Available online at www.sciencedirect.com
SCIENCE DIRECT®
Data & Knowledge Engineering 58 (2006) 107–128
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

QoS-oriented negotiation in disk subsystems

Konstantina Stoupa, Athena Vakali *

Department of Informatics, Aristotle University, 54006 Thessaloniki, Greece

Available online 13 June 2005

Abstract

Quality of service (QoS) has emerged as a new term in relation to adding innovation criteria in large scale applications such as network based services, multimedia applications, storage services, etc. QoS has been proposed in storage management towards effective disk space utilization and request servicing. This paper presents a QoS based storage model for effective client negotiation in terms of performance, cost and reliability. Clients can create their own profile with respect to certain QoS attributes in order to specify their profile and requirements. A QoS negotiation model is proposed based on an available disk simulator which is experimented under artificial request workload towards proposing improved system's responsiveness, performance and functionality. Certain remarks and conclusions are raised with respect to meeting the clients's QoS requirements under the negotiated scheduling algorithms, the redundancy scheme and the capacity available to the client's environment according to the client's QoS requirements.

Keywords: Quality of service; Storage subsystems; Attribute managed storage; QoS negotiation

1. Introduction

The term "Quality of service" (QoS) was introduced to describe certain technical characteristics (mainly in communications technology) such as performance, speed and reliability. As demands

* Corresponding author. Tel.: +30 2310 998415; fax: +30 2310 998419.
E-mail addresses: kstupa@csd.auth.gr (K. Stoupa), avakali@csd.auth.gr (A. Vakali).

0169-023X/$ - see front matter
doi:10.1016/j.datak.2005.05.001

for reliable and speedy storage increase drastically, it is effective to introduce QoS in storage subsystems. The increasing need for more efficient and effective storage configurations has become more imperative due to the wide spread of multimedia data which demand great storage capacities together with synchronization and appropriate retrieval. Thus, it is quite important to investigate and evaluate QoS based storage subsystems.

Initially, the idea of QoS emerged to support networks functionality and became one of the most elusive and "confusing" topics not only in the area of networks but in many areas of computer science, since computer society individuals have assigned multiple meanings to this term. It seems that hardware and software vendors, consumers and researchers have their own ideas and definitions of the meaning of QoS, its functionality and its effectiveness. One first definition (by the Reference Model for Open Distributed Processing) states that QoS is "*a set of quality requirements on the collective behaviour of one or more objects*" [25]. A more specialized definition in relation to QoS with multimedia applications is given in [25]: "*Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application*".

Conventional storage subsystems place data in random blocks on the disk. Such an architecture cannot be widely applied for the storage of current applications data (such as multimedia) due to possible unpredictable delays. Thus, new storage architectures and policies should be used in order to succeed in implementing more effective and functional applications. Conventional disk modeling and disk performance issues are discussed in [16,20,13] whereas in [5] disk arrays are proposed as an attempt to maximize the performance and the reliability of the existing storage systems. The main idea was that a storage subsystem could reach those goals by logically grouping multiple disk drives into disk arrays. In such implementations the data array organization is defined by their data distribution schemes and redundancy mechanisms. Disk scheduling policies have been studied in early research efforts (e.g. [22]) whereas it always remain an open research topic due to the importance of disk scheduling in relation to the storage system's functionality and responsiveness [15,19]. A variety of disk scheduling algorithms based on rotational position is discussed in [9] whereas different approaches for continuous media disk scheduling have been reported in [15,21].

Redundant array of inexpensive disks (RAID) are presented in [2,3]. More specifically, in [3] design issues are studied, mathematical models are developed, and the performance of different disk array architectures is examined, for both small and large I/O environments. Furthermore, various redundancy storage models have been proposed, as in the case of using the so called *shadowed* disks [24]. Performance in relation to cost of redundant arrays of inexpensive disks is discussed in [14]. Furthermore, the use of parity disk(s) to serve as a redundancy mechanism for a potential disk and system failure has been widely investigated in earlier research efforts (e.g. [12,18]). A key technique for exploiting the potential for parallel access of the disk array is *striping*. In [11] two striping alternatives are considered Staggered Striping is a striping method and uses the minimum width that can guarantee the continuity of the delay-sensitive data whereas Streaming RAID method [23], when placing delay- sensitive data, uses the maximum width (i.e. equal to the number of disks of the array) to achieve good performance through load balancing and high disk utilization. Furthermore, methods of random data allocation on disks combined with partial replication to order to achieve load balance and high performance have been implemented in [11,17].

Attribute managed storage is discussed in [1,8] where the development of a new storage system is proposed such that the system's core would support user-oriented QoS. In such models a mapping of virtual to physical storage devices is introduced with quality of service guarantees. The mapping of virtual stores on physical storage devices is expected to be optimized in order to achieve balance in system performance against total system cost. Here, we consider the case of attribute-managed storage in order to effectively manage storage resources with respect to the most important characteristics such as scheduling, redundancy and multiple disk configurations. Our goal is to facilitate the storage process by efficient resource utilization. It is quite important for each client to determine the level and the type of QoS that (s)he requires. The clients negotiate with the storage subsystem in order to specify a "profile" of storage medium usage such that each user can utilize storage device(s) according to their needs.

The emergence of such systems has guided the present research effort towards evaluating the QoS negotiation in storage subsystems. Since most organizations avoid buying heavy equipment such as huge storage subsystems, the emergence of digital services "renting" such functionality has been inevitable. In such a case the client should negotiate with the leasing company the QoS characteristics of the storage system to be used. Client may have some QoS requirements related to cost, performance, reliability, etc., under a disk system which has several characteristics. As most clients are not specialized in technical issues, the existence of a user friendly QoS parameters negotiation application intermediating between the client and the disk subsystem seems quite efficient. The configuration of this system is shown in Fig. 1 where the client specifies its requirements which are filtered by the QoS negotiation system. Then a process is employed for mapping the client's QoS requirements to the underlying storage characteristics. Afterwards, the negotiation process leads to a document with the resulting features of the subsystem able to satisfy the needs of a client (possible contract). In case of an agreement, a contract signed by the QoS negotiation system and the client indicates the end of the negotiation process, otherwise a cycle of renegotiation immenses. Under a contract the "leasing" company is obliged to fulfill the terms of it, e.g. the performance, the cost and the reliability levels.

The remainder of the paper is organized as follows: the next section introduces negotiation process between the client and the system in order to agree on the required levels of QoS. Sections 3–5 are devoted to each of the considered phases namely the initialization, the association and the negotiation phases (respectively). Section 6 presents the potential applicability of the proposed work over storage hierarchies. Finally, conclusions and further research topics are highlighted in Section 6.

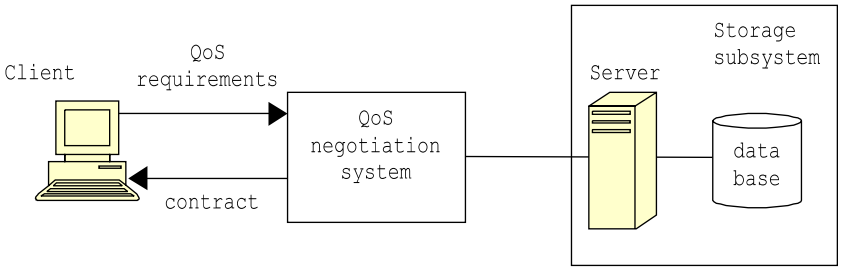

Fig. 1. QoS negotiation process information.

