3. Storage and Retrieval of Video

We have some recently-donated storage devices, including disk, digital linear tape and solid state disk. This project would involve storing video on the different storage devices and experimenting with retrieval of the data. The purpose of this project is to evaluate the performance of a storage server that has a variety of storage devices attached to it. Video objects may migrate between different levels of the storage hierarchy, based on their changing popularity or based on predictions, either that the video will soon be accessed or is unlikely to be accessed again. This project would include measuring the write and read bandwidth and latency of the storage devices. It would also include simulating a variety of workloads that might be applied to a server (for example, much has been predicted about a large video-on-demand system). Based on popularity of objects in the workload, we can assign particular objects to particular levels of the storage hierarchy. Over time, as popularity of the objects changes, we may migrate objects between levels of the hierarchy. This project would include writing software under Solaris to manage the storage hierarchy; this storage manager should be object-oriented and should be flexible enough to allow experimentation with a wide range of migration strategies.

4. Experiments with the new video server

Recently, our brand new, $160,000 video server has arrived. You can be the first to use it! Experiment with the capabilities of this server, which includes a SPARC 1000, six fast-wide SCSI buses, a disk array, and special disk layout and scheduling software to support up to 100 simultaneous MPEG-2 or 270 simultaneous MPEG-1 streams. This project would include storing video on the server, measuring its performance to the finest grain possible (e.g., get into the guts of Solaris and the video server software, and try to understand what is going on), and evaluate the quality of video delivered to clients over the fast Ethernet.

5. Real-time Scheduling

This project analyzes how various multimedia specific storage techniques (e.g., Rangan, Chang, Tobagi) can be combined with thread and communication scheduling to minimize resource requirements (memory, processors) while maximizing such performance parameters as number of streams and quality of service.

The analysis may be based on a complete system simulation, or only the I/O activity may be simulated while the actual server may be implemented with a user threads package such as Cthreads on Solaris. In the latter case, a user level scheduler should be written.

Project Suggestions
CS8113D Spring 1996

Overall, each project should contribute to the infrastructure that we are building that will include a video server, a hierarchical storage system, an ATM network that feeds into a hybrid fiber-coax network.

1. General experimentation with compression software

Scott Register in OIT has access to several hardware and many software compression systems. Working with Scott and using the resources in CoC and OIT, experiment with the steps of video compression: capture, compression, storage, delivery and decoding. Evaluate the different schemes based on quality metrics, compression achieved, frames per second delivered, etc. Try to look at video clips that include high and low amounts of motion.

2. Multithread a commercial software compression scheme under Solaris

Software MPEG encoding is notoriously slow. Scott Register has approached a commercial compression company about speeding up this process by multithreading their encoder. They are agreeable to working with graduate students, as long as non-disclosure agreements are signed. Here is part of their email discussion on the project:

> Earlier, I spoke w/ someone about working together > to port your compression software to Solaris multithread and/or > Sun UltraSparc (SIMD). Have you already done this? If not, are you > still interested and pursuing those ports, possibly together? > Please let me know your thoughts.

First let me ask what level of effort you can apply. There are three levels of work that could be done to take advantage of multi-threading: (1) simply port the multi-processor motion estimation code (currently running on sgi). This code works but quickly runs into Amdahl's law. It operates on a single picture in a single search direction. (2) change the core algorithm to "dispatch a set of pictures" and do motion estimation on them. This would improve on the current approach by allowing parallelism to improve a larger job. (3) Rewrite the core algorithm so that each GOP is compressed by a separate thread. This would require applying certain constraints on the end of a GOP (for example, the last B pictures could not refer forward to the next I frame; the constraints on buffer occupancy would need to be more stringent at the end of each GOP).

> I'm sure that, working alone, we could get the motion-estimation code > ported to Solaris threads, although I'm sure we would need to learn some > more about MPEG first. However, I would really like to be able to do more.

If you only get the Solaris thread version running, you need not understand anything about MPEG. The current implementation is very localized and the only issue is a thread implementation for solaris. If you did (2) or (3), then you would need to understand MPEG.