Distinguish Checkpoint/Savepoint Format and Support Stop-with-checkpoint

Yu Li

In this document we will firstly discuss the concept of checkpoint and savepoint in comparison of database snapshots and backups, and try to clarify why these two are not only conceptually but also should be physically different. And then we will explain why stop with checkpoint is a better choice to support job quick recovery and how to resolve the "system-control leak" concern.

As described in our Flink document [1], the concept of checkpoint and savepoint could be mapped to database mechanisms, and below are more details:

The use-case of savepoint in Flink includes:

- Recover from failure
- Refresh job environment
  - Rescale or change job graph
  - Upgrade and state migration
- Switch state backend
- Import and export (FLIP-43 [2])
- Fork job for blue/read deployment

Relatively, database use backups [3] to:

- Recover from accidental deletes
- Refresh development environments
- Migrate databases or switch storage engine [4]
- Import and export data
- Create database copies for testing, training and demonstrations

On the other hand, in our document we analogize checkpoint to recovery log in database. While this makes sense according to the well known streaming based paradigm (as indicated in figure 1), for the time being snapshot maps better with checkpoint than recovery log.

Figure 1. Streaming Based Paradigm

As documented [1], checkpoint is to provide a recovery mechanism with below design goals:

1) Light-weight to create
2) Fast to restore from

Which is similar to the usage of snapshots to provide a light-weight way of making a "point-of-time" copy of a database [3].

From above analysis, we could tell the difference between checkpoint and savepoint referring that of snapshot and backup in database [3] [5]:

- Not only conceptually but also physically different.
  - Snapshots only consist of pointers of pages and stores in format of sparse file, while backups contain real data.
- Performance of system may be impacted during backup but snapshot is more lightweight.

However, currently in Flink checkpoints are not fully physically different with savepoints (savepoint and full-checkpoint use the same format, while incremental-checkpoint is different), which needs some improvement and FLIP-41 [6] (unify savepoint format while keeping the diversity of checkpoint format between backends, thus distinguish savepoint and checkpoint formats) is a good start.

Since database backups have 3 different types: full, differential and transaction log, in the long run we should also consider to support more feasible types of savepoint.

Regarding stop with checkpoint, since in database we will flush everything out when stop the instance and only replay redo log during failover, job termination always be accompanied by a checkpoint sounds reasonable in Flink (and only do source rewinding during job failover).

In conclusion, we propose to:

1. Differentiate checkpoint and savepoint formats to keep the concept clean
2. Support stop with checkpoint through configuration
   a) This combines with the retained checkpoint feature.
   b) This mainly targets at meeting the user requirement of recovering from the status where it stops as well as stopping job as quickly as possible.
3. Consider supporting more feasible types of savepoint in the long run
   a) The “savepoint with optimized format” proposal is good for supporting differential backup, but should be implemented in a different way than “using incremental checkpoint format in savepoint”. It’s a good supplement for the existing full backup mechanism in Flink, but will still be much slower than snapshot (incremental checkpoint) decided by the nature of backup (supporting much more use-case leads to complexity)

[1]
https://ci.apache.org/projects/flink/flink-docs-release-1.8/ops/state/savepoints.html
[2]
https://cwiki.apache.org/confluence/display/FLINK/FLIP-43%3A+Savepoint+Connector
[3]
https://www.sqlshack.com/backup-and-restore-or-recovery-strategies-for-sql-s

erver-database
[4]
https://github.com/facebook/mysql-5.6/wiki/Migrating-from-InnoDB-to-RocksDB
[5]
https://www.oracle.com/technetwork/database/availability/rman-fra-snapshot-322251.html
[6]
https://cwiki.apache.org/confluence/display/FLINK/FLIP-41%3A+Unify+Keyed+State+Snapshot+Binary+Format+for+Savepoints