Access Time Estimation for Tertiary Storage Systems

Darin Nikolow[1], Renata Słota[1], Mariusz Dziewierz[1], and Jacek Kitowski[1,2]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059, Cracow, Poland
[2] Academic Computer Centre CYFRONET AGH, Cracow, Poland,
{darin, rena, kito}@uci.agh.edu.pl
phone: (+48 12) 6173964, fax: (+48 12) 6338054

Abstract. We propose two approaches for estimating the Tertiary Storage System access time: open approach and gray-box approach. In the first case the source code of the storage system is available, so changes in the code are made by adding event reporting functions. In the second approach - the essential system information is accessible via its native tools only. In this paper we describe an implementation of the open approach for access time estimation. The second approach we have shortly described as our future work.

1 Introduction

As the requirements for storage capacity grow exponentially each year many applications of different types (e.g. archiving, backup, scientific, DBMS, and multimedia) make use of Tertiary Storage Systems (TSS). The access time of a file stored on the TSS can vary a lot: from few seconds to hours. This depends mainly on the system load at the time of issuing a request, but other parameters like location of data on the storage medium, transfer rates and seek times of the drives are also important. In some cases a priori knowledge of the access time is essential, e.g., in the case of a Grid data replication system [1,2]. This will allow more efficient usage of storage resources and will decrease the overall latency times. In addition, the user's satisfaction increases when the service time of his or her request is predicted (e.g. user waiting to watch a selected video sequence requested from a near-on-demand video server; administrator recovering from backup).

Estimating the access time of a request for a given TSS state by using a single analytical function is not applicable here because of the algorithmic nature of the request processing. The Queuing Systems Theory could be used to compute analytically the average access time of requests for a certain TSS state. The goal of this study is to develop a method for accurate estimating the latency time of a given request to the TSS. Since the analytically computed mean values are not sufficient in many cases, the event simulation approach to estimate the latency for a real system is adopted.

B. Monien and R. Feldmann (Eds.): Euro-Par 2002, LNCS 2400, pp. 873–880.

We propose two approaches for estimating the TSS access time:

- **Open TSS Approach**, in which the source code of the TSS is available, so event reporting functions can be introduced, and
- **Gray-Box TSS Approach**, in which the essential system information is accessible via its native tools only.

In this paper we describe an implementation of the open TSS approach for access time estimation of own developed FiFra (File Fragmentation) TSS. The second approach is shortly presented as our future work.

The rest of the paper is organized as follows. The next section presents some related works. The third section describes briefly FiFra TSS. The fourth one represents design and implementation details as well as experimental results for the open TSS approach. Future work is presented in the following section and the last one concludes the paper.

2 Related Work

Rodney Van Meter in [3] proposes Storage Latency Estimation Descriptors (SLEDs) as a method of supplying to the client a predictive information about the I/O performance of the underlying storage systems. By using SLEDs the application can be more efficient by rescheduling its I/O calls in such a way that the less expensive (for instance cached) I/O calls are invoked first. Rodney Van Meter and Minxi Gao in [4] implement SLEDs for the Linux operating system and show significant performance improvement of the applications modified to take advantages of SLEDs. In their implementation the I/O performance estimation is based on latency and bandwidth measurements done during the boot process for each storage device attached to the system.

Shen et. al in [5,6] present a multi-storage architecture and a performance prediction method to increase I/O efficiency of scientific applications. They also developed a run time library on top of Storage Resource Broker (SRB) [7] for optimizing tertiary storage access. Their prediction algorithm is based on time measurements of basic SRB file operations (open, seek, read, close, etc.) and assumes that the file is in the disk cache. Therefore, they do not concentrate on prediction of the staging time for data located on tertiary storage.

3 Description of the FiFra TSS

3.1 Background

Since files stored on TSS get bigger the problem of efficient access to fragments of them arises. This is essential for systems which require that the latency is kept below a certain limit, like continuous media stream servers for instance. The problem of efficient access to fragments of files stored on TSS is one of the subjects covered in the task which we will carry out within the CrossGrid project [2].

A TSS capable to access video sequences called Video Tertiary Storage System (VTSS) [8] was developed during our previous work. Next, based on VTSS, a more general version of the system (allowing access to plain file fragments) was developed. We called it FiFra (File Fragmentation) TSS.

The access to the data stored in the FiFra TSS is sequential. Examples of applications with sequential data access are: restoring data from backup copy, CD-image archiving, anonymous ftp server, serving software depots, applications using multimedia data (video, audio, image), logging systems, etc. Some of these applications will probably need to access fragments of the files, e.g., playing interesting parts of a video file, extracting log records for a certain period of time or continue an interrupted file transfer.

