Ceph Distributed File System

Ceph is a distributed network file system designed to provide good performance, reliability, and scalability.

Basic features include:

- POSIX semantics
- Seamless scaling from 1 to many thousands of nodes
- High availability and reliability. No single point of failure.
- N-way replication of data across storage nodes
- Fast recovery from node failures
- Automatic rebalancing of data on node addition/removal
- Easy deployment: most FS components are userspace daemons

Also,

- Flexible snapshots (on any directory)
- Recursive accounting (nested files, directories, bytes)

In contrast to cluster filesystems like GFS, OCFS2, and GPFS that rely on symmetric access by all clients to shared block devices, Ceph separates data and metadata management into independent server clusters, similar to Lustre. Unlike Lustre, however, metadata and storage nodes run entirely as user space daemons. File data is striped across storage nodes in large chunks to distribute workload and facilitate high throughputs. When storage nodes fail, data is re-replicated in a distributed fashion by the storage nodes themselves (with some minimal coordination from a cluster monitor), making the system extremely efficient and scalable.

Metadata servers effectively form a large, consistent, distributed in-memory cache above the file namespace that is extremely scalable, dynamically redistributes metadata in response to workload changes, and can tolerate arbitrary (well, non-Byzantine) node failures. The metadata server takes a somewhat unconventional approach to metadata storage to significantly improve performance for common workloads. In particular, inodes with only a single link are embedded in directories, allowing entire directories of dentries and inodes to be loaded into its cache with a single I/O operation. The contents of extremely large directories can be fragmented and managed by independent metadata servers, allowing scalable concurrent access.

The system offers automatic data rebalancing/migration when scaling from a small cluster of just a few nodes to many hundreds, without requiring an administrator carve the data set into static volumes or go through the tedious process of migrating data between servers. When the file system approaches full, new nodes can be easily added and things will "just work."

Ceph includes flexible snapshot mechanism that allows a user to create a snapshot on any subdirectory (and its nested contents) in the system. Snapshot creation and deletion are as simple as 'mkdir .snap/foo' and 'rmdir .snap/foo'.

Ceph also provides some recursive accounting on directories for nested files and bytes. That is, a 'getfattr -d foo' on any directory in the system will reveal the total number of nested regular files and subdirectories, and a summation of all nested file sizes. This makes the identification of large disk space consumers relatively quick, as no 'du' or similar recursive scan of the file system is required.

Finally, Ceph also allows quotas to be set on any directory in the system. The quota can restrict the number of bytes or the number of files stored beneath that point in the directory hierarchy. Quotas can be set using extended attributes 'ceph.quota.max_files' and 'ceph.quota.max_bytes', eg:

```
setfattr -n ceph.quota.max_bytes -v 100000000 /some/dir
getfattr -n ceph.quota.max_bytes /some/dir
```

A limitation of the current quotas implementation is that it relies on the cooperation of the client mounting the file system to stop writers when a limit is reached. A modified or adversarial client cannot be prevented from writing as much data as it needs.

Mount Syntax

The basic mount syntax is:

```
# mount -t ceph user@fsid.fs_name=/[subdir] mnt -o mon_addr=monip1[:port][/monip2[:port]]
```

You only need to specify a single monitor, as the client will get the full list when it connects. (However, if the monitor you specify happens to be down, the mount won't succeed.) The port can be left off if the monitor is using the default. So if the monitor is at 1.2.3.4:

```
# mount -t ceph cephuser@07fe3187-00d9-42a3-814b-72a4d5e7d5be.cephfs=/ /mnt/ceph -o mon_addr=1.2.3.4
```

is sufficient. If /sbin/mount.ceph is installed, a hostname can be used instead of an IP address and the cluster FSID can be left out (as the mount helper will fill it in by reading the ceph configuration file):

```
# mount -t ceph cephuser@cephfs=/ /mnt/ceph -o mon_addr=mon-addr
```

Multiple monitor addresses can be passed by separating each address with a slash (/):

```
# mount -t ceph cephuser@cephfs=/ /mnt/ceph -o mon_addr=192.168.1.100/192.168.1.101
```

When using the mount helper, monitor address can be read from ceph configuration file if available. Note that, the cluster FSID (passed as part of the device string) is validated by checking it with the FSID reported by the monitor.

Mount Options

mon_addr=ip_address[:port][/ip_address[:port]]
: Monitor address to the cluster. This is used to bootstrap the connection to the cluster. Once connection is established, the monitor addresses in the monitor map are followed.

fsid=cluster-id
: FSID of the cluster (from *ceph fsid* command).

ip=A.B.C.D[:N]
: Specify the IP and/or port the client should bind to locally. There is normally not much reason to do this. If the IP is not specified, the client's IP address is determined by looking at the address its connection to the monitor originates from.

wsize=X
: Specify the maximum write size in bytes. Default: 64 MB.

rsize=X
: Specify the maximum read size in bytes. Default: 64 MB.

rasize=X
: Specify the maximum readahead size in bytes. Default: 8 MB.

mount_timeout=X
: Specify the timeout value for mount (in seconds), in the case of a non-responsive Ceph file system. The default is 60 seconds.

caps_max=X
: Specify the maximum number of caps to hold. Unused caps are released when number of caps exceeds the limit. The default is 0 (no limit)

rbytes
: When stat() is called on a directory, set st_size to 'rbytes', the summation of file sizes over all files nested beneath that directory. This is the default.

norbytes
: When stat() is called on a directory, set st_size to the number of entries in that directory.

nocrc
: Disable CRC32C calculation for data writes. If set, the storage node must rely on TCP's error correction to detect data corruption in the data payload.

dcache
: Use the dcache contents to perform negative lookups and readdir when the client has the entire directory contents in its cache. (This does not change correctness; the client uses cached metadata only when a lease or capability ensures it is valid.)

nodcache
: Do not use the dcache as above. This avoids a significant amount of complex code, sacrificing performance without affecting correctness, and is useful for tracking down bugs.

noasyncreaddir
: Do not use the dcache as above for readdir.

noquotadf
: Report overall filesystem usage in statfs instead of using the root directory quota.

nocopyfrom
: Don't use the RADOS 'copy-from' operation to perform remote object copies. Currently, it's only used in copy_file_range, which will revert to the default VFS implementation if this option is used.

recover_session=<no|clean>
: Set auto reconnect mode in the case where the client is blocklisted. The available modes are "no" and "clean". The default is "no".

  - no: never attempt to reconnect when client detects that it has been blocklisted. Operations will generally fail after being blocklisted.
  - clean: client reconnects to the ceph cluster automatically when it detects that it has been blocklisted. During reconnect, client drops dirty data/metadata, invalidates page caches and writable file handles. After reconnect, file locks become stale because the MDS loses track of them. If an inode contains any stale file locks, read/write on the inode is not allowed until applications release all stale file locks.

More Information

For more information on Ceph, see the home page at
https://ceph.com/
The Linux kernel client source tree is available at

- https://github.com/ceph/ceph-client.git
- git://git.kernel.org/pub/scm/linux/kernel/git/sage/ceph-client.git

and the source for the full system is at

https://github.com/ceph/ceph.git