2. QoS negotiation

The idea of QoS can easily be imported in the implementation and usage of a storage system. Certainly such an attempt may be proved really effective nowadays that a variety of applications have emerged, i.e. commercial, scientific, etc. Moreover, much research has been conducted for guaranteeing high QoS levels in multimedia applications which are categorized into: (a) *presentation-based*, such as video-on-demand services, where requirements are really limited, and (b) *interactive applications*, where real-time multimedia communication is obligatory.

The manipulation of the QoS parameters negotiation between the client and the system is really significant process that a QoS-based storage system has to perform. QoS is determined by three indicative issues:

- *Performance*: High performance in storage systems has been associated with minimum response time (i.e. the time period between the generation of a request and its servicing) or minimum seek time (i.e. the time period between the generation of a request and the detection of the position of the first requested data block).
- *Cost*: Performance and cost are rising in accordance, i.e. while performance is maximized so does the cost. Therefore, the evaluation of the total cost of a storage subsystem is really crucial since in another case the user would always ask for best performance. In general, the cost of a storage subsystem is determined by:
  - The number of disks used.
  - The disk model used.
  - The maintenance cost in case of a failure (or avoiding a failure. Therefore, cost is in contrast with reliability, i.e. low cost and high reliability can never meet.

    It is meaningful to examine the cost of a storage architecture against its storage capacity. (Here the cost of a disk system is expressed in US dollars.)
- *Reliability*: The possibility of a disk failure rises as the number of disks increases in a storage subsystem (i.e. the more the disks the higher the possibility of ones failure) [5]. Therefore, in a modern storage system consisting of a disk array we may win in availability and performance but lose in reliability. As mentioned earlier, data redundancy has been widely used to increase data availability and reliability in critical applications and several methods have been proposed to organize redundant data across a disk array (used to recover lost data in the event of a disk failure). Reliability is quantified by the possibility of failure during a given time period and the number of concurrent failures the disk system can survive.

The required level for these characteristics may vary according to the application and the user. As it will be discussed later, the goal of the negotiation system is to suggest a disk subsystem offering the requested performance, cost and reliability levels.

In this paper we consider a negotiation process as the one shown in Fig. 2. The process is divided into three repeated phases:

1. *Initialization phase*: during this phase the client provides information about his/her needs and requirements. Moreover, (s)he specifies whether (s)he is an advanced client (i.e. a client having technical knowledge about storage subsystem features) or a typical one. Such a catego-

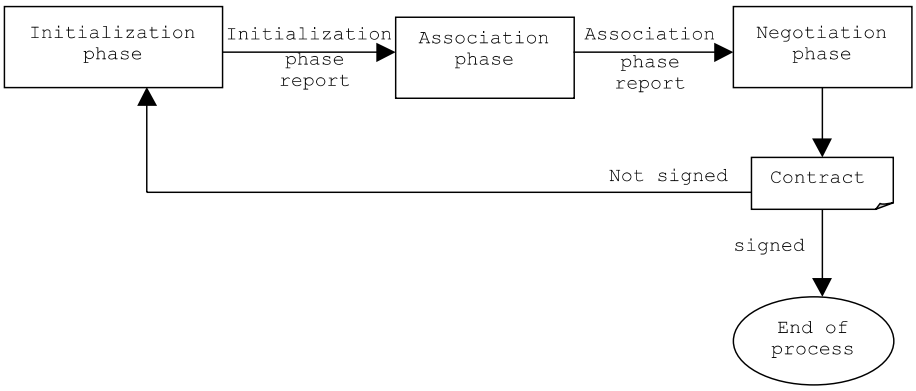

Fig. 2. The negotiation cycle.

rization will be proved useful in the next phases. Afterwards, the QoS negotiation system tries to identify which QoS requirements (e.g. performance, cost, etc.) are more significant according to the presented information. The outcome of this phase is a report which contains a summary of the above information and a list of ascending significance concerning the performance, cost and reliability requirements. That report is internal and it is not presented to the client.

2. *Association phase*: the QoS-based information given by the client is associated with disk system parameters so as to fulfill the QoS requirements (e.g. high performance level). In case the client has declared (her)himself an advanced client, (s)he is given the opportunity to decide by (her)himself on the appropriate parameters through a user-friendly interface. The outcome of this phase is again an internal report which is passed to the next phase mechanism. An advanced client may be able to see this report.
3. *Negotiation phase*: the client is presented with the features of the suggested storage subsystem and the QoS parameters values. In case of an advanced client, a sequence of diagrams are produced showing how a QoS value (e.g. response time) varies according to the chosen scheduling algorithm or data placement technique, etc. The outcome of this phase is an external report, shown to the client, which contains the values of those parameters indicating the levels of the QoS features (e.g. performance is indicated by the response and seek time). The negotiation process ends through the signing of this report (in such a case it plays the role of a contract). If the negotiation fails, then the above activities will be repeated until negotiation succeeds. In this case we result in a renegotiation cycle.

The real burden is to maintain the agreed QoS values. Therefore, the system should continuously monitor the system performance and apply correction mechanisms in order to restore the system to its required condition.

As such a system is client-oriented, it should be user-friendly. Therefore, a careful design of the user interface is quite significant for the success of a QoS-dependent storage subsystem. The best method to be employed is the ''Quality query by example'' suggested in [10], i.e. to hide the internal system QoS parameters and ask the user to decide through a list of QoS examples (as we shall see through graphs or reports).

3. The initialization phase

During this phase the client is asked to provide some useful information which will assist the QoS negotiation system to specify the storage subsystem parameters which will guarantee the requested QoS level. The client input is given under a user friendly interface depicted in Fig. 3 where two possible examples are given. In both cases the clients are advanced and need a storage subsystem used for textual or numeric data. In both cases the workload is read-oriented. As for the application these subsystem are destined to support are differentiated. i.e. in the first case the supported application is commercial one whereas in the second scientific one. Moreover, both systems are rather demanding since they are about to serve real-time applications. Finally, the first system will contain crucial data (i.e. data of high importance).

The first question that the client should answer according to Fig. 3 is whether (s)he is advanced in technical issues concerning disk subsystems or not. Such an information will be proved useful in later phases.

An important issue is whether the storage subsystem is required to manage synchronized (e.g. multimedia) or textual workload. Textual and synchronized workload is considered since it is the most widely used type of workload. Synchronized data is characterized by its continuous nature (e.g. video or audio). As a result it demands a quite different manipulation due the following main characteristics:

- *Real-time storage and retrieval*: Continuous media (CM) devices produce a stream of different media which have to be stored in real-time. Also, CM data should be displayed on time and in certain order. In summary, CM data places time deadlines and demands strict synchronization.

