NFSv4 Directory Performance Scalability

Chuck Lever
<chuck.lever@oracle.com>

Anecdotes

- Single application frequently inserts or deletes a directory entry. A typical NFS client invalidates its cache for that directory after every modification. This can mean repeatedly re-reading millions of entries.

- A directory can contain entries that are to be visible only to certain users. How will clients manage their directory caches when the server filters what entries are visible to which user?

- untar - NFS metadata operations are serialized with durable storage on NFS servers, making directory replication slow. Especially problematic for cloud data-center filesystems.

Directory Scalability Goals

- Help legacy applications that issue metadata operations sequentially: tar/untar, rm -rf

- Help applications that interleave readdir and directory modification

- Fewer client cache invalidations for directories with millions of entries

- Better security for tree-wide operations (*e.g.*, find)

The Usual Tricks

- Latency of durable storage on server - e.g. solid-state media
- Aggressiveness of caching - e.g., delegation
- Parallelism - e.g., pNFS
- Transport throughput - e.g., RPC-over-RDMA
- Server offload - e.g. COPY/CLONE

Possible Solutions

- Less ambiguous requirements for directory cookie persistence.
- Cheating (i.e. ignoring inconvenient POSIX requirements).
- Directory write delegation.
- Directory unlink offload.
- Enable client retrieval of a range of directory entries (directory trie, cookie ranges, etc).
- Sharding large directories across multiple storage servers.