

“Serverless Network File Systems”

Based on the Thomas E. Anderson, et. al. paper
“Serverless Network File Systems”, Computer
Science Division, University of California at
Berkeley

Overview of Serverless Network File Systems

- Rather than one central server, peer workstations cooperate to provide all aspects of network file system services. “Anything, anywhere.”
- Prototype called xFS- though it has not yet (as of ‘95) implemented all aspects proposed in the paper. Some performance measurements are based on simulation.
- Provides fault tolerance and high availability in the event of network failures and machine crashes.
- Performance benchmarks suggest that xFS scales almost linearly with increasing numbers of workstations.
 - Draws mainly from fast switching networks like ATM.
- Based on several recent and ongoing research efforts including RAID, LSF, Zebra, and multiprocessor cache consistency models.

Background Inspiration for xFS

- RAID (Redundant Array of Inexpensive Disks)
 - Given N disks, partitions a stripe of data into N-1 data blocks and 1 parity block.
 - Parity block is the exclusive or of bits in other N-1 blocks.
 - Provides higher bandwidth & fault tolerance.
- LSF (Log-structured File System)
 - e.g. Sprite & BSD LFS prototypes.
 - Individual writes are buffered into memory and committed to the end of an append-only log file.
 - As blocks are overwritten and appended to the log, the previous entry in the log is deleted and inodes are updated to reflect the new address of the data block.
- Zebra
 - A network file system that uses a software RAID and LSF file systems.
 - Uses a mechanism known as a “delta” to assure atomic commitment of actions taken on multiple machines and allow the system to replay modifications during failure recovery.

xFS File System Entities

- Four basic entities cooperate to perform the tasks of the serverless network file system.
 - Clients: Similar to the clients of any other network file system, but maintain a more active role, especially concerned with cooperative cache consistency.
 - Storage Servers: Store data in a software RAID configuration in striping groups.
 - Managers: Maintain the cache consistency state and disk location metadata. Each file is mapped to a unique manager at any one time.
 - Cleaners: Defragments holes in log file segments when data is overwritten making room for new log files.
- Each machine may contain all four entities or a subset.
- Managers and cleaners require a client to be running on the machine. Storage servers may be independent.

Metadata & Data File Mappings

- xFS uses four types of file mappings to locate data on the storage servers.
 - File Directories: Provide a mapping of a file's name to its index number.
 - Manager Map: Locates the manager of a certain file based on its index number. They are maintained by managers and replicated to all clients and other managers. Maps allow for course-grained load balancing among managers.
 - Imap: Maps a file's index to the log address of that file's inode. The inode maintains a list of pointers to the data blocks of that file.
 - Stripe Group Map: Maps a file's log address to a list of storage servers that contain the file's data segments.

Disk Reads

- Client first checks local cache for data block, or requests the block from the manager when not in cache.
- If manager (maintains cache consistency state of clients) knows of some other client that has the data cached, the request is forwarded to that client, data is sent directly to the requesting client, and the manager updates its cache consistency state.
- When data is not in any client's cache it must be read from disk by the manager which checks its imap to find the log address which itself may or may not be cached and have to be read from disk.
- Once the address is found the stripe group map may be used to locate the storage servers which hold the requested information and send them directly to the client.

Disk Writes

- Since small writes are inefficient, clients buffer them in memory until they can periodically be committed to the file log.
- For each explicit write to the disk log file, the disk address associated with the particular block being written is updated, leaving a hole where that block was last written to in the log file. (Recall that cleaner entities defragment these holes.)
- Managers are notified as well to update inodes and imaps.
- Like Zebra, xFS uses the delta concept to make both log file updates and notification of the manager an atomic operation maintaining consistency even in the event of a machine crash.

Cache Consistency

- Token-based cache consistency, a per-block granularity basis.
- Managers maintain cache consistency state.
- Clients must acquire write ownership of a block in order to write to it.
 - Write permission request sent to manager.
 - Manager invalidates caches of all clients caching that block.
 - Manager updates cache consistency information.
- Once acquired, client may write repeatedly to the block without requesting permission again until that block is read from or written to by another client. At that point ownership is revoked, writes are flushed from cache to disk, and block is forwarded to new client.