Rapid Replication of Multi-Petabyte File Systems

Justin Sybrandt
Grove City College, NERSC
SybrandtJG1@gcc.edu

Jason Hick
NERSC
JHick@lbl.gov

ABSTRACT

As file systems grow larger, tools which were once industry standard become unsustainable at scale. Today, large data sets containing hundreds of millions of files often take longer to traverse than to copy. The time needed to replicate a file system has grown from hours to weeks, an unrealistic wait for a backup. Distsync is our new utility that can quickly update an out-of-date file system replica. By utilizing General Parallel File System (GPFS) policy scans, distsync finds changed files without navigating between directories. It can then parallelize work across multiple nodes, maximizing the performance of a GPFS. The National Energy Research Scientific Computing Center (NERSC) is currently using distsync to replicate file systems of over 100 million inodes and over four petabytes.

1. MOTIVATION

NERSC is moving to a new facility, and will need to transfer its data to the new location. A novel method is needed to quickly ensure that the new file systems contain accurate and up-to-date information.

Data centers have a similar problem when performing routine backups or restoring data from a backup. In all of these cases there are two sets of files, a fresh copy and a stale copy, wherein the stale copy needs to be freshened as quickly as possible.

2. RELATED WORK

Two linux default tools are often used to perform file replications, *cp* [1] and *rsync* [2]. Additionally, data centers have begun to use *mcp* [3], which can move files in parallel, and *shift* [4], which can leverage whole clusters. By using more system resources, these tools can duplicate individual large files much faster than their single-core counterparts.

Unfortunately, none of these tools were designed to synchronize two large file systems containing similar files. NERSC previously used *shift* along with *rsync* to perform multi-petabyte backups; a process which often spanned days as *shift* and *rsync* each processed every file in both the fresh and stale copies.

3. ARCHITECTURE

Our new design creates and compares two lists of file attributes, one for the fresh and stale file system. The differences between these scans are cataloged in various job files, each representing different types of changes between the fresh and stale dates. These jobs are then distributed to a set of machines so that changes from the fresh system can be propagated in parallel. These tasks take place in three modules, the Job Generator, the Manager, and the Worker.

The Job Generator takes advantage of the GPFS [5] policy scan feature to quickly query the file system's metadata. This produces two sorted, plain-text lists of file attributes, one each for the fresh and stale systems. Then, the job generator compares those lists to determine the changes that have occurred since the stale copy was made. The paths of changed files are recorded in job files; each has a type representing the change that occurred in each of its files.

The Distsync Manager uses job files to create a schedule, ensuring that files are never added before their parent directories are created, and that directories are never removed before their containing files are removed. The Manager also spawns the Distsync Worker on a set of provided machines. Lastly, the Manager begins communicating with the Workers, assigning jobs and recording each machine's status.

The Distsync Worker, after receiving a job file from the manager, uses system commands to perform each task. For example, if receiving a job containing modified files, a worker will launch multiple parallel instances of *rsync* in order to propagate any changes from the fresh to the stale file system.

4. PRELIMINARY RESULTS

Preliminary results show that Distsync outperforms the hand-scripted method, and scales in regard to the number of changes, instead of the total file system size. This means that the more frequently a system is backed up, the less time each backup takes. For example, in one test the initial replication of a 350 TB system took over forty hours to complete. A backup three days later only took three hours.

5. REFERENCES

[1] "cp(1) linux user's manual."

[2] "rsync(1) linux user's manual."

[3] P. Z. Kolano and R. B. Ciotti, "High performance multi-node file copies and checksums for clustered file systems," in *Proc. of the 24th USENIX Large Installation System Administration Conf.*

[4] P. Z. Kolano, "High performance reliable file transfers using automatic many-to-many parallelization," in *5th Wkshp. on Resiliency in High Performance Computing.*

[5] F. Schmuck and R. Haskin, "Gpfs: A shared-disk file system for large computing clusters," in *FAST '02 Proceedings of the 1st USENIX Conference on File and Storage Technologies* (editor, ed.), no. 19, USENIX Association Berkeley, CA, USA.