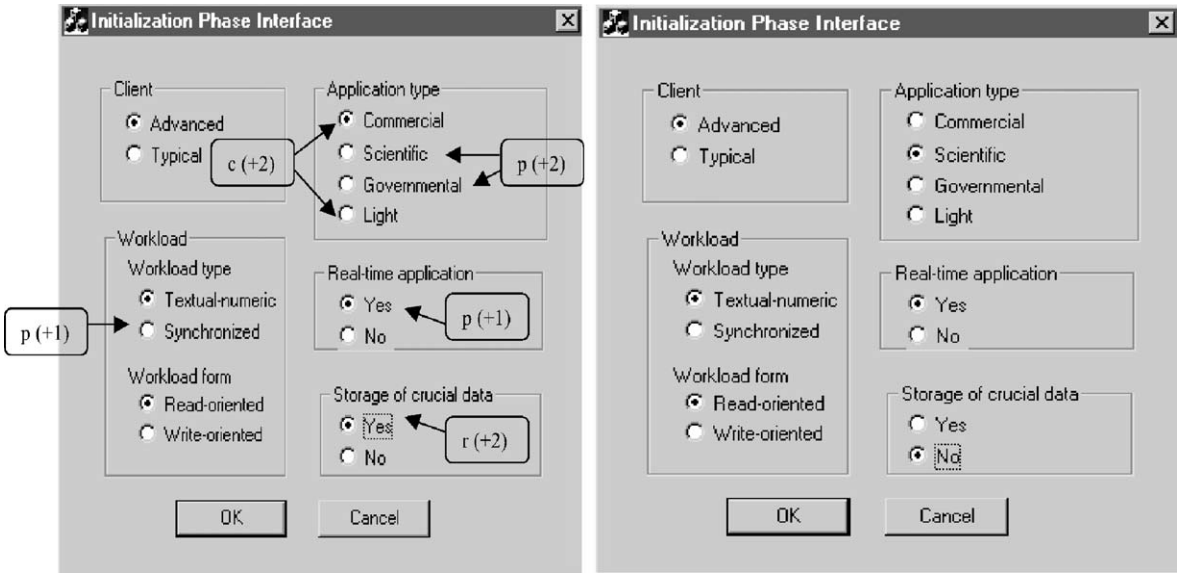

Fig. 3. Initialization phase interface for collecting QoS-related information (two examples).

- *High data transfer rate and large storage space*: Continuous data (such as video) require high transfer rate and high disk storage capacity.
- *Timing requirements*: When the client asks requires for time-dependent multimedia objects the system is obliged to consume and produce data in a constant, gap-free manner (*intramedia synchronization*).
- *Intermedia synchronization*: The different objects (video, audio and text) of a multimedia stream have to be synchronized.

Moreover, we should be given an indication of the type of the workload, i.e. whether it consists mostly of read or write requests. Write requests usually ''suffer'' from lengthy delays since the system remains idle until it receives a confirmation about the completion of the write task. Therefore, it is really important to adopt techniques and methods reducing this delay.

The rest parameters are associated with one or more QoS issues. For example, the type of workload is associated with performance. Table 1 shows that type of association. Based on the type of application emphasis is given to either cost or performance (i.e. minimized response and seek times). For example, in commercial and light applications cost is of great importance, whereas in scientific and governmental applications performance is considered as more significant. Another characteristic that is highly associated to performance is whether the application is real-time one or not since in the first case great concern should be given in performance. Finally, the storage of crucial data demands great deal of reliability.

These associations are quantified by a point-system where each choice is given points concerning performance, cost and reliability. These values are shown in Fig. 3 inside the ellipses. The considered points assignment is used to balance the criteria of performance, cost and reliability. Performance is expressed as p, cost as c and reliability as r. Inside the parenthesis the points that are added to the specific QoS issue are given (e.g. p (+1) means that one point is added to the performance). At the end of the procedure the performance points are added up and so are the cost points and the reliability ones. Afterwards, these QoS issues are listed according to ascending significance and they are written into a distinct internal document which will be passed to the next phase. In our examples, the reports shown in Fig. 4 will be produced. On the top of the document the information given by the client is summarized. The most important part of the document is the list of QoS issues which specify that in the first case the negotiation system should primarily focus on low cost and high reliability and then on high performance whereas in the latter case great concern should be given to performance. Therefore, the appropriate parameters should be chosen in the next phase.

Table 1
Associating client-given information with QoS issues

	Performance	Cost	Reliability
Type of application	+	+	
Real-time application	+		
Crucial data storage			+

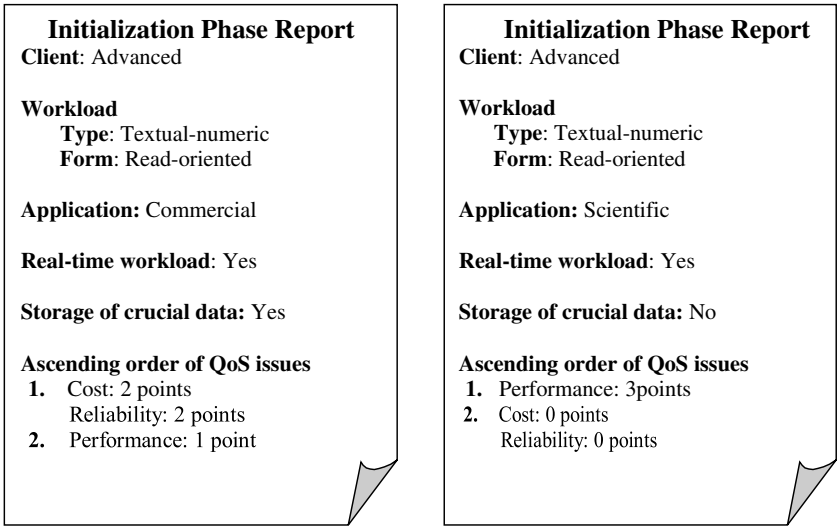

Fig. 4. The initialization phase reports related to the information given in Fig. 3.

4. The association phase

The next phase in the negotiation process is the definition of the storage system parameters in accordance to the client-given information and the report produced in the previous phase. These parameters are categorized in:

1. Disk parameters: which consist of the *data placement* and *redundancy* parameters. Moreover, the *number* and the *models* of the used disks are defined.
2. Scheduling algorithm: the disk scheduling algorithm for servicing requests.
3. Initial delays: According to the workload form (read or write-oriented) parameter, the overhead time and the initial delays are defined.

Table 2 refers to the association between the QoS issues (performance, cost and reliability) and some other characteristics of the application to use such a system (given through the report of the initialization phase) given in the rows, with the storage system parameters given in the columns.

Table 2
Associating client-given information and QoS parameters with storage subsystem parameters

	Disk parameters				Disk scheduling algorithm	Initial delays
	Data placement	Data redundancy	Number of disks	Disk model		
Type of workload	+		+		+	+
Read–write requests	+	+				+
Performance	+	+	+	+	+	+
Reliability		+	+			
Cost	+	+	+	+		

For example, reliability is associated with data redundancy technique and the number of the used disks, i.e. in order to modify the values of the reliability parameters (e.g. possibility of failure, ability of surviving multiple failures) the negotiation system should modify data redundancy techniques and the number of disks.

Disk parameters refer to the definition of the data placement techniques and the redundancy scheme to be used. Two more disk parameters are extremely important during the association phase these are: (a) the number of disks, and (b) the model of the disks used. These parameters are highly associated with the evaluation of the cost of a storage subsystem and moreover, its reliability and performance as shown in Table 2. As the number of disks increases so does the cost, the performance and the reliability. The disk model can also affect those issues.