3.2 Architecture of the FiFra TSS

The architecture of FiFra TSS is shown in Fig. 1. The system consists of two main daemons: the Repository Daemon (REPD) and the Tertiary File Manager Daemon (TFMD). Since the FiFra TSS is based on the VTSS described in [8] only brief descriptions of REPD and TFMD are given.

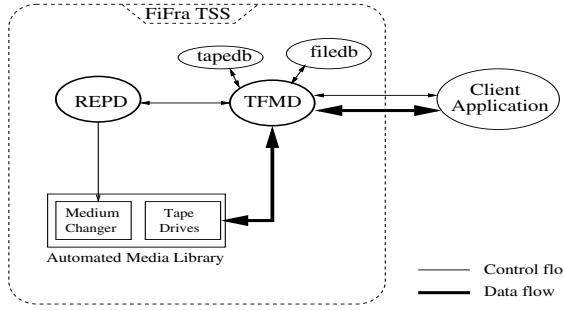

Fig. 1. Architecture of the FiFra TSS.

REPD keeps repository information in its internal data structures. When a mount request is received REPD issues appropriate SCSI commands to the robot arm of the library. TFMD manages information about files and media and transfers the files from the removable media devices to the client. In the case of tapes the files should be stored with the hardware tape drive compression turned off if we want to enable a direct file fragment retrieval from tape.

Since client requests (like write a new file or read a file fragment) can compete for access to the storage resources they are served according to the FIFO strategy. The most often used request is expected to be the read fragment request.

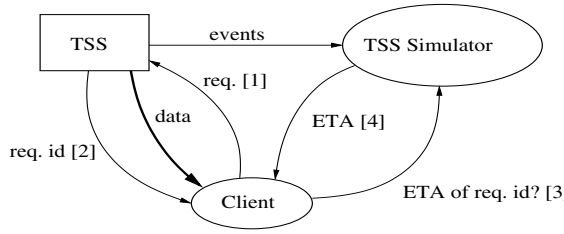

Fig. 2. Open TSS approach.

4 Open TSS Approach for Access Time Estimation

4.1 General Design

The open TSS approach shown in Fig.2 is based on simulation of the TSS in order to obtain the ETA (Estimated Time of Arrival) for a given request processed by the TSS. ETA in this study is defined as the startup latency imposed by the tertiary storage hardware and its managing software. In other words it represents the local startup latency (the network influence is not taken into account). The real TSS has to be changed to report essential events to the TSS simulator. The calling sequence is mentioned by square brackets.

In this approach the Client issues its request to the TSS and receives an identifier for that request. While waiting for the data to come the Client can ask the TSS Simulator for the ETA of its request. Another possibility (just for prediction) is to ask for the ETA before actually issuing a request (not presented in the figure).

In this approach we mainly concentrate on TSS equipped with DLT drives, due to their complex access time model. The processing of a request by the TSS goes through subsequent states, triggered by the events shown in Fig.3. The most probable path is mentioned thicker. The request processing goes to *Waiting* state

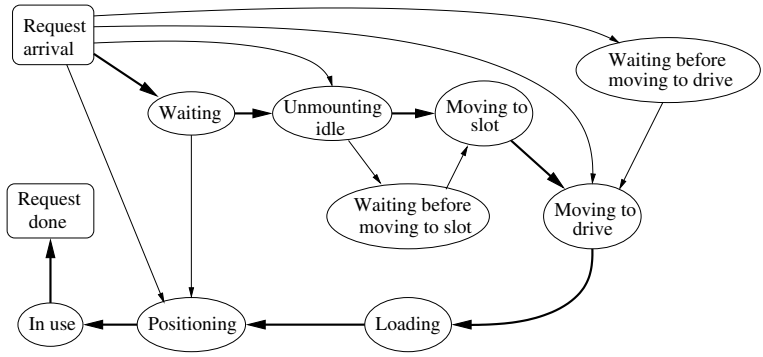

Fig. 3. State transition diagram of a request processed by the TSS.

if there are no resources (drive or tape) available to proceed further. The state *Unmounting idle* means preparing an idle tape for ejecting. The state *Moving to slot* indicates that the idle tape is being moved to an empty slot. If the robot arm is busy serving another request when *Unmounting idle* is finished then the *Waiting before move to slot* state will be visited. The next state *Moving to drive* points that the needed tape for the current request is being moved to an empty drive. *Loading* represents loading the tape into the drive. The state *Positioning* indicates that the tape is being positioned and the state *In use* means that the tape is being read or written. When the transfer is finished the tape becomes idle and the request is done.

Shortcuts of the described path are possible for certain cases: for instance if the needed tape is already mounted then the request processing starts from the *Positioning* state.

