The *INSTANCE* Project: Operating System Enhancements to Support Multimedia Servers

Pål Halvorsen, Thomas Plagemann, and Vera Goebel
University of Oslo, UniK - Center for Technology at Kjeller
P.B. 70, N-2027 KJELLER, Norway
Email: {paalh, plageman, goebel}@unik.no

1 Introduction

Traditional operating systems do not provide adequate support for large scale multimedia-on-demand servers. The objective of the INSTANCE project [10] is to identify and eliminate possible bottlenecks of traditional operating systems used in most server based systems. So far, we have developed with three possible orthogonal system improvements: network level framing, integrated error management, and a zero-copy-one-copy memory architecture.

2 Asynchronous Communication Between Information Provider and Consumer(s)

In our project, we regard server-based systems as asynchronous people-to-people communication and consider the server as an intermediate storage node. As shown in Figure 1, this means that an event, like a soccer game, is recorded at a remote site, transmitted to a multimedia storage server, and stored on persistent storage (disks). A remote client can retrieve and play out the information whenever he wants, i.e., like in a Video-on-Demand or News-on-Demand scenario. Thus, the challenge is to design a system that supports a large amount of concurrent users retrieving data from the server, e.g., requesting one of the top ten videos from Blockbuster or the latest news from CNN. The bottleneck is the data retrieval and transmission of data from disk to network. In this context, the most important performance factors are disk I/O, communication protocol processing, error management, and memory management. In this extended abstract, we present our proposed approach to improve the design of such a multimedia storage server to get a high performance, low overhead system supporting a huge amount of concurrent users by a single storage node.

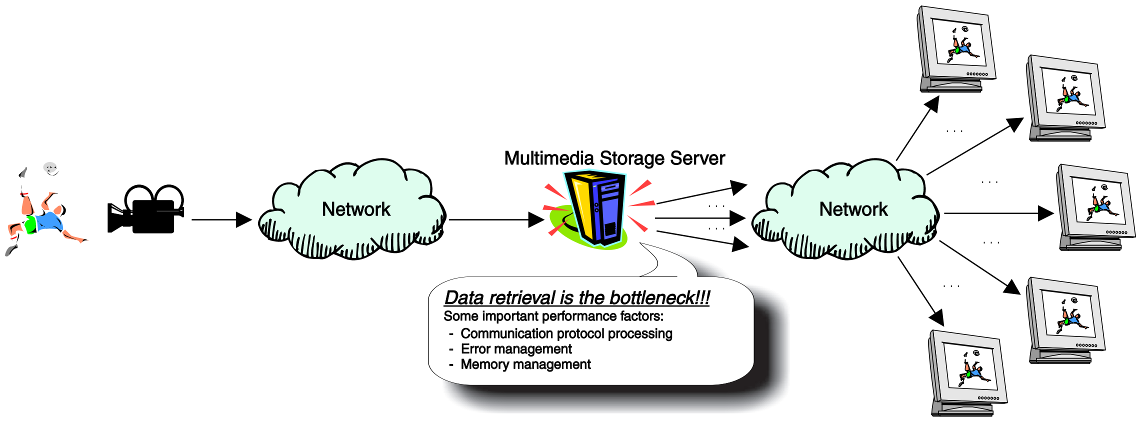

Network
Multimedia Storage Server
Network
Data retrieval is the bottleneck!!!
Some important performance factors:
- Communication protocol processing
- Error management
- Memory management

Figure 1: Application scenario.

3 Network Level Framing

End-systems have to handle in the OSI reference model all seven protocol layers and in the Internet protocol suite all four protocol layers, while intermediate systems - or nodes - have only to handle the three lowest layers. The bottleneck in communication is located in transport and higher layer protocols where for example checksum calculation is among the most time expensive operations.

As shown in Figure 2, current server based systems see an end-to-end relationship between server and client. Consequently, servers have to handle all the end-to-end protocols when satisfying a request from a client. Data management system and application are placed on top of the end-to-end protocols. Thus, for each client the same data must be processed through all the end-to-end protocols performing the same CPU intensive operations.

