A Network Striped Storage System for Video on Demand

Feng Shi
feng.shi@cl.cam.ac.uk
[†]Computer Laboratory
University of Cambridge
Cambridge, CB2 3QG, UK

Andy Hopper[†‡]
ah@cam-orl.co.uk
[‡]Olivetti Research Ltd.
24a Trumpington Street
Cambridge, CB2 1QA, UK

Abstract

This research advocates the architecture of using the switched network as the interconnect among the loosely coupled storage devices for large scale video on demand(VOD) servers. The article proposes a flexible and scalable network striped distributed storage system framework to exploit the above architecture. Quality of Service(QoS) and implementation issues are also discussed in the end.

1 Introduction

Large scale storage servers for video on demand(VOD) are becoming more important. Though a single modern disk can support the playing back and recording of several MPEG-I or MPEG-II streams, the bandwidth requirement of many concurrent streams is well beyond the performance of a server based on one disk or a small scale disk array.

The most important aspect of the architectural part of a large scale video server is the interconnect structure from the basic storage units(the disks) to the customers(the set-top boxes(STB)). Generally, there are four types of interconnect: backplane, channel, parallel computer interconnection and network. Most current high-end video servers use a centralised server structure exploiting the combination of the first three interconnect methods, and network is only used to connect the server to the STBs. Several problems exist for this kind of structure: single server machine bottleneck, non-scalable and non-cost-effective.

Recent researches have made Gigabit switches a reality, while the cost/performance ratio of microprocessors is dropping steadily. All these make it attractive to use the off-the-shelf components inside the VOD server with the scalable switched network as the primary interconnect. This work advocates the architecture of attaching the storage peripherals directly to a network style interconnect(ATM or switched Fibre Channel(FC)) and striping video/audio data to multiple peripherals for VOD applications. The storage peripheral could be a single disk or a cost-effective small scale RAID using internal backplane or non-switched channel such as SCSI or FC-AL(Arbitrated Loop). The rationale behind this network striped storage system architecture is cost-effectiveness, scalability, flexibility and high-performance.

2 Storage System Framework

2.1 System Components

To exploit the above video server architecture, a network striped storage system framework is proposed(Figure 1). The framework is composed of the storage server(SS) which resides in each network attached storage device, the directory server(DS) which presents a hierarchical directory to the clients, the file server(FS) which manages logical objects(LO) such as files and directories, and the stream control factory(SCF) which will create a stream control agent(SCA) for each client's request for each video/audio file's playing back or recording. An LO is striped to several SSs, and the set of stripes of a single LO on the same SS forms a physical object(PO) that is managed by that SS. One of the stripes is the parity stripe as in the RAID scheme. Striping and parity computing are performed by the client. Data are moved directly between the SSs and the client under the control of the SCA. Because of the overhead of the small stripe unit access, the high cost of RAM, and the difficulties of the synchronisations among the storage devices across the network, a RAID5 striping scheme is recognised to be better than a RAID3 one.

This system framework is highly flexible:

1. Different LO can have different striping unit size(SUS) and each FS can stripe any of its LO to any subset of SSs on the network.

2. Directory service is separated from the underlying file and storage services. A three level naming

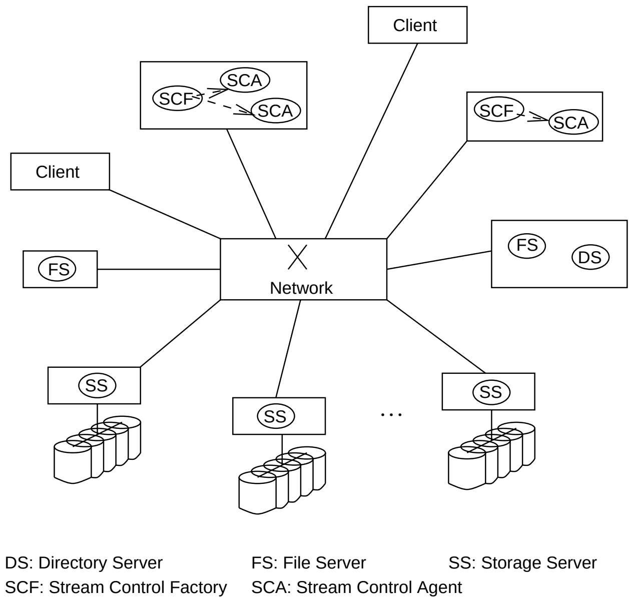

Figure 1: The Network Striped Storage Framework

scheme is supported: textual name, LO identifier(LOID) and PO identifier(POID).