The simulation algorithm is based on the state transition diagram shown before. Based on previous measurements the time of completion for each state (except the *Waiting* state) is estimated. It can be fixed or dependent on such parameters like position of the tape, size of fragment, transfer rate. The completion time of the *Waiting* state is estimated by simulating processing of the previous requests, which have possibly occupied resources for which the current request is waiting. The overall estimation is done by summing up the completion times for the states passed through during the processing.

4.2 Implementation

For the FiFra TSS implementation of the open approach an additional daemon, called SIMUD (Simulator Daemon) is introduced. It communicates with REPD, TFMD and the client application. At its initializing phase SIMUD requests information from REPD about the current state of the TSS and starts waiting for events. REPD has been modified by adding a new command `sendstat`, which is used by SIMUD to retrieve the initial TSS state information. Event reporting functions have been added to the source code of the REPD and TFMD daemons. The reporting is done via dispatching an appropriate command to SIMUD. This command can be `NEWREQ` reporting a new request, `STATCH` reporting state change of request, `DELREQ` reporting that a request has been finished. The client can ask SIMUD to simulate ETA of a given request by using `SIMETA` command.

4.3 Preliminary Results

Comparisons between the latency times for the real and simulated systems are presented below. Measurements were done for a sequence of 100 requests, generated every 60 or 100 seconds, according to the zipf distribution. The simulated TSS was configured with two DLT tape drives.

The results presented in Fig.4 show that the TSS is overloaded, because the latency time of the subsequent requests tends to be higher. The characteristics for the real and the estimated startup latency are similar - the fluctuations occur for the same request number. In Fig.5 where the interval between requests

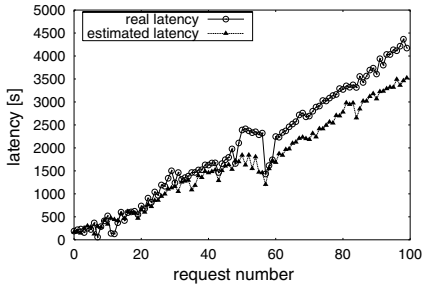

Fig. 4. Comparison between the real and estimated startup latency (with 60 seconds interval between requests).

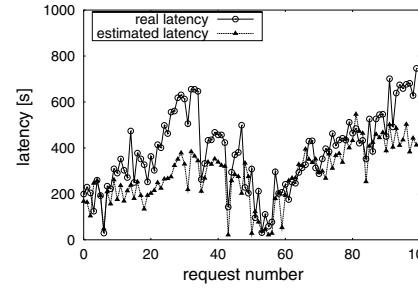

Fig. 5. Comparison between the real and estimated startup latency (with 100 seconds interval between requests).

is higher the system can handle the coming requests better (with no increasing tendency). In this case the characteristics are also similar but the relative error is higher.

In Fig.6 and Fig.7 the histograms of the relative error of the estimated latency are presented. For the overloaded system (Fig.6) about 80% of the estimations are obtained with the error below 20%. For the latter case (see Fig.7) 80% of the estimations have error below 45%. The relative error in this case is higher due to smaller latency absolute values, because for only few requests in the queue the positioning time error can vary a lot. This feature follows from the ideal DLT tape positioning model implemented in this study. The difference is higher especially when the block to position to is far from the beginning of the tape. When there are more requests this error is smaller because of averaging of positioning times. More accurate results could be obtained by using the low cost access time model for serpentine tape drives proposed in [9] which we plan to use.

5 Future Works

The future work concentrates on the estimation of the access time for tertiary storage systems in which no changes of the source code are allowed. Our first target is the UniTree HSM system with the gray-box approach implemented. Estimation of the access time will be based on knowledge about UniTree HSM system operations and on various data gathered from the available utilities delivered by the vendor.

In Fig.8 the mentioned approach is presented. The core of the system is the TSS Simulator similar to that presented in Section 4. Again, the square brackets represent the calling sequence. TSS Monitor will collect the necessary data about the UniTree state. Request Monitor & Proxy catches client requests in order to

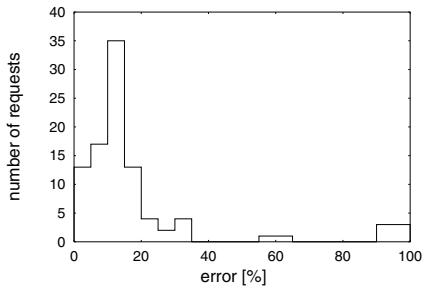

Fig. 6. Histogram of relative error of the estimated latency (with 60 seconds interval between requests).

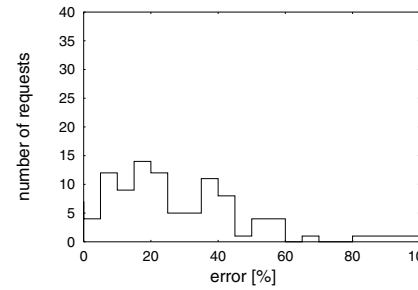

Fig. 7. Histogram of relative error of the estimated latency (with 100 seconds interval between requests).

