SLASH2 – Filesystem for Widely Distributed Systems

Pittsburgh Supercomputing Center, Advanced Systems Group

SLASH2 is a highly portable, user mode filesystem which has been created to aid users who frequently deal with large datasets in widely distributed environments. SLASH2 has been designed from the ground up to meet a technological need in the space of wide-area data management. SLASH2 has been built as a portable filesystem which may be layered atop existing storage system deployments so that users may more easily leverage various elements of organically changing storage topologies. This project recognizes that such a 'storage topology' may span a multitude of heterogeneous storage systems and possibly several institutions. The SLASH2 software stack has been positioned so that it may link independent storage systems in a relatively unobtrusive fashion. By binding otherwise independent storage systems with the SLASH2 object-based protocol, SLASH2 aims to provide users with fine grained control of data locality while presenting a single name space.

Experience at the Pittsburgh Supercomputing Center has shown that many data oriented groups spend large amounts of time managing data transfers between various HPC organizations or from instrument based system. Numerous types of work flows require researchers to explicitly control the locality of their data at specific points through the progression. Unfortunately, in cases where the input or output datasets are large, the tasks associated with wide-area data exchange become cumbersome and time consuming. In general researchers are forced to deal with retrying failed transfers, juggling simultaneous transfers, learning about the I/O and network stratum at the source and destination sites, and verifying data integrity along the way. The goal of the SLASH2 file system is incorporate features relevant to wide-area data management so that the complexity and tedium of large data transfer and name space management may be effectively handled by the filesystem at the behest of the user.

The SLASH2 filesystem incorporates several features designed to overcome limitations frequently found in wide-area storage environments.

Multi-Residency

SLASH2 does not force users into volume wide replication schemes but rather, allows them to specify replication or residency properties on an entire file or specific file chunks. The SLASH2 replication engine allows for a rich set of data residency properties, fine grained control, and data movement which is managed through system-level automation. The SLASH2 metadata structures are able to describe complex residency states, not just on behalf of an entire file, but also on individual file chunks. Note that the replication engine supports recursive operations and two policy types: one-time and persistent. A 'one-time' policy denotes that chunk replicas invalidated by overwrites or truncates will not be automatically reinstated by the system. Conversely, the 'persist' policy will automatically rectify invalidated chunks once I/O to that region of the file has quiesced. Below are example commands which illustrate usage of the replication system:

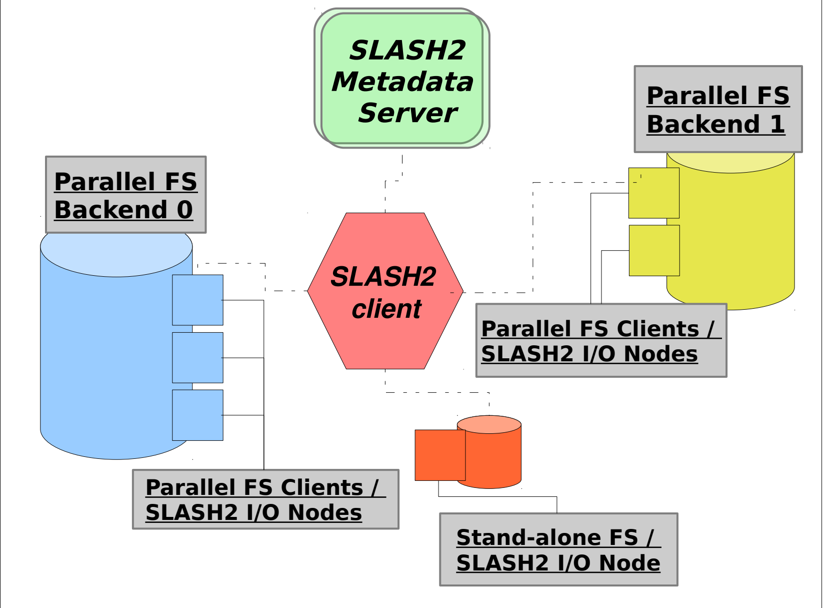
SLASH2 Metadata Server
Parallel FS Backend 1
Parallel FS Backend 0
SLASH2 client
Parallel FS Clients / SLASH2 I/O Nodes
Parallel FS Clients / SLASH2 I/O Nodes
Stand-alone FS / SLASH2 I/O Node

Figure 1: Example SLASH2 Storage Architecture

- Replicate an entire file to the I/O System bessemer@PSC.

msctl -Q bessemer@PSC::/s2/pauln/testfile0*

- Recursively select all files in the directory testdir requesting that the first chunk be replicated to bessemer@PSC and archiver@WVU.
  msctl -RQ bessemer@PSC,archiver@WVU:0:/s2/pauln/testdir
- Apply replication policy to an entire file (affecting only new chunks).
  msctl -f new-repl-policy=one-time:/s2/pauln/testfile0
- Apply a replicated policy recursively to the first bmap of each file.
  msctl -Rf bmap-repl-policy=persist:0:/s2/pauln/testdir

To provide the necessary level of robustness, the replication engine was built such that the MDS could resume uncompleted operations on reboot or restart. Accomplishing this requires all modified chunk residency states to be journaled and flushed to the backing file system. Then a persistent file reference to the target SLASH2 metadata object is stored on disk in a known area of the file system. At system startup, these file references are scanned for files whose replication criteria have not been fulfilled. For each unfulfilled file or chunk located in the system, a work item is established and scheduled for replication activities. In addition to robustness, this methodology provides means of scalability by enabling the SLASH2 MDS to queue a massive number of potential replications requests without being limited by available system memory. Performance of SLASH2 replication has been measured at > 100MB/s for large files sent between I/O nodes connected with gigabit ethernet.

Coherency Model

For data, SLASH2 manages coherency at the chunk level. For multi-writers, or mixed readers and writers, the client will perform direct I/O to the I/O server bypassing the local cache. A callback mechanism manages the process by instructing the respective client or clients to flush or drop any cached pages in preparation for shared mode. Metadata coherency is managed in a more relaxed fashion to allow clients to cache attributes for some number of seconds and to minimize the amount of state managed by the metadata server.

Inline Data Verification

The SLASH2 I/O subsystem produces checksums for each block of modified data and exports them, along with the objects' size, to the metadata server. On read or replication, the checksum tables are retrieved from the metadata server and used by the I/O to verify the data prior to returning to the client. Should faulty data be detected, clients may redirect their request to another replica should one exist. It should be noted that these checksums are only valid for detecting bit rot since because in certain double fault scenarios the checksums could be compromised. This limitation is a side effect of the system's portability – specifically that it does not mandate a specific backing file system for the I/O servers.

Other Features of Interest

- *Stateless Design* - SLASH2 client and I/O services are totally stateless, neither require any sort of additional hardware for journal.
- *Fully Usermode* - To prevent administrative headaches, no SLASH2 subsystems run in the kernel. Filesystem services may be halted without requiring a reboot of the node.
- *Posix I/O Support* - The SLASH2 client utilizes the FUSE (Filesystem in Userspace) module for POSIX I/O support.
- *File size maintained on the metadata server and Readdir+ Implementation* - Together these features greatly improve the performance of attribute acquisition and directory operations by eliminating costly RPCs.
- *Lease Timeout* - All leases in the system have strict timeouts to prevent unresponsive clients and I/O servers from indefinitely holding file system resources.
- *Parallel Data Replication* – Where available, the SLASH2 replication engine can utilize multiple endpoints for replication activities in a load balanced fashion to avoid over scheduling slow endpoints.