Introduction of Mesos persistent storage

Weitao Zhou @ DataMan

Content

- How to run stateful service against current Mesos-0.22
- Disk isolation and monitoring
- Persistent Volumes
- Dynamic Reservations
- What we can contribute for Mesos persistent storage

How to run stateful service against current Mesos-0.22

- Making it outside of Mesos cluster
- Storing data into local filesystem of the specified Mesos slave
- Setting DFS to get rid of static reserve
- Taking advantage of stateful service build-in data-replication feature

Making it outside of Mesos Cluster

- Scenarios：
  - independent cluster: RabbitMQ cluster
  - constant resource usage, etc.
- Pros:
  - mature solution
  - effortless
- Cons:
  - decreasing resource usage

Storing Data into Local Filesystem of the Specified Mesos Slave

- Scenarios：
  - Data disk is NOT included into Mesos resource
  - Avoid of data revoked by Mesos
  - The next task can restore data
- Pros:
  - Using the CPU, RAM, network resource of the Mesos cluster: one single dockerized MySQL task
- Cons:
  - Resource STATIC RESERVED to make sure the stateful service can be launched on the specified slave successfully always
  - Mesos can NOT release the outdated persisted data, have to manually do it
  - Data conflict: avoid it by launching one task per slave temporarily
  - how to share the data for HA?

Setting DFS to get rid of static reserve

- Pros:
  - HA
  - Get rid of static reserve
- Cons:
  - network delay of data transfer. that can be accepted by MySQL/PG?
  - The network resource