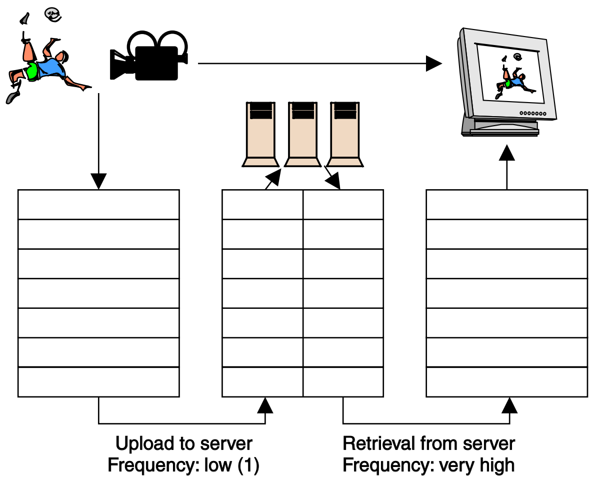


Figure 2: Traditional data storage in a server.

In contrast to the traditional approach, INSTANCE regards a server as an intermediate storage node as shown in Figure 3. Thus, servers might have to handle in heterogeneous networks only the lowest three layers and in homogeneous networks only the lowest two layers. This means that when the data is processed through the "intermediate system layers", the data is directly stored on disk including the higher layer packet headers, i.e., the output of time expensive operations like checksum calculation is stored on disk and can be omitted when the data is later sent to remote clients. The advantage of this approach is that the CPU intensive end-to-end protocol handling is not necessary - respectively reduced to a minimum - at the intermediate storage node. This results in a reduced load of CPU and system bus, i.e., in other words, more consumers (clients) can be served with the same hardware resources.

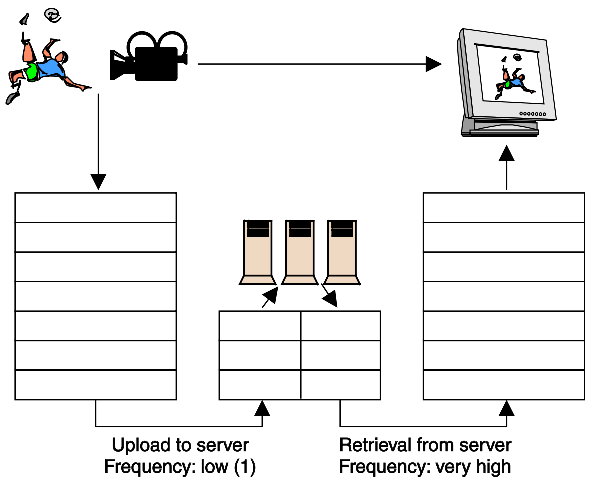


Figure 3: Network level framing.

4 Integrated Error Correction

When transferring data from a remote server in a distributed environment, several components along the data path may fail and introduce errors. To offer an "error-free" service, error detection and correcting schemes are used in several subsystems, i.e., in the storage system and the communication system. Thus, the correctness of the information is checked multiple times, and the functionality is redundant. In INSTANCE, we look at the possibility of integrating the error management in a disk array (RAID) [3, 9] and a forward error correction scheme in the communication system [6], i.e., since we are looking at a multicast scheme (see Section 5) for multimedia data, a forward error correction scheme may be appropriate to avoid retransmissions of data which then will arrive to late to multiple failing clients.

Traditionally, as depicted in Figure 4, the storage system and the communication system use different error management mechanisms. A RAID system generates redundant parity information to be able to recover from a disk crash and restore the data, and when the data is sent to the communication system, a new error mechanism is applied where a new set of redundant parity information is computed.

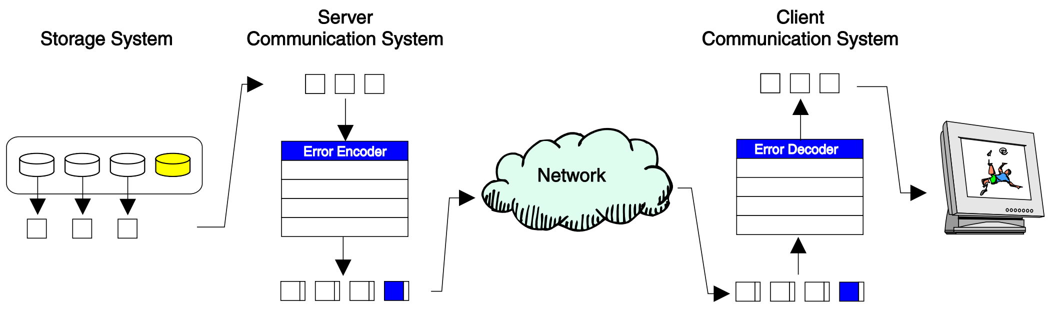


Figure 4: Traditional error management.