4.1. Data placement

The term *data placement* defines the mapping of logical addresses used by the host into the disk devices in the subsystem. Such a mapping can be employed under:

- *Data placement for textual workload:*
  - a *random* method, according to which each disk is addressed independently and the mapping from logical disk numbers to physical ones is done directly. The main disadvantage of this method is that it needs system administrators to execute this distribution, when load balancing is required. Therefore, it may be proved time-consuming and thus influences negatively performance. However, it is the most economic method.
  - *Disk striping*: This method proposes the folding of multiple disk address spaces into a single, unified space (seen by the host). This is achieved by considering the distribution of logical data units among the several disks in a round-robin fashion. There are various striping policies usually categorized into the policies where data is distributed in such a way that all disks cooperate to service every request (*Fine-grained striping*) or to policies where disks cooperate only to service large requests and not elsewhere (*Coarse-grained striping*). Unlike independent addressing, fine-grained striping provides perfect load balancing as each disk contains equal fractions of each block. Due to the fact that N disks work on servicing one request the transfer rate is N times greater than that of independent addressing. On the other hand, load balancing is always achieved in coarse-grained striping so it can provide high levels of performance with minimum maintenance costs.
- *Data placement for synchronized workload:* According to intramedia synchronization (i.e. constant gap-free retrieval of the synchronized data) there are four policies [7]: (a) *Interleaved policy* according to which multimedia data is stored on disk in an interleaved way. This approach guarantees smooth, gap-free display and this is a result of smoothing the speed gap between disk and multimedia devices. Thus, it can reduce the buffer requirement significantly (i.e. better performance and lower cost). (b) *Contiguous policy* where data is stored on disk contiguously. This method has been introduced to avoid intrafile seeks that random placement demands. Of course, this approach can achieve high levels of effective bandwidth but it imposes high overhead because it entails enormous copying work during insertions and deletions. This policy is effective for read-only environments. (c) *Constrained placement* which is used not to extinguish the intrafile seeks but reduce them to a reasonable bound. This method is really attractive when

Table 3
Associating data placement techniques with QoS parameters

	Performance	Cost	Type of workload	Read–write requests
Random	–	–	Textual	Both
Fine-grained striping	+	+	Textual	Both
Coarse-grained striping	+	–	Textual	Both
Interleaved policy	+	–	Synchronized	Both
Contiguous policy	+		Synchronized	Read
Constrained policy	–		Synchronized	Both
Log-structured policy	+	+	Synchronized	Write

the block size must be small. Unfortunately, using constrained placement demands also the application of complicated algorithms to ensure that separation between blocks conforms to the required constraints. Moreover, the scheduling algorithm should immediately retrieve all blocks for a given stream before switching to any other stream. A SCAN-like algorithm that orders blocks regardless of the stream they belong to, reduces significantly the drawbacks of constrained placement. (d) *Log-structured placement* has been invented to reduce disk seeks. The main idea of the function of this scheme is that the modified data are not stored to their original positions but in a large contiguous space. Thus, performance levels are increased extremely. From the above it is inferred that log-structured placement is really effective in systems where a lot of modifies take place. However, in a read-only environment such a method imposes large burden because modified blocks may change position.

Table 3 analyzes the various data placement techniques in relation to the associated QoS parameters (Table 2). The "+" symbol indicates an increase whereas "–" a decrease. For example, fine-grained data placement is appropriate for textual data and it improves performance levels (+) but it is not so cost-effective as it leads to rising cost (+). Therefore, it may be chosen in case performance is the first goal to be satisfied (according to the initialization phase report) and minimum cost is not that important.

4.2. Data redundancy

Data redundancy has been widely used to increase data availability in critical applications and several methods have been proposed to organize redundant data across a disk array. Data redundancy consists of either total data replication or the spreading of the data across the disk array along with parity information which can be used to recover missing data in the event of disk failure.

Cost is increased when redundancy is required (in relation to non-redundant systems) while capacity is reduced and performance is degraded. Fortunately individual disk failures are infrequent and total system failures are rare. The problem is initiated by the time we further increase the number of disks. Then the mean time between failure decreases extremely and it seems like non-redundant disk systems are not feasible. This leads to the manipulation of mechanisms to recover data in case of disk failure. For this purpose, two different approaches have been proposed,

both based on data redundancy, and the resulting disk array architectures are known as redundant arrays of inexpensive disks (RAID) systems. Thus, use of redundancy mechanisms is obligatory especially when storing critical data. This mechanisms are divided into two main categories:

- *Data replication*: It is the simplest way to protect data. Two or more copies of the same data are stored on different disks, so that, if one disk fails, the data is still available from the other disks [5,14].
  This type of redundancy is widely known as *disk mirroring*, *disk shadowing and RAID1*. The cost of a RAID architecture is determined by the number of disks so if the disk array consists of N disks then the cost depends on $N-1$. Thus, data replication methods are associated with high costs.
  Performance is affected with the workload format. The disadvantage is that a write function should be applied on every copy. This may cause data loss or increased response time depending on the system. On the other hand, read function benefits from such architectures as N (number of disks = N) requests can be serviced simultaneously which means increasing throughput and decreasing queue times. Moreover, each read request can be scheduled to the disk in which is expected to face the smallest access delay. The improved read performance can decrease the cost/performance ratio accordingly to non-redundant disk systems, even though the absolute cost is multiplied by N. Therefore, data replication is suggested in cases when the workload is read-oriented.
  Several data replication techniques have been introduced but the oldest is the *disk shadowing* where the N disks are divided into two groups each consisting of $N/2$ disks. Therefore, we have $N/2$ pairs of disks where the ith pair is called *mirror i*. Every set of N identical sets can survive $N-1$ failures without data loss. However, each disk failure reduces the performance of a set relative to other sets. This imbalance can cause reduction of the overall performance if a damaged set becomes a bottleneck. A variation of this technique is the *mirrored declustering* where data and copies are striped over the $N/2$ disks. *Chained declustering* (every disk contains both primary and secondary data) and *interleaved declustering* (every disk contains both primary and secondary data but secondary copies are striped across all the disks) have been implemented to minimize the bottleneck phenomenon in case of a failure. With combination of clever scheduling, these methods can maintain a balanced load after a failure, though the probability of surviving multiple failures is reduced. It has been proved that chained declustering leads to higher availability in the shared—nothing environment, i.e. an environment where distinct processes have distinct address spaces.
- *Parity-based protection*: that form of redundancy is used to recover missing data when disk fails. It is based on error detection and correction algorithms. According to coding theory, a lost bit can be recovered from a parity bit, the other protected bits and knowledge on the erasure. Usually this approach does not require a large amount of physical disk space to be reserved for redundancy. Therefore, it is attractive for economic reasons. Unfortunately, such systems performance is still significantly lower than non-redundant disk systems or systems which exploit data replication as in case of a write request it burdens the system with more accesses when a write request arrives. That is due to the unavoidable update function of the parity bits which may lead to very poor performance.

Table 4
Associating data redundancy techniques with QoS parameters

	Total performance	Performance (in case of disk failure)	Cost	Reliability	Read–write requests
Disk shadowing	+	–	+	+	Read
Mirrored declustering	+	+	+	+	Read
Chained declustering	+	+	+	+	Read
Interleaved declustering	+	+	+	+	Read
Dedicated parity disk	–	–	–	+	Both
Striped parity	–	+	–	+	Both
Declustered parity	–	+	–	–	Both

Several methods for distributing parity information across the array have been proposed and analyzed in the bibliography [5]. *Dedicated parity disk* is the most simplified method. There is one disk in the system that contains only parity information. Although, the scheme facilitates the mapping function of logical addresses to physical ones, every write must update the associated bits on the single parity disk. Therefore, this method is suitable for fine-grained striping where only one request is serviced at a time. In case of other distribution schemes the parity disk can minimize performance. *Striped parity* avoids the potential bottleneck that the previous scheme may cause. According to this scheme, the parity block of the first stripe is on disk N, the one of the second stripe is on disk $N-1$ and so on, and data blocks are stored according to increasing values of their index while moving from disk 1 to disk N. By using striped parity the array can perform multiple parity updates in parallel. An extension of the previous method is the *declustered parity* which combines several smaller arrays protected by striped parity. It can recover from a failure more quickly than other schemes but it can only survive a single failure.

