APACHE COTTON

MySQL on Mesos

Yan Xu 🦆xujyan
SHORT HISTORY

- Mesos: cornerstone of Twitter’s compute platform.
- MySQL: backbone of Twitter’s data platform.
- Mysos: started as a hackweek project @twitter.
- Apache Cotton: fluffy, elastic & cloudy.
- Call for collaboration / contribution!
WHY MYSQL ON MESOS

• From manual MySQL administration to self-service.

• Distributed systems should manage themselves.
WHY COTTON

• Reusability vs. flexibility.

• Generic schedulers: stateless tier one services.

• MySQL: stateful & complex.

• Interactive: Cotton coordinates MySQL instances during their lifecycles.

• Push common functionality down!
COTTON INTRO
MYSQL ON A MESOS CLUSTER

Services

Spark

frameworks

to

to

to

mesos-master

mesos-slaves

S

S

S

S

S

S
MYSQL ON A MESOS CLUSTER

Services

frameworks
- cotton-scheduler
- mesos-master
- mesos-slaves

Spark

Mesos-Slaves

Mesos-Master
MYSQL ON A MESOS CLUSTER

Services

Spark

frameworks

cotton-scheduler

mesos-master

mesos-slaves

cotton-executor
MYSQL ON A MESOS CLUSTER

Services

Spark

frameworks

mesos-master

mesos-slaves

cotton-executor
MYSQL ON A MESOS CLUSTER

Services

frameworks  mesos-master  mesos-slaves
cotton-executor

Spark
“COTTON CLUSTERS”

- Cotton cluster
- MySQL cluster

Mesos cluster

- Cotton cluster
- MySQL cluster
SCHEDULER + EXECUTOR

- Scheduler: global view, central scheduling, react to events (from Mesos and executors), coordinate instances (e.g. Elector).

- Scheduler deploy, upgrade, HA, service discovery, managed by Aurora, state stored in ZK.

- Executor: Bootstrap, launch, monitor, interact.
CREATE CLUSTER
'cluster0': nodes=3,
size=SMALL, user, passwd,
backup_id

Cotton Scheduler

Offers

mesos-master

mesos-slaves
Cotton Scheduler

mesos-master

mesos-slaves

LAUNCH TASK

LAUNCH TASK

LAUNCH TASK

LAUNCH TASK

LAUNCH TASK

LAUNCH TASK
Master = Slave(MAX(GTIDs))

Cotton Scheduler

GTIDs

mesos-master

mesos-slaves
ELECT A

Cotton Scheduler

A elected

mesos-master

mesos-slaves
Master: A
Slaves: B, C

Cotton Scheduler

zk://

mesos-master
discharged

mesos-slaves

A PROMOTE
B REPARENT
C REPARENT
ELECT B

Instance A unhealthy

Host C down

Cotton Scheduler

mesos-master

mesos-slaves
Cotton Scheduler

LAUNCH TASKS D,E

mesos-master

mesos-slaves
SERVICE DISCOVERY

• MySQL connection: ip, port, user, password

• API input: cluster name, user(, password)

• API out: (immediate) ZooKeeper path(, password)

• Cotton publishes to ZK:
  • ip:port
  • master/slave roles
DISCOVERY MECHANISMS

- ServerSets w/ ZK libraries
- Proxying (haproxy-marathon-bridge)
- DNS (mesos-dns)
- Scheduler REST API
- Mesos commons library
MONITORING

- Watching ZK group:
  - When is the MySQL cluster ready? (e.g. in an hour)
  - When has old master died & a new master elected?
- Monitoring stats: Is the cluster healthy? Is Cotton service healthy?
  - `<scheduler>/vars.json`
  - Mesos exported stats (container stats, `<slave>/monitor/statistics.json`)
  - Executor exported stats (MySQL specific)
USING COTTON
CUSTOMIZING COTTON

• Organization level (e.g. Twitter)
  • Merge custom scripts and configs.
  • Implement Installer, BackupStore interfaces.

• Cotton cluster level (e.g. Devel): scheduler flags.

• MySQL cluster level (e.g. Ads): API request arguments.
BACKUP STORE

• Discover, fetch, decrypt, extract, post-processing, etc.

• Customization: TwitterBackupStore

• Flags: --backup_store_args. (e.g. HDFS args)

• API args: backup_id.
  
• Organization specific: group, cluster, partition, timestamp, etc.

• Supporting defaults and conventions: latest, etc.
INSTALLER

- Discover, fetch, install.

- Customization: TwitterPackageInstaller
  - Twitter uses its own MySQL releases (with its own custom config variables)
  - Auxiliary utils and libs.

- Flags: --installer_args. (e.g. Package management args)

- API args: package.
  - release, version, tags (latest, live), etc.

- Filesystem isolation (target 0.25): MySQL as a separate layer.
CONFIGS AND SCRIPTS

- Customization: Executor files
  - e.g. my.cnf.
  - Unpacked to disk upon launch.
- Flags: --executor_environ.
  - JSON => ExecutorInfo::command::environment.
  - e.g. MYSOS_DEFAULTS_FILE=<custom_my_cnf>.
DEPLOYING COTTON

- Cotton (PyPI) + Your customization => Your Cotton.

- **PEX**: Python EXecutable with all its dependencies bundled.

- `cotton_executor.pex`: fetchable by `mesos-fetcher`.

- `cotton_scheduler.pex`: managed by Aurora.
DEVELOPING COTTON
RETAINING MYSQL STATE

- Persistent volumes: available since 0.23.
- Reserve hosts for Cotton through roles.
- Scheduler: create persistent volumes.
- Executor: sandbox bind mounted from a persistent volume (a directory managed by Mesos and not GCed).
- Scheduler: look for the old volumes from offers and reuse if possible.
ROAD AHEAD

• Client auth & authz; admin privilege.
• Replicate and failover using GTID.
• Isolate disk IO on shared hosts.
• Multiple MySQL versions per Cotton cluster.
• Scheduling constraints.
• Add instances to existing clusters.
RESOURCES

- https://github.com/apache/incubator-cotton
- https://issues.apache.org/jira/browse/cotton
- #apache-cotton on freenode.org
- https://twitter.com/apachecotton
- dev-subscribe@cotton.incubator.apache.org