Figure 5 shows our approach to integrate the different error recovery mechanism in the storage system and the communication system. Opposed to the traditional data read from a RAID system where the parity information is only read when a disk error occurs, we would also like to read the redundant error recovery data and transmit this data to the remote client for use as forward error correction scheme in the communication system. Thus, instead of reading only the original data, we also read the parity data. All the data retrieved from the storage system is sent to the communication system where the error encoder, which performs CPU expensive operations, now can be committed and removed. Finally, the data is sent to the client where the communication system has a similar error decoder for forward error correction as in the storage system on the server side.

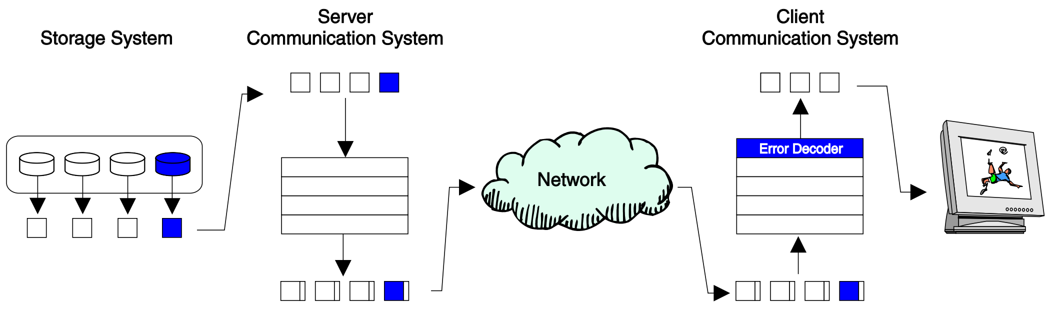
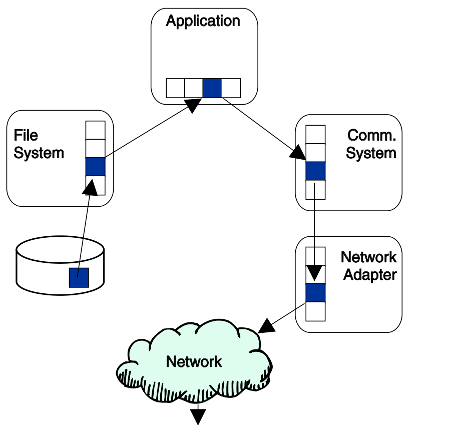


Figure 5: Integrated error management.

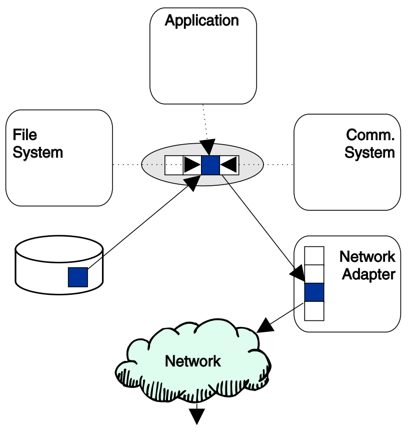
5 Zero-Copy-One-Copy Memory Architecture

The system performance is strongly influenced by the memory architecture where the high cost of copy operations have been addressed for a long time as one of the major bottlenecks. To be able to support multimedia playout to a large amount of users, we need a memory architecture which minimizes the number of in-memory copy operations and the number of identical data copies within the memory.

In the traditional disk-to-network data path, data is copied between the memory locations of all different subsystems (see Figure 6A). Therefore, it is not suitable for high performance systems like a multimedia storage server. This approach results in several expensive cross domain copy operations. To reduce the overhead of copy operations, several zero-copy memory architectures (for example IO-Lite [8], mmbufs [2], the UVM virtual memory system [4], the Genie I/O system [1], etc. - see [11] for more references), as shown in Figure 6B, have been proposed to enable data transfers between disk-to-network without physically copying data between the subsystems.