In general, data replication presents better performance (especially in read-oriented environments) than parity-based protection but it is less economic and less reliable. Therefore, if performance is more significant data replication will be chosen. Table 4 shows how data redundancy technique influences the levels of performance, cost and reliability and for which applications are appropriate. For example, striped parity renders low levels of performance in general but quite good in case of failure (comparing to other parity-based protection policies), it is quite cost-effective and offers high levels of reliability. Finally, it can serve both read and write requests.

4.3. Scheduling algorithm

The most typical disk scheduling algorithms (i.e. algorithms used to define the sequence of servicing requests arriving to the disk subsystem) used by the conventional storage systems are divided into: (a) *Seek time optimizing algorithms* and (b) *Seek time and rotational latency optimizing algorithms* [19]. The algorithms belonging to the first category focused on the optimization of seek time solely. The simplest scheduling algorithm of this type is first-come-first-served (FCFS) which is considered to be the worst algorithm referring to the average response time. The shortest-seek-time-first (SSTF) was the next step towards improving performance. The main

drawback of SSTF is the larger variance in response times. The most recently introduced algorithm of this type is the V-SCAN algorithm which is similar to SCAN with the difference that the possibility of keeping a direction depends on a variable. In this variable has a value of 0, V-SCAN becomes SSTF, while in case it has a value of 1, V-SCAN becomes SCAN (it always keeps the same direction of search) [6].

The most recent research efforts have focused on the introduction of disk scheduling algorithms minimizing both seek and rotational delays. This move was partially caused by the advances in disk technology which have reduced seek time and not rotational latencies. The shortest-processing-time-first (SPTF) algorithm was meant to serve that need. According to SPTF the request which yields the shortest seek time and rotational latency is chosen to be served first. Unfortunately, SPTF may lead to starvation, leading to very poor response times [26]. The weighted-shortest-processing-time-first (WPSTF) algorithm has been the evolution of SPTF since it keeps SPTFs main idea but also employs an aging function. First, a threshold of maximum accepted delay is assumed. Afterwards, for each SPTF calculation the actual response time is multiplied by a weighting factor (i.e. the time left before the request will exceed the threshold). Therefore, the longer time the request is waiting, the smaller the weighted factor becomes, and thus the request is more likely to be served. WPSTF renders remarkable performance.

Given that multimedia data has to meet deadlines, the earliest deadline first (EDF) algorithm can provide real-time scheduling. This algorithm schedules the data block with the earliest deadline, i.e. the time limit that a block of data should be presented to the client. Thus, there is little possibility that multimedia objects will miss their deadlines. However, such an algorithm may cause great delays and maybe starvation. A variation of the above algorithm with enhanced performance is SCAN-EDF. According to that algorithm the disk head scans the disk surface back and forth and service the requests with the earliest deadline. In case there are more than one requests with the same deadline, their data block are retrieved using SCAN algorithm. SCAN-EDF shows high-performance when there are requests having the same deadline because in another case it shows the same characteristics as EDF.

Another popular algorithm that reduces the drawbacks of the previous ones is grouped sweeping scheme (GSS) which minimizes seek latencies and interstream delays. According to GSS each round is partitioned into groups each containing a number of streams. Groups in a round are serviced always in the same order. Finally, SCAN algorithm is applied in each group. Therefore, if we succeed in optimizing the deriving of groups then we will achieve the best performance [7,15].

Table 5
Associating scheduling algorithms with QoS parameters

	Performance	Type of workload
FCFS	–	Textual
SSTF, VSCAN	+	Textual
SPTF	–	Textual
WPSTF	+	Textual
EDF	–	Multimedia
SCAN–EDF	+	Multimedia
GSS	+	Multimedia

Therefore, a scheduling algorithm is selected firstly according to the type of workload and secondly according to the required performance level. Table 5 shows which scheduling algorithm is for which type of data and how they affect performance.

4.4. Client-oriented negotiations

A final characteristic defined by the client is the form of the workload (whether most requests are read or write), which negotiation subsystem depicts to real values associated with the initial delays.

The result of this phase is a report containing the QoS needs and the recommended parameters able to satisfy them. This report is presented only to advanced clients, otherwise is an internal report which is passed to the next phase. For the needs expressed in the initialization phase report shown in Fig. 4 the report shown in Fig. 5 could be produced. In the first case random data placement technique may be used as it is the most economic. Coarse-grained striping is economic and more effective than random placement. Since high levels of reliability are required striped parity disk is recommended as parity-based protection is general more economic than data redundancy and more reliable. Of course, a dedicated parity disk may also be used as it is more economic than striped parity (due to less maintenance cost). Unfortunately, it renders to worse response times but performance is not the core need of the specific client. The SSTF and SPTF scheduling algorithms are recommended as they render rather high levels of performance, they are common scheduling algorithms and they are appropriate for textual data. Finally, client is recommended to use five disks since it needs both economic and reliable system. The report is supposed to suggest a disk model but we will not refer to a specific model so as not to be accused for advertising a specific brand.

On the other hand, the second client is concerned primarily about performance. Therefore, fine-grained or coarse-grained striping is suggested. Moreover, data replication methods are proposed due to their better performance. Two or three copies may be used since minimizing cost is not one of the goals of this client. Finally, WPSTF is suggested as it is the most effective for every form of textual workload.

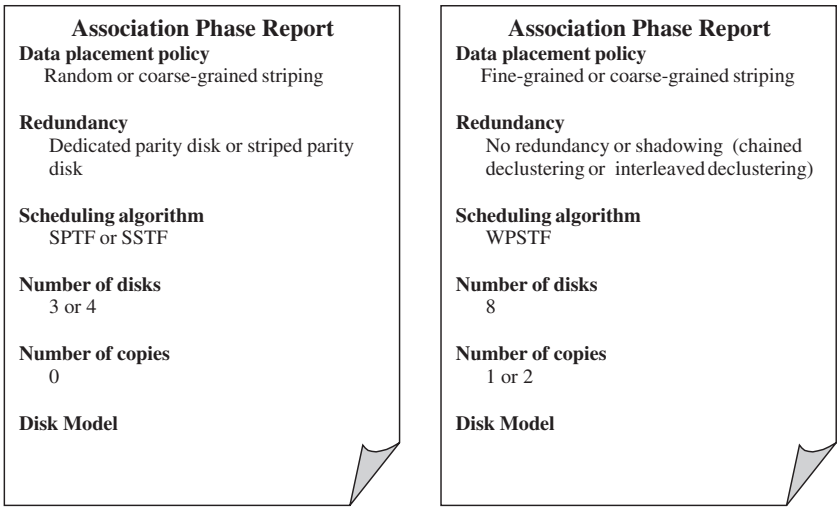

Fig. 5. The association phase reports related to the information given in Fig. 3.

