The following paper was originally published in the
Proceedings of the USENIX Windows NT Workshop
Seattle, Washington, August 1997

IntelliJuke - a Caching Jukebox-Based Storage Server

Yitzhak Birk, Uri Kareev and Mark Mokryn
Technion - Israel Institute of Technology
Haifa 32000, Israel

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

IntelliJuke - a Caching Jukebox-Based Storage Server

Yitzhak Birk, Uri Kareev and Mark Mokryn
Technion - Israel Institute of Technology
Haifa 32000, Israel
birk@ee, kareev,mark@psl.technion.ac.il; www.psl.technion.ac.il

Background

The cost-effectiveness of CDs applies only to the storage medium itself; once a drive is included, CD-ROM often becomes inferior to magnetic hard drives. Moreover, a CD is often far from full, thus further reducing its cost-effectiveness. (For this reason, DVD will not help in many cases.)

Jukeboxes (AKA changers) feature high volumetric storage density with moderate communication bandwidth and multi-second access times. Jukebox manufacturers have focused on building very robust systems with high-performance robotics in order to mitigate the intrinsic performance shortcomings, resulting in a price-per-slot of $30-$120 in late 1996!

We believe that jukeboxes should provide access to a very large number of CDs at minimal cost, and that caching on magnetic disk drives should provide the performance. A byproduct of caching is reduced jukebox activity, so the jukebox's robustness may be reduced without increasing the mean time between failures. Our focus is on efficient caching.

Caching approach

We have adopted a hybrid caching approach: speculative fetching based on "intelligence", typically involving entire files and even related files, with removal based on "de facto" information and applied at a much finer, intra-file granularity. Our rationale is as follows: the miss penalty is very high; the cache (disk) is relatively inexpensive and can be large, possibly permitting an unaccessed item to reside in the cache for several days before it must be removed. If, however, in the course of several days in the cache, certain blocks in a given file were accessed while others were not, there is reason to believe that the unaccessed blocks will not be accessed in the near future and can be discarded. It should be noted that this observation hinges not merely on the relative access times of the two storage layers; rather, it takes into account the time constants of a human user!

Since it is easy to construct "good" and "bad" scenarios for such an approach, we decided to build a prototype so as to permit experimentation and measurements in a real environment.

IntelliJuke overview

IntelliJuke is a network-accessible hierarchical storage server comprising a PC running Windows NT, connected to a Kubik CDR-240 juke box with 240 slots and two 12X Plextor CD-ROM drives. Data is presented to users as a collection of NTFS directory trees on a hard drive, one tree per CD (single drive letter for all). The initial goal is to enable users to seamlessly access any data that would be accessible if the entire CD were copied onto a hard drive using file manager. Our focus has been on providing support for novel caching policies while using the native NT services whenever possible. The project is presently in advanced implementation stages at the Parallel Systems Lab of the EE dept., with most of the difficult problems out of the way. The caching algorithms will be tuned once the system is fully operational.

NT- related implementation challenges

- The hybrid caching approach and requirement for network access mandated the insertion of a "highest level" driver above an NTFS partition.
- Construction of a communication mechanism between our kernel driver and user-mode code. One complication is that communication is initiated by the driver.
- Overcoming the locking of files by the operating system. This occurs, for example, when the OS decides to issue a read-ahead request while we are handling an earlier miss to the same file, and prevents us from writing the requested data into the cache.
- The desire to manage the cache at block granularity and to free up disk space while using NTFS forced us to support "sparse" NTFS files.

Acknowledgments and credits. Parts of IntelliJuke were implemented by Amnon I. Govrin, Ran Herzberg, Eran Rosenberg and Eyal Zangi. Tomer Kol and Evgeny Rivkin have provided ongoing assistance. The project has been supported in part by Microsoft through product donations and by a grant from EMC (Israel) Storage Systems Ltd.