC User Manual. UCLA Parallel Computing Laboratory, September 1999.
- [8] A. K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated service networks: the multiple node case. *IEEE/ACM Transactions on Networking*, 2(2): 137-150, 1994
- [9] Gang Peng, Srikant Sharma, and Tzi-cker Chiueh. A Case for Network-Centric Buffer Cache Organization. Proceedings of 11th Symposium on High Performance Interconnects. (Hot-I 2003)
- [10] S. Rao, P. Shenoy, P. Goyal, and H. Vin. Symphony: An integrated multimedia file system. In Proc. of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN'98) (San Jose, CA USA), 1998
- [11] P. Shenoy and H. M. Vin. Cello: A disk scheduling framework for next generation operating systems. In Proceeding of ACM SIGMETRICS Conference, Madison, WI, pages 44-55, June 1998.
- [12] The Storage Networking Industry Association (SNIA). Storage Virtualization I: What, Why, Where and How.
- [13] R. Wijayarathne and A. Reddy. Providing QoS guarantees for disk I/O. *Multimedia Systems* (Springer-Verlag), pages 705-825.
- [14] R. Wijayarathne and A. Reddy. Integrated QoS management for disk I/O. In Proc. Of IEEE Int. Conf. On Multimedia Computing and Systems (ICMCS'99), Florence, Italy, June 1999.
- [15] L. Zang. VirtualClock: A new traffic control algorithm for packet-switched networks. *ACM transactions on Computer Systems*, 9(2): 101-124, 1991.