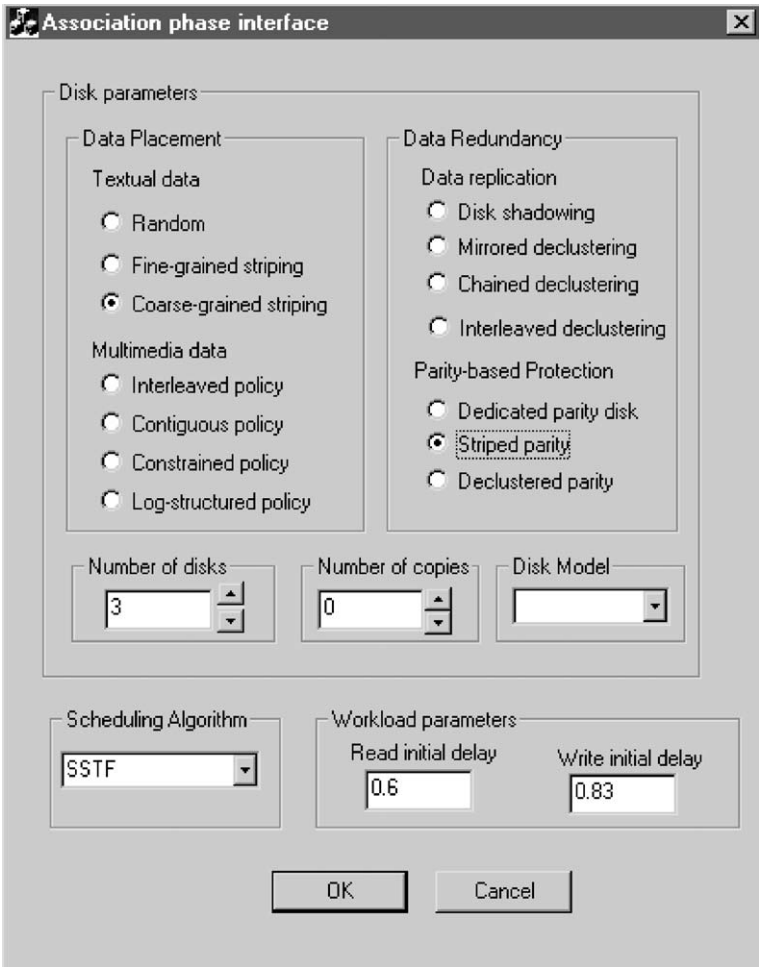

Fig. 6. Association phase interface for an advanced client.

In case of a specialized client, this report is presented and, moreover, an interface in order to define (him)herself the requested storage system parameters. Fig. 6 depicts that interface filled in with the above discussed parameters referring to the first example.

5. The negotiation phase

The goal of the negotiation phase is to use the recommended sets of parameters in a simulation environment so as to test their effects. Of course, the client may ask for the simulation of a specific set, otherwise all of them are tested. The tool used for such a task is the *DISKSIM simulator*.

DiskSim is an effective, strong disk system simulator implemented by Ganger et al. [4]. Its effectiveness, results from the fact that it includes detailed modules for the most significant components of a disk storage subsystem. The components that are emulated are: disks, controllers,

buses and disk block caches. Their configuration is very detailed as it involves a large number of parameters. We experiment with the adequate ones of them in order to assess the several data placement schemes, data topologies and scheduling policies. DiskSim major advantage is its ability to be used as part of a larger simulation environment and the output provides valuable results about the disk subsystem (such as response time, seek time, etc.). One last feature of DiskSim is that it can work by using either traces or internally generated synthetic workload. Therefore, an analyzer is able to evaluate real workload by exploiting existing traces and to produce simulation results which will be the outcome of the service of synthetic workload. It has been proved [4] that the results are really similar, a fact that exacerbate this tool's power and capabilities.

After the recommended parameters sets are simulated with the help of DISKSIM, a results report is produced containing the performance, cost and reliability results (e.g. response time, seek time, total cost, failure frequency, ability of surviving multiple failures, etc.) as the one shown in Fig. 7. That report will play the role of a contract in case it will be signed by both the client and the storage subsystem. In case of advanced clients, graphs are also produced depicting the results of the tests so as to choose (him)herself the required set of parameters describing the storage subsystem.

The client may not consent to the QoS parameters values. In such a case the whole negotiation cycle is repeated until it results to a contract acceptable by both parts which will be finally signed by the two peers.

5.1. Negotiation experimentation

As it has already been mentioned, in case of an advanced client this phase produces graphs depicting the performance characteristics of the suggested parameters. The aim of this task is

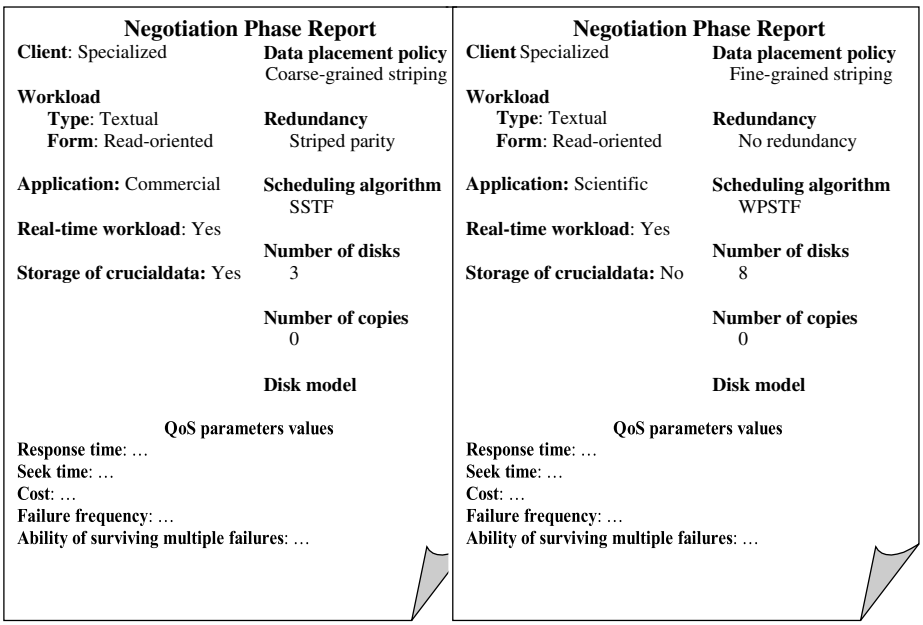

Fig. 7. The negotiation phase reports (possible contracts) for the two cases.

to aid client in choosing the storage system that will perfectly fit (his)her needs. In order to test the proposed storage subsystem parameters for the two cases, we used a workload with the following characteristics:

- The number of requests was 10,000 and the used disk model was HP-C2490A.
- It included 80% of read requests and 20% of write requests.
- There were included sequential read/write requests. Seven percentage of the read requests was sequential and 4.8% of the write requests.
- All of the requests were time-critical.
- The average size of the requests was 10.16678 with a variation 10.423334.
- In case of shadowing one copy is used (i.e. if the storage system consists of two disks, the one contains the original data and the other plays the role of the shadow).

Indicative results of the experimentation are depicted in Fig. 8 which can justify the employment of the storage parameters referred to the possible contract. More specifically, Fig. 8(a) has the curves of response time with respect to the number of disks when different scheduling algorithms are used under a 10,000 requests workload. As depicted in this figure, FCFS algorithm shows the worst response time as it was expected. This performance is extremely obvious in the case of two disks where the FCFS response is around 51 ms when the SSTF, SPTF and VSCAN response time is between 45.5 and 47.5 ms. Furthermore, in case of a non-redundant system, the WPSTF algorithm renders a response time around 49 ms which is quite disappointing as its performance is similar to that of FCFS. In the first case, that performance is of not that much significance, both FCFS and SSTF algorithms may be used. A really interesting feature is the sharp fall of response time from 2 disks to 4 disks. All of algorithms decrease their response times by about 50%. Thus, the more disks a subsystem has the more effective it is. Of course, after a certain number of disks the decrease in response time is not so great as to balance the increase in cost, the difference in response time for a subsystem with 12 disks and one with 14 disks is only 0.5 ms. Therefore, for the second case a system with 8 disks is proposed whereas in the first case 3. Another remark is that the response times for the various scheduling algorithms are becoming equal as the number of disks is increased.