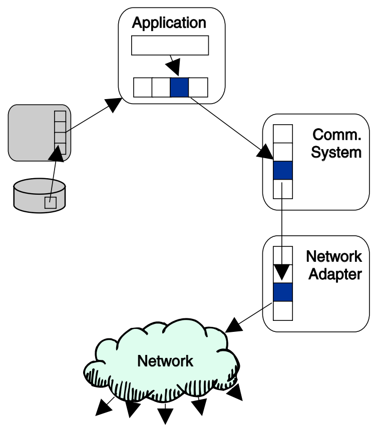
(A) Datapath in a traditional memory architecture



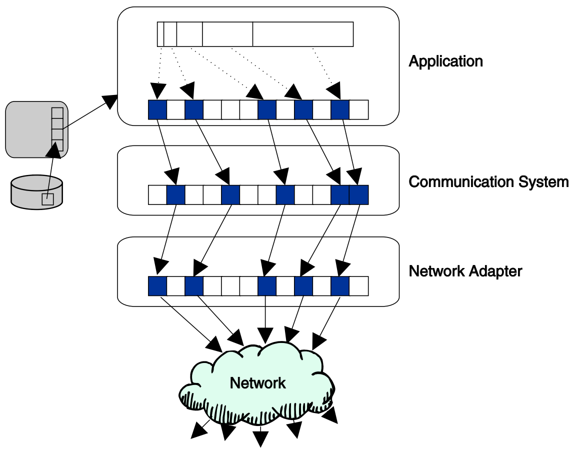
(B) Zero-copy memory architecture

Figure 6: Eliminating physical copy operations.

A "problem" with the current zero-copy architectures is that each client gets an own connection each requiring an own set of resources. This may result in multiple copies in memory and a lot of processing overhead sending the same data to a large amount of users through the communication system. This can be solved by a broadcast/multicast (hereafter broadcast) transmission (Figure 7A) where the interested clients join a broadcast reducing the server workload to send one copy of the data with a broadcast address. The drawback of a traditional broadcast is that all the clients have to start viewing (retrieving) the same data at the same time. Thus, a client arriving late has to wait until a new broadcast transmission begins, or he has to join the current transmission where he loses the already transmitted data. Figure 7B shows a delay minimized broadcasting scheme (á la pyramid broadcasting [12], skyscraper broadcasting [7], or greedy disk conserving broadcasting [5]) where each video is partitioned in growing partitions. Each partition is then broadcasted in rounds, i.e., when it is finished it just restarts at the beginning of the partition. The basic idea behind the partitioning scheme is that the download of the next partition does not take longer than the playout of the former. Thus, making the first partition small reduces the start delay to the time it takes to transmit the first partition.



(A) Traditional broadcast



(B) Delay minimized broadcast

Figure 7: Reducing resource requirements and response time.

As we can see, there has been done a lot of work in the area of eliminating in-memory copy operations and in the area of serving multiple clients with a minimized initial delay. However, current zero-copy architectures may still have a copy per client in memory, and each client's data is processed through the communication system. Furthermore, the delay minimized broadcasting schemes copy data between subsystems. Thus, in INSTANCE, we will integrate a zero-copy memory architecture with the delay minimized broadcasting scheme to remove all in-memory copy operations, remove per client copies, reduce processing overhead, and still minimize the response time (Figure 8). We have called our architecture “Zero-Copy-One-Copy” referring to a zero-copy data path and one copy of the data shared among all users.