3. All the meta-data are stored on the SSs where the LO data are stored. No additional supporting file system is needed.

4. Each SS could have its own policy to store the POs on the devices it manages.

5. The system can support multiple FSs if necessary. FSs, SCFs and SSs can be added incrementally and independently.

2.2 Meta-data Management

The above claimed system flexibility largely comes from the meta-data management mechanism for the FS and the SS. An LOID uniquely identifies an LO among those managed by all the FSs of the same type, while a POID uniquely identifies a PO among all the SSs. A server instance with epoch number scheme as in the Multi-Service Storage Architecture(MSSA)[1] can be used to produce LOIDs and POIDs. The FS has to resolve the LOID to a list of POIDs, and its meta-data is a table of $< LOID : \{POID\} : SUS : ATTR >$ tuples(i.e. the LOID table), where $\{POID\}$ represents a set of POIDs and ATTR refers to other attributes of the LO. Given a POID, a proper SS can be found. The SS can use the POID to resolve into a list of disk blocks which store the PO. So the SS has the $< POID : \{disk{-}block{-}address(DBA)\} >$ table(i.e. the POID table) as a component of its meta-data(the actual format of this index meta-data is implementation dependent). Besides, the SS has to maintain the free block bitmap for its disk devices.

There are several requirements for the storing of the meta-data. First, it should make the meta-data management as independent as possible. Second, it should make it possible to reconstruct the FS meta-data and the LO data after one SS is lost. Third, it should make the system as flexible as stated above. Finally, multiple level naming resolution should be optimised since both tables can be very large. To address the first three problems, the concepts of the FS handle and the FS area are introduced. The last problem can be solved by using the effective schemes in MSSA.

For each FS, its meta-data(the LOID table) is just a special kind of LO and is striped across a subset of SSs, and an FS identifier(FSID) is used to identify this special LO(Figure 2). An *FS handle* is an $< FSID : \{POID\} : SUS : ATTR >$ tuple, which tells where the FS's meta-data is striped. This tuple is created when an FS is created and is obtained subsequently to get the LOID table when the FS is booted. An *FS area* is a fixed area reserved in each SS where FS handles can be stored and replicated.

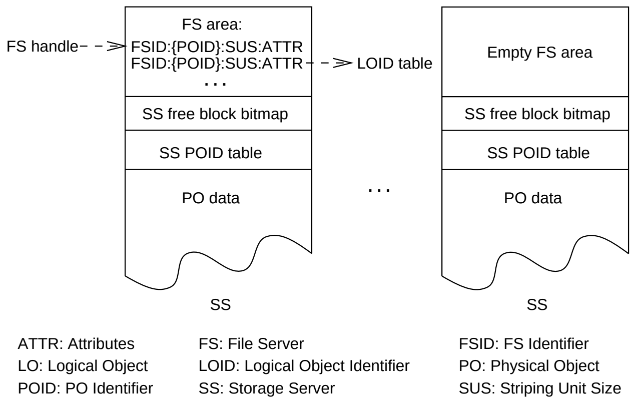

Figure 2: Storage System Meta-Data Management

The SS specific meta-data, i.e. the $< POID : \{DBA\} >$ table and the free block bitmap, are neither striped nor replicated. When an SS is lost, new SS specific meta-data is created during reconstruction of the LO data. This simplifies the management of SS meta-data and achieves the effect of an abstract and encapsulated SS. When a new SS is created to replace a previously lost one, it first replicates the FS area from another SS if necessary. Then it reconstructs the meta-data lost in the SS for the FSs. And finally the SS reconstructs the LO data lost in this SS for the FSs.