Fig. 8(b) tests the redundancy schemes since it shows the curves of response time when using different redundancy schemes. It is obvious that shadowed and no redundancy exhibit much better performance than parity disk redundancy but the first are less reliable and less cost-effective. Therefore, in the first case where reliability is important, parity disk redundancy is proposed, whereas in the second case no redundancy as the goal is high performance and low reliability. Furthermore, in the second case, when 8 disks are suggested to be used both methods render similar response times. It is really interesting that parity disk redundancy results in almost double response time compared with shadowed redundancy.

Fig. 8(c) is related to a subsystem which uses striped data placement and various types of redundancy. Our choices are to use ''No redundancy'', ''Shadowed redundancy'' and ''A dedicated parity disk''. As it is obvious, when parity-disk redundancy is used response time increases significantly. This was expected because as discussed in Section 3, although parity information redundancy provides high protection levels, it burdens response time due to the overhead that write requests impose. On the other hand, response time of shadowed redundancy is almost equal

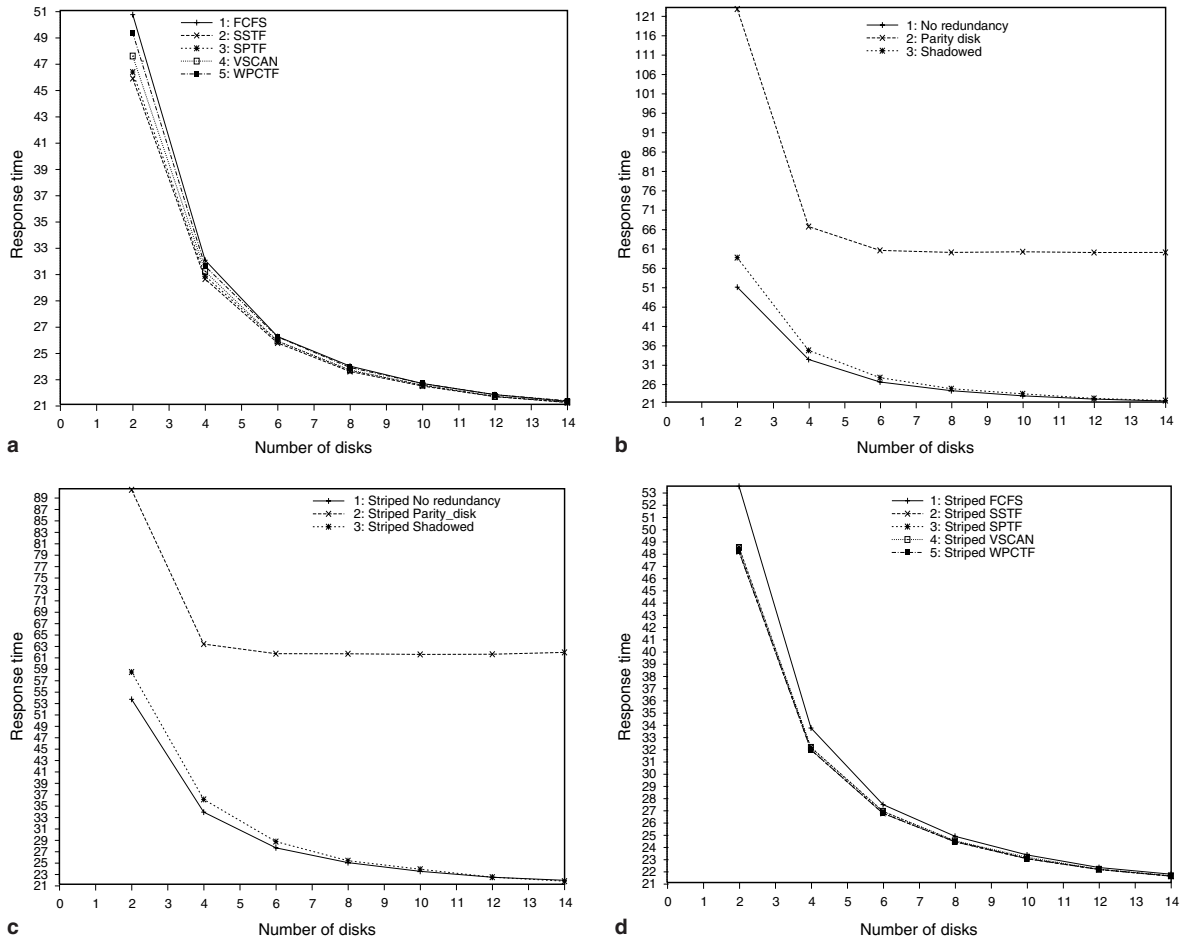

Fig. 8. (a) Response time for different scheduling algorithms under a 10,000 request workload, (b) response time for different redundancy schemes under a 10,000 request workload, (c) response time for different striped data under different scheduling algorithms under a workload of 10,000 requests, and (d) response time for different striped data under different scheduling algorithms under a workload of 10,000 requests.

to that of a disk subsystem without redundancy. Therefore, we can gain much profit by using shadowed redundancy with almost no impact on response time. Furthermore, parity redundancy shows almost the same performance for subsystems with four or more disks. Thus, it is not advisable to use many disks with this type of redundancy, as we will increase cost without increasing performance. For this in the first case only 3 disks are suggested. By comparing this figure with Fig. 8(b) one can notice that the combination of striped data placement with parity-based protection is more effective than random placement with parity-based protection that is why it is recommended in the first case (response time decrease of almost 1 ms for 5 disks). In the second case, striped data placement in conjunction with no or shadowed redundancy is recommended since it renders better response times comparing to random data placement.

Fig. 8(d) has the response time of striped placement under various scheduling algorithms. The curves of response time for striped placement involve using FCFS, SSTF, VSCAN, WPCTF. The interesting point is that all of the algorithms (except for FCFS) lead to (almost) equal response times. Again, a great decrease is obvious when switching from a disk system containing two disks to a one containing four disks. As the number of disks increases, response time faces little improvement. In case of the first client when 3 disks will be used it is important to use SSTF and not FCFS as for this number of disks there is a performance gap of about 2 ms between FCFS and the rest algorithms.

By studying the above graphs the second client may object to the suggested contract since no redundancy and shadowed redundancy for a system with 8 disks renders similar response times but shadowed redundancy, moreover, offers reliability. In such a case the negotiation cycle is executed again.

6. Extensions to storage hierarchies

Currently, due to heavy storage needs, a group of disks is not always efficient. Frequently, other peripheral storage mediums are used, such as tapes, cd arrays, etc. Therefore, a logical storage hierarchy can be formed. A storage hierarchy consists of various levels form which the first one is a local memory or cache which provides high speed but temporary storage. The second level is the whole group of local memories of every computer in the distributed network. Finally, the third level is every permanent storage device, such as hard disks, tapes, optical disks, etc. The format of such hierarchy is shown in Fig. 9.

The proposed negotiation environment can also be extended to support the whole storage hierarchy and not only the hard disk subsystem. Therefore, according to the client's preferences regarding the permanence and performance of storage, the negotiation system can decide whether

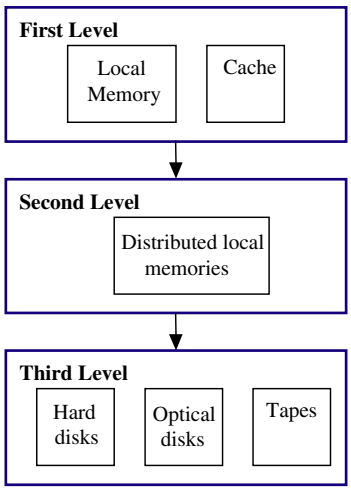

Fig. 9. A storage hierarchy.