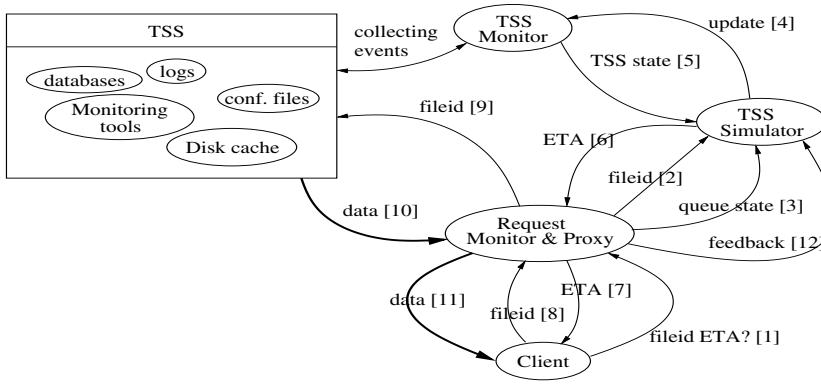

Fig. 8. Gray-box TSS approach.

get more detailed information (queue order, file identifiers, time statistics) about the requests being processed by the UniTree.

We also plan to check (and change eventually) the DLT access time model for the cases when the hardware compression is on (which is the usual case).

6 Conclusions

In this paper we have focused on the problem of estimating the access time of data stored on the tertiary storage. We implemented the access time estimation method for a TSS (developed at our site) using the open system approach. The system has been tested and preliminary results obtained. The results show that the estimation errors are lower when the TSS is overloaded. A further improving of the system is necessary since our goal is to estimate the latency time more accurately even if there is only one or few requests in the queue. This will be done by implementing better access time model for the DLT tape drives.

Further increasing of the accuracy will be obtained by changing the simulator to automatically tune itself based on comparing each state completion estimated time with the real one.

The lessons learned during the implementation, testing and tuning of the open TSS approach are useful for the gray-box TSS approach.

The presented work concentrates on TSS with DLT drives, which seem to have the most complicated access time model compared to the other tertiary storage devices. Adding support for the other devices, like magneto-optical drives will imply just simplifying of the DLT model.

The TSS access time estimation could improve the efficiency of the Grid replica selection and migration services. The TSS Simulator could be used to supply information about the TSS state to the Grid monitoring services for optimizing data access for the Grid applications.

Acknowledgements

The work described in this paper was supported in part by the European Union through the IST-2001-32243 project "CrossGrid". AGH grant is also acknowledged.

References

1. Vazhkudai, S., Tuecke, S., Foster, I., "Replica Selection in the Globus Data Grid", in Proc. of the IEEE International Conference on Cluster Computing and the Grid (CCGRID 2001), Brisbane, Australia, May 2001.
2. "CROSSGRID - Developement of Grid Environment for Interactive Applications", EU Project no.: IST-2001-32243.
3. Meter, R., V., "SLEDs: Storage latency estimation descriptors", In Ben Kobler, editor, in Proc. 6th NASA Goddard Conference on Mass Storage Syst. and Tech. in Coop. with 15th IEEE Symp. on Mass Storage Syst., pp. 249-260, March 1998.
4. Meter, R., V., Gao, M., "Latency Management in Storage Systems", in Proc. of the 4th Symp. on Operating Syst. Design and Implementation (OSDI'00), October 2000.
5. Shen, X., Choudhary, A., "A Distributed Multi Storage Resource Architecture and I/O Performance Prediction for Scientific Computing" in Proc. 9th IEEE Symp. on High Performance Distributed Computing, pp.21-30, IEEE Computer Society Press, 2000.
6. Shen, X., Liao, W., Choudhary, A., "Remote I/O Optimization and Evaluation for Tertiary Storage Systems through Storage Resource Broker", in IASTED Applied Informatics, Innsbruck, Austria, February, 2001.
7. Baru, C., Moore, R., Rajasekar, A., Wan, M., "The SDSC Storage Resource Broker", in Proc. CASCON'98 Conference, Toronto, Canada, Dec. 1998.
8. Nikolow, D., Słota, R., Kitowski, J., Nyczyk, P., Otfinowski, J., "Tertiary Storage System for Index-based Retrieving of Video Sequences", Lecture Notes in Computer Science, **2110**, pp.435-444, Springer, 2001.
9. Sandstå, O., Midstraum, R., "Low-Cost Access Time Model for Serpentine Tape Drives", in Proc. of 16th IEEE Symposium on Mass Storage Systems the 7th NASA Goddard Conference on Mass Storage Systems and Technologies, San Diego, California, USA, March 1999, pp. 116-127.