Epochs

Alexander Zarochentsev (alexander.zarochentsev@sun.com)
based on the Epochs arch page by Nikita Danilov (nikita.danilov@sun.com)

2008-02-14

1 Introduction

In the following, an epoch is a collection of file system modifications, subject to certain constraints detailed below, that is used to merge distributed state updates (both data and meta-data) in a redundant cluster configuration.

Typical (in fact, motivating) redundant configurations are WBC, where redundancy is between objects on servers, and their copies cached by client, meta-data replication, and (clustered) meta-data proxy servers.

2 Definitions

For the purpose of the following discussion let's separate out two components in the cluster:

source a part of the cluster that caches some updated file system state, that is to be merged into destination. Source can be either a single client node (WBC case), a single server node (meta-data proxy), or a collection of server nodes (proxy cluster); destination a part of the cluster (server nodes) where updated state from the source is merged into.

destination can be either a next level proxy, or a home server; There might be multiple sources and destinations in the single cluster, arranged in a tree-like pattern: WBC client is a source whose destination is a proxy server, that, in turn is a source for its master.

file system consistency a file system is consistent when the state maintained on its servers, both on persistent and volatile storage together defines a valid name-space tree consistent with the inode/block allocation maps, and other auxiliary indices like FLD, SEQDB, OI, etc. File system consistency is only meaningful provided some notion of a logical time, making it possible to reason about collection of states across multiple nodes;

operation an unitary modification of user-visible file system state, that transfers file system from one consistent state to another consistent state. Typical example of an operation is a system call;

state update an effect of an operation on a state of an object it affects. Single operation can update state of multiple objects, possibly on multiple nodes;

undo entry a record in a persistent server log that contains enough information to undo given state update even if it were partially executed, or not executed at all;

epoch a collection of state updates over multiple nodes of a cluster, that is guaranteed to transfer source file system from one consistent state to another. Ultimate sources of epochs are: a WBC client:

for a single WBC client epoch boundaries can be as fine grained as a single operation;

- a single meta-data server: when MDS executes operations on behalf of non-WBC clients, in packs state updates into epochs defined by the transactions of the underlying local file system;
- a cluster of meta-data servers: when non-WBC client issues meta-data operation against the cluster of meta-data servers, this operation is automatically included into an epoch defined by the Cuts algorithm running on the cluster;

Once epoch is formed it traverses through the proxy hierarchy, and can be merged with other epochs, but cannot, generally, be split into smaller pieces.

epoch number a unique identifier of an epoch within particular source. For a single WBC client this can be as simple as an operation counter, incremented on every system call. For a single meta-data server, epoch number is tid (identifier of the last transaction committed to the local file system). For a proxy or replication cluster, epoch number is determined by the CUTs algorithm.

domain destination fs subset where WBC clients apply their changes

global epoch an epoch for non -WBC clients (old-style clients)

batch a collection of state updates across multiple objects on the same source node, such that (A) if a batch contains an update from some epoch, it contains all updates from this epoch belonging to this node (i.e., batch contains no partial epochs), and (B) the source node holds exclusive locks over the objects in the batch (at the logical level that is: in the presence of NRS locks might be already released);

epoch marker a special dummy record inserted in a batch that marks epoch boundary; sub-batch a set of state updates from the batch that are to be merged to the particular destination server. The batch is divided into the sub-batches;

batch reintegration a process of sending a batch over the network from the source to the destinations, followed by execution of state updates in the batch;

batch recovery a roll-forward of already reintegrated, but not committed batch in the case of destination failure; batch roll-back a roll-back of partially committed batch in the case of destination failure.

epoch dependency means one epoch cannot be undone without undoing another. Simple case of epoch dependency is subsiquent epochs from one source. More complex case is epochs with overlapping domains.

epoch referencing a reference of in-memory epoch object keeping the epoch and corresponding undo logs from purging.

epoch header meaningful for WBC-epochs, a header preceeding all batches containing epoch number and all participating servers.

3 Requirements

recoverability, atomicity - maintain cluster-wide consistent snapshots, atomic applying of client changes.

non-blocking - do not introduce delays in fs operations.

concurrency - parallel batch reintegration

isolation - independent epoch rollback, at least not completed WBC-epoch can be rolled back independently from other epochs

usability - use for WBC, replication, etc.

4 Functional specification

4.1 Overview

There are two different types of epochs. There is a global epoch for non-WBC clients, the disign of that borrowed from the CMD Rollback HLD document. And the WBC epochs, sourced by WBC-clients.

WBC-epochs of different sources and global epochs are logged independently on each server by server redo-logs.

The server undo-logs are used to restore cluster consistent state, by undoing not-completed epochs.

4.2 WBC-epochs

A WBC clients group individual fs operations into own epochs. WBC client has an epoch counter and informs the servers about epoch change (or add epoch number to each request).Each server has a specific undo-log for each WBC-client.

4.3 Global epochs

A brief explanation of CMD Rollback Design:

- The global epochs are sourced by one server (epoch generator) in the cluster. The role of epoch generator is periodically switched to another server.
- Each server has own current epoch. The server epochs are changed by explicit EPOCH_CHANGE requests or during a distributed transaction (op/depop, in CMD Rollback HLD terminology).
- A distributed transaction started after the servers have synchronized their epochs. Epoch switching might wait running disctributed transaction to complete or the epoch containing the transaction gets referenced and current epoch is increased.
- Meta data changes are recorded in server undo-logs, to allow not completed epoch to be rolled back.
- The controlling server gatheres info about committed batches info from all the servers, calculates and broadcasts a minimum committed epoch number to use for purging server undo-logs.
- more details are in CMD Rollback HLD.

4.4 Undo logs

Each epoch has own undo-log file. When the epoch is considered as no more needed for further recovery, the file is removed.

An undo-log file contains undo-records describing fs state before the change as well as epoch meta information: distributed epoch participants and epoch dependency records.

4.5 Distributed epoch commit, garbage collection issues

4.5.1 Distributed WBC-epochs

The WBC client sends special epoch header the servers participating the epoch. The epoch header contains a list of all those servers ids, so the participants know each other.

When a server commits all batches from the epoch, it exhanges batch commit information with other participants of the epoch. If it is known that all participant have committed all the epoch batches, the epoch can be marked as committed on all participating servers.

4.5.2 Global and WBC- epochs dependency

There is a dependency between global and WBC- epochs that affects purging of old epochs and corresponding undo logs.

Suppose a WBC client acquired locks during some global epoch GP0. Then WBC client started to modify fs in its WBC-epoch P1 during the same global epoch. It is impossible to undo GP0 without undoing P1. The WBC undo-logs no matter how many WBC-epochs there should be preserved until GP0 is committed.

In general, any epochs with overlapping domains depend on each other, regardless whether the epochs are global or not.

Tracking epoch dependencies

For tracking that dependency an in-memory database is used, anytime new lock is acquired, new dependency record may be added to the database.

Storing epoch dependency on disk

A list of epochs, depended to the given epochs, extracted from the database. The list is stored within epoch log file by records of special type.

When undo-log file should be kept from deletion?

4.5.3 Client redo-logs GC

Client redo-logs are purged when the corresponding epoch gets committed.

4.6 Recovery

4.7 Epoch transformation/aggregation

Proxy server case.

5 Use cases

6 Logic specification

7 State management

7.1 State invariants

7.2 Scalability & performance

7.3 Recovery changes

7.4 Locking changes

7.5 Disk format changes

7.6 Wire format changes

7.7 Protocol changes

7.8 API changes

7.9 RPCs order changes

8 Alternatives

9 Focus for inspections