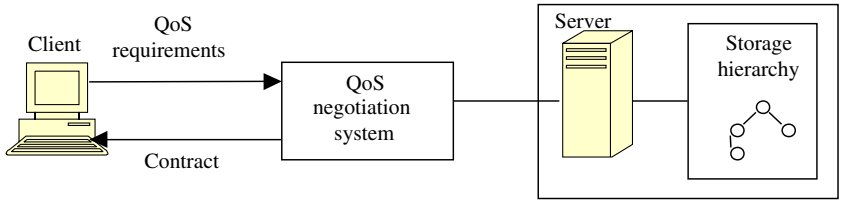

Fig. 10. The extended QoS negotiation architecture.

it will employ the first, second or third storage level and which device of the specified level. Thus, the extended system can have the format of Fig. 10. Of course, the most used level will be the third one since the modifications of the devices belonging to it does not highly affect the function of the machine (something not right for the first level).

7. Conclusions—future work

In this paper, QoS has been introduced towards improving performance and utilization under a considered storage topology. The adoption of a negotiation interface has been proposed to enhance the performance and functionality of a disk subsystem towards user request servicing facilitation and effectiveness. A disk simulator has been used to experiment under varying storage subsystem parameters and certain conclusions were discussed about the proposed QoS parameters specification. The negotiation cycles and the client demands specification has been proven quite beneficial with respect to the overall response times.

Further work can consider more complicated storage subsystems which should involve storage hierarchies such as caching, disks and tapes. It will be quite beneficial to employ QoS parameters at all storage levels in order to result in more effective user request servicing. Furthermore, the system could store client profiles in order to prevent them from repeating their QoS requirements and each client should be informed of the effectiveness of their choices in order to re-negotiate on their QoS parameters specification.

References

[1] E. Borowsky, R. Golding, A. Merchant, L. Schreier, E. Shriver, M. Spasojevic, J. Wilkes, Using attribute-managed storage to achieve QoS, HP Technical Report, 1994.
[2] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, D.A. Patterson, RAID: high performance, reliable secondary storage, ACM Computer Surveys 26 (2) (1994) 145–185.
[3] S. Chen, D. Towsley, A performance evaluation of RAID architectures, IEEE Transactions on Computers 45 (10) (1996) 1116–1130.
[4] G. Ganger, B. Worthington, Y. Patt, The DiskSim Simulation Environment. Available from: <http://www.ece.cmu.edu/ganger/disksim>, 2003.
[5] G.R. Ganger, B.L. Wothington, R.Y. Hou, Y.N. Patt, Disk arrays: high-performance, high-reliability storage subsystems, IEEE Computer 27 (3) (1994) 30–36.
[6] R. Geist, S. Daniel, A continuum of disk scheduling algorithms, ACM Transactions on Computer Systems 5 (1) (1987) 77–92.

[7] J. Gemmell, H.M. Vin, D.D. Kandlur, P.V. Rangan, L.A. Rowe, Multimedia storage servers: a tutorial, IEEE Computer 28 (5) (1995) 40–49.
[8] R. Golding, E. Shriver, T. Sullian, J. Wilkes, Attribute-managed storage, Workshop of Modeling and Specification of I/O MSIO, San Antonio, TX, 1995.
[9] D.M. Jacobson, J. Wilkes, Disk scheduling algorithms based on rotational position, HP Laboratories Technical Report, HPL-CSP-91-7rev1, 1995.
[10] G. Kalkbrenner et al., Quality of service (QoS) in distributed hypermedia systems, in: Proc. of the 2nd Int. Workshop on Principles of Document Processing, 1994.
[11] J. Korst, Random duplicated assignment: an alternative to striping in video servers, in: Proc. of the 5th ACM Int. Conf on Multimedia, 1997, pp. 219–226.
[12] E.K. Lee, R.H. Katz, Performance consequences of parity placement in disk arrays, in: Proc. of the 4th ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems, vol. 19, No. 2, 1991, pp. 190–199.
[13] S.W. Ng, Advances in disk technology, performance issues, The IEEE Computer 2 (2) (1998) 75–81.
[14] F. Quaglia, B. Ciciani, Performance vs. cost of redundant arrays of inexpensive disks, Simulation Practice and Theory 7 (1999) 153–170.
[15] Y. Rompogiannakis, G. Nerjes, P. Muth, M. Paterakis, P. Triantafillou, G. Weikum, Disk scheduling for mixed-media workloads in a multimedia server, in: Proc. of the 6th ACM Int. Conf. on Multimedia, 1998, pp. 297–302.
[16] C. Ruemmler, J. Wilkes, An introduction to disk drive modeling, IEEE Computer 27 (3) (1994) 17–28.
[17] J.R. Santos, R. Mutz, Performance analysis of the RIO multimedia storage system with heterogeneous disk configurations, in: Proc. of the ACM Int. Conf. on Multimedia, 1998, pp. 303–305.
[18] E.J. Schwabe, I.M. Sutherland, Flexible usage of parity storage space in disk arrays, in: Proc. of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures, 1996, 99–108.
[19] M. Seltzer, P. Chen, J. Ousterhout, Disk scheduling revisited, USENIX Winter Technical Conference, 1990, pp. 313–323.
[20] E. Shriver, Performance modeling for realistic storage devices, Ph.D. thesis, Computer Science, New York University, 1997.
[21] R. Steinmetz, Multimedia file systems survey: approaches for continuous media disk scheduling, ACM Computer Communications 16 (3) (1995).
[22] T.J. Teorey, T.B. Pinkerton, A comparative analysis of disk scheduling policies, Communications of the ACM 15 (3) (1972) 177–184.
[23] F.A. Tobagi, Streaming RAID—a disk array management system for video files, in: Proc. of the 1st ACM Int. Conf. on Multimedia, 1993, pp. 393–400.
[24] A. Vakali, Y. Manolopoulos, An exact analysis on expected seeks in shadowed disks, Information Processing Letters 61 (1997) 325–329.
[25] A. Vogel, B. Kerhervé, G. von Bochmann, J. Gecsei, Distributed multimedia and QoS: a survey, IEEE Multimedia 2 (2) (1995) 10–19.
[26] B.L. Worthington, G.R. Ganger, Y.N. Patt, Scheduling algorithms for modern disk drives, in: Proc. of the ACM SIGMETRICS Conf. on Measurement and Modelling of Computer Systems, vol. 22(1), 1994, pp. 241–251.

Konstantina Stoupa received her B.Sc. degree in Computer Science from the Department of Informatics at the Aristotle University of Thessaloniki where she is currently a Ph.D. graduate student. She has also received the M.B.A. degree from the University of Macedonia in Thessaloniki, Greece. She also works as a teacher in the department of Information Management of Technological Education Institute in Kavala, Greece. She has published papers on Access Control using XML and Storage Subsystems in international journals and conferences. Her primary research interests include Access Control especially in distributed, widely accessed environments.

Athena Vakali received a B.Sc. degree in Mathematics from the Aristotle University of Thessaloniki, Greece, a M.Sc. degree in Computer Science from Purdue University, USA (with a Fulbright scholarship) and a Ph.D. degree in Computer Science from the Department of Informatics at the Aristotle University of Thessaloniki. Since 1997, she is a faculty member of the Department of Informatics, Aristotle University of Thessaloniki, Greece (currently she is an Assistant Professor). Her research interests include design, performance and analysis of storage subsystems and data placement schemes for multimedia and Web based information. She is working on Web data management and she has focused on XML data storage issues. She has published several papers in international journals and conferences. Her research interests include storage subsystem's performance, XML and multimedia data, management and data placement schemes.