2.3 Stream Control

Another system component is the stream control factory(SCF) which can spawn a stream control agent(SCA) to control the transfer of the data for each stream. The SCA could be universal during playing back by associating each variable bit rate(VBR) file with an index file. For recording, index file is created by the client and flow control is maintained mainly by the client and the SSs using peak rate reservation, as the SCA has no idea about the rate of the live source if it is of VBR. This is reasonable since live recording is either rare compared to playing back or initiated by the system operator in a VOD system. However, if the file is of constant bit rate(CBR), no index is needed for either recording or playing back. Data of fixed striping unit size(SUS) is accessed by the SCA for playing back or by the client for recording from or to different SSs in turn, which facilitates the memory management and reduces memory contention in the SSs. This also eases the tension of disk layout schemes provided that one SUS's data for a file is stored continuously or very close by an SS.

The SCF/SCA components relieve the SS of the timing analysis of stream data and effectively provides a rate control mechanism between the SSs and the clients. For example, the SCA can be used to adjust the rate of each stream in order to reduce the I/O demand in the SSs[2]. It can also provide timing information for the SS to achieve initial synchronisations between related streams for playing back, though it is preferred that stringent synchronisation is performed near the sink by a stream/group agent method[3]. In this case, the SCA can be used to adjust subsequent data sending time from the SSs based on the feedback from the stream/group agent for the latter to maintain jitter and skew bound in the sink.

An SCF usually resides on one machine attached to the network. If the load on that machine is high, additional machines and factories can be added to process more client requests. Thus the scaling of the streaming capacity and the scaling of the storage performance are independent.

3 Quality of Service Issues

From the perspective of the storage server, the QoS only represents the ability of the server system to send/receive the data of a stream in time and the degree of the initial synchronisations provided among related streams in playing back. There are many published work on video servers providing deterministic or statistical QoS guarantees. However, most of them are considered limited or impractical in a network striped environment because they usually assume one disk or a synchronised disk array, or build their work on mathematical models or simulations with many further assumptions, or ignore network and other system activities.

The arguments against the above practice are of four folded. First, there is disk device uncertainty because of the uncertain seek and rotation delay, defect management, zoned bit recording(ZBR), thermal recalibration, retrying, and the intelligence of the controllers etc.. Second, even if most of the disk parameters can be extracted[4] and simulated[5], the on-line estimation and application of the service time based on simulation can be computationally prohibitive. Third, there is activity dependency, as disk activities are not the only ones in a storage system. Fourth, there is processing time variability. For VBR streams, the amount of time needed to deal with(disk read/write, CPU process, network transmit/receive) the data that can be played back for the same period will vary. Therefore, it is believed that only when storage devices with deterministic behaviours(e.g. [6]) are used and the activities of the whole system besides that of the disk subsystem is considered can the modellings and simulations be realistic.

Therefore, an important consideration for QoS guarantee which seems always omitted by most current video server research is that the collective activities associated with one stream should not or should only minimally affect the QoS contract of another during temporary overload situations. In the same spirit as Nemesis[7], this work will reserve a particular resource bandwidth for each stream and police each SCA's request to guarantee that share in the SSs, while resources underused by one stream can be picked up by another. This, coupled with optimised disk scheduling, data layout and prefetching policies, will provide a more predictable service to the clients.

4 Implementation Status

Using the ORL direct peripheral infrastructure and ATM interconnect[8], a prototype system based on the above framework has been implemented but without underlying QoS support yet. Video data can be read from several network attached disks and displayed in video tiles which are also attached to the ATM network. SCFs, FSs and DSs normally reside in ATMos boards, which are simple processing units with direct ATM connections.

