On-Demand Zookeeper Backup and Restore with Streaming Capability

Background

Zookeeper is designed to withstand machine failures. A Zookeeper cluster can automatically recover from temporary failures such as machine reboot. It can also tolerate up to (N-1)/2 permanent failures for a cluster of N members due to hardware failures or disk corruption, etc. When a member permanently fails, it loses access to the cluster. If the cluster permanently loses more than (N-1)/2 members, it disastrously fails and loses quorum. Once the quorum is lost, the cluster cannot reach consensus and therefore cannot continue to accept updates.

To recover from such disastrous failures, we need a backup and restore mechanism. The usual approach of Zookeeper backup and restore is through customized scripts to copy data around, or through some 3rd party tools (exhibitor, etc), which have operation overhead. To improve efficiency, we need an automated and built-in backup and restore solution from Zookeeper.

When different databases such as Zookeeper and etcd are managed within an organization, we need on-demand backup and restore with streaming capability, so different databases can be backed up and restored through a generic external management platform.

This document captures the design of on-demand Zookeeper backup and restore with streaming capability.

Goals

- Supporting on-demand backup and restore so they can be performed whenever needed
- Supporting streaming capability for backup and restore so they can be managed by external systems

High Level Design

The following changes are proposed to provide on-demand backup and restore with streaming capability:

1. Providing a snapshot AdminServer API for taking a snapshot of Zookeeper database and streaming out the data
2. Providing a restore AdminServer API for restoring data from an input stream and rebuilding the in-memory database
3. Providing auth support on AdminServer APIs by leveraging the existing Zookeeper auth schemas and ACL check
4. Extending the ZK AdminServer framework to support streaming capability

Design Details

Snapshot Admin API

1. API Design
    a. HTTP GET method
    b. An optional parameter to disable streaming for testing purpose
    c. Returns snapshot data via HttpServletResponse OutputStream and the metadata of the snapshot via HTTP headers
        i. last_zxid: last processed zxid when the snapshot starts
        ii. snapshot_size: the size of snapshot
2. Implementation Details
    a. Adding a SnapshotCommand which performs the following
        i. handling auth and performing ACL check
        ii. applying all the transactions persisted in the transaction logs to the data tree
        iii. taking a snapshot and writing it to a file in the datadir
        iv. streaming out the data
    b. Adding serializeLastProcessedZxid() API to DataTree to serialize lastProcessedZxid in snapshot
    c. Adding rate limiting mechanism to ensure Zookeeper instance not overloaded by too many snapshot requests
    d. Modifying Commands class to support the new SnapshotCommand
    e. Changing ZookeeperServer.takeSnapshot() API and FileTxnSnapshot.save() API to return File object for steaming
    f. Documenting the new feature in ZookeeperAdmin

Restore Admin API

1. API Design
    a. HTTP POST method
    b. No request parameters
    c. Returns JSON response payload with the last restored zxid, i.e. last_zxid
2. Implementation Details
    a. Adding a RestoreCommand which takes an InputStream of snapshot data and rebuilds the Zookeeper database
    b. Adding rate limiting mechanism to ensure Zookeeper server not overloaded by too many restore requests
    c. Modifying Commands to support the new RestoreCommand
    d. Adding a new MAINTENANCE Zookeeper state
    e. Changing ZookeeperServer.submitRequest() to not enqueuing any request if ZK is in MAINTENANCE state
    f. Adding a new restoreFromSnapshot() API to ZookeeperServer, which rebuilds the database from the snapshot InputStream by performing the following:
        i. Rebuilding the new ZKDatabse by deserializing the snapshot
        ii. Stopping taking incoming requests by setting the Zookeeper state to MAINTENANCE
        iii. Setting the ZKDatabase to the newly restored one
        iv. Creating SessionTracker by ZookeeperServer.createSessionTracker()
        v. Setting the ZK state back to RUNNING
    g. Adding deserilizeLastProcessedZxid() to DataTree to restore the lastProcessedZxid from snapshot

Enhance AdminServer Framework to Support Streaming

1. Zookeeper AdminServer framework currently only supports JSON string response. For snapshot, we need to extend it to stream out a byte array response. The following changes are proposed for providing streaming capability:
    a. Extending JettyAdminServer.CommandServlet.doGet() method to support streaming byte array response
    b. Adding a new output() API to the CommandOutputter for streaming HTTP response
    c. Adding a StreamOutputter class which extends CommandOutputter. It gets an InputStream object from CommandResponse and writes to the ServletOutputStream as response
    d. Extending CommandResponse to take InputStream for streaming HTTP response
2. For restore, we need to extend the ZK AdminServer framework to support HTTP POST and streaming capability. The following changes are proposed:
    a. Extending JettyAdminServer.CommandServlet to support doPost()

b. Modifying Commands.runCommand() signature to take an additional InputStream as input parameter

Auth Support on AdminServer APIs

Zookeeper enforces security by checking the auth info submitted by the Zookeeper client against access to each znode, however there is no auth support for performing the AdminServer commands. With adding snapshot and restore commands, the scope of AdminServer APIs is expanded from “reading some data from the server” to “performing some operations on the server”. The following is proposed to provide auth support on AdminServer APIs in general and snapshot and restore particularly:

- Providing auth check by leveraging Zookeeper existing auth schemas and ACL check
    - Using HTTP authorization header to pass auth schema and auth info if applicable (for example, digest). For x509 and IP, the auth info is extracted from HttpRequest, so only the auth schema needs to be passed.
    - Supporting different auth schemes including digest, x509 and IP.
    - Supporting SASL is not in scope because HTTP protocol doesn’t support and work well with SASL.
- Snapshot and restore commands can only be performed with HTTPS
- Performing ACL check against permission ALL on the root znode for the snapshot and restore commands

JIRA Tickets

- Snapshot and Restore Command Initiative
    1. https://issues.apache.org/jira/browse/ZOOKEEPER-4570
    2. https://issues.apache.org/jira/browse/ZOOKEEPER-4571

- Providing Auth Support for Admin Server APIs Initiative
    1. https://issues.apache.org/jira/browse/ZOOKEEPER-4639