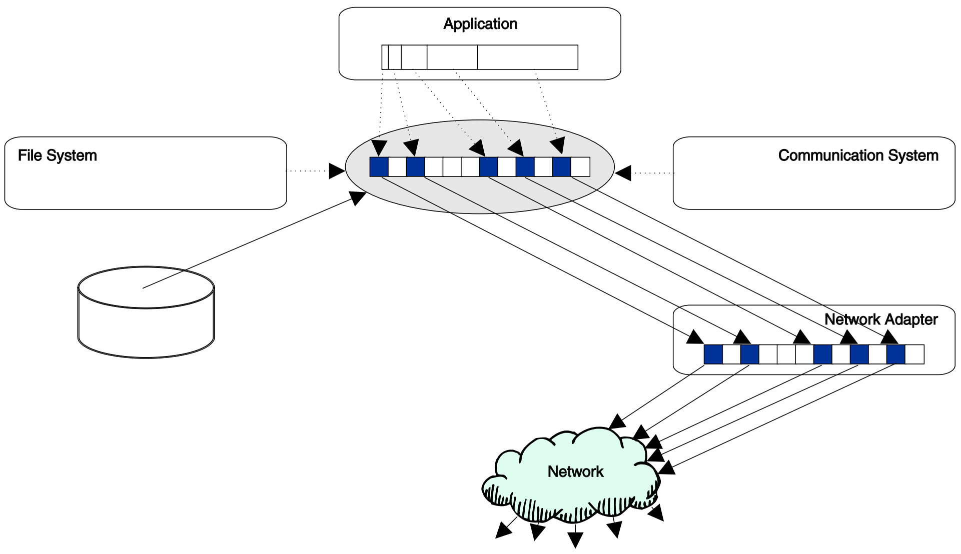


Figure 8: Integrated zero-copy and minimized delay broadcasting.

6 Status and Future Work

In our current status, we are about to start implementing a test system for the network level framing and evaluate the use and performance of such a mechanism. The integrated error management mechanism is under development where we evaluate the use of different error recovery schemes (and combinations of such schemes) in our integrated mechanism. The zero-copy-one-copy memory architecture are also under development where we have the zero-copy memory architecture called mmbufs [2] and the UVM virtual memory system [4] supporting a zero-copy data-path, both from Washington University, St. Louis, to build the delay minimized broadcasting scheme on. We are currently working on NetBSD (in which both mmbuf and UVM are implemented), but operating systems such as the library operating system Nemesis will also be evaluated.

Future work include implementation and evaluation of these mechanisms, evaluation of other existing mechanisms which may be integrated including existing error mechanisms and zero-copy mechanisms, and the integration of all these concepts into the operating system for support of a multimedia storage server.

References

[1] Brustoloni, J. C.: *“Interoperation of Copy Avoidance in Network and File I/O”*, Proceedings of the 18th IEEE Conference on Computer Communications (INFOCOM’99), New York, NY, USA, March 1999

[2] Buddhikot, M. M: *“Project MARS: Scalable, High Performance, Web Based Multimedia-on-Demand (MOD) Services and Servers”*, PhD Thesis, Sever Institute of Technology, Department of Computer Science, Washington University, St. Louis, MO, USA, August 1998

[3] Chen, P. M., Lee, E. K., Gibson, G. A., Katz, R. H., Patterson, D. A.: *“RAID: High-Performance, Reliable Secondary Storage”*, ACM Computing Surveys, Vol. 26., No. 2, June 1994, pp. 145 - 185

[4] Cranor, C. D.: *“The Design and Implementation of the UVM Virtual Memory System”*, PhD Thesis, Sever Institute of Technology, Department of Computer Science, Washington University, St. Louis, MO, USA, August 1998

[5] Gao, L., Kurose, J., Towsley, D.: *“Efficient Schemes for Broadcasting Popular Videos”*, Proceedings of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV’98), Cambridge, UK, 1998

[6] Halsall, F: *“Data Communications, Computer Networks and Open Systems”*, Fourth edition, Addison-Wesley, 1995

[7] Hua, K. A., Sheu, S.: *“Skyscraper Broadcasting: A New Broadcasting Scheme for Meteropolitan Video-on-Demand System”*, Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM’97), Cannes, France, September 1997, pp. 89-100

[8] Pai, V. S., Druschel, P., Zwaenepoel, W.: *“IO-Lite: A Unified I/O Buffering and Caching System”*, Proceedings of the 3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI’99), New Orleans, LA, USA, February 1999, pp. 15 - 28

[9] Patterson, D. A., Gibson, G., Katz, R. H.: *“A Case for Redundant Arrays of Inexpensive Disks (RAID)”*, Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, IL, USA, June 1988, pp. 109 - 116

[10] Plagemann, T., Goebel, V.: *“INSTANCE: The Intermediate Storage Node Concept”*, Proceedings of the 3rd Asian Computing Science Conference (ASIAN’97), Kathmandu, Nepal, December 1997, pp. 151-165

[11] Plagemann, T., Goebel, V., Halvorsen, P., Anshus, O.: *"Operating System Support for Multimedia Systems"*, to be published in The Computer Communications Journal, Elsevier, Fall 1999 or Spring 2000

[12] Viswanathan, S., Imielinski, T.: *“Metropolitan area Video-on-Demand Service Using Pyramid Broadcasting”*, Multimedia Systems, Vol 4., No. 4, 1996, pp. 197-208