Different system components interact through CORBA[9] interfaces. The software platform is the locally developed OmniORB which is a multi-threaded implementation of ORB. It is found that distributed object computing fits very nicely with the proposed storage framework, as the system components and the logical/physical objects can be mapped directly to distributed objects, and object references can be passed to a third party easily. Also additional functionalities can be added to one component without any modification to existing services in the same component or in other components. However, the CORBA interfaces are only used for control. Data are transferred between the clients and the SSs directly in a separate channel.

5 Related Work

Three file systems based on network striping are Swift[10], Zebra[11] and HPSS[12]. As they are not oriented to multimedia applications, explicit rate control and synchronisations are not dealt with. Also the meta-data management scheme and the separation of DS/FS/SS make the proposed framework more flexible and simpler than other systems.

More related work are ISS[13], MARS[14] and SPIFFI[15]. However, none of them presents a flexible software system framework for the network striped architecture, which is the focus of this article. In addition, they are more concerned with data placement, disk scheduling and prefetching instead of system level QoS considerations.

6 Conclusions and Future Work

This article advocates the use of storage devices to connect to the network directly for large scale VOD applications and proposes a network striped storage system framework which is flexible, scalable and practical. Future work include QoS enforcement with the timing support from GPS(global positioning system) and NTP(network time protocol) as well as performance measurements.

Acknowledgements

Olivetti Research Ltd. provides the hardware equipments. Feng Shi holds a Cambridge Overseas Trust Scholarship.

References

[1] S. L. Lo. *A Modular and Extensible Network Storage Architecture.* PhD thesis, University of Cambridge, Computer Laboratory, Jan. 1994. Technical Report No. 326.

[2] L. Golubchik, J. C. S. Lui, and R. Muntz. Reducing i/o demand in video-on-demand storage servers. In *SIGMETRICS'95*, Ottawa, Ontario, Canada, 1995. ACM.

[3] C. J. Sreenan. *Synchronization Services for Digital Continuous Media.* PhD thesis, University of Cambridge, Computer Laboratory, Mar. 1993. Technical Report No. 292.

[4] B. L. Worthington, G. R. Ganger, Y. N. Patt, and J. Wilkes. On-line extraction of scsi disk drive parameters. In *SIGMETRICS'95*, Ottawa, Ontario, Canada, 1995. ACM.

[5] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, Mar. 1994.

[6] R. R. Birge. Protein-based computers. *Scientific American*, Mar 1995.

[7] T. Roscoe. *The Structure of a Multi-Service Operating System.* PhD thesis, University of Cambridge, Computer Laboratory, Aug 1995.

[8] A. J. Chaney, I. D. Wilson, and A. Hopper. The design and implementation of a raid-3 multimedia file server. In *NOSSDAV'95*, Durham, NH, US, April 1995.

[9] The common object request broker: Architecture and specification. Revision 1.1 Draft 10 OMG Document Number 91.12.1, Dec 1991.

[10] L. F. Cabrera and D. D. E. Long. Swift: Using distributed disk striping to provide high i/o data rates. *Computing Systems*, 4(4), Fall 1991.

[11] J. H. Hartman and J. K. Ousterhout. The zebra striped network file system. *ACM Tranc. on Computer Systems*, 13(3), Aug 1995.

[12] R. W. Watson and R. A. Coyne. The parallel i/o architecture of the high-performance storage system(hpss). In *14th IEEE Computer Society Mass Systems Symp.*, 1995.

[13] B. Tierney, W. E. Johnston, H. Herzog, G. Hoo, G. Jin, J. Lee, L. T. Chen, and D. Rotem. Distributed parallel data storage systems: A scalable approach to high speed image servers. In *Multimedia'94*, San Francisco, CA, USA, Oct 1994.

[14] M. M. Buddihikot and G. M. Parulkar. Efficient data layout, scheduling and playout coutrol in mars. In *NOSSDAV'95*, Durham, NH, US, April 1995.

[15] C. S. Freedman and D. J. DeWitt. The spiffi scalable video-on-demand system. In *SIGMOD'95*, San Jose, CA, USA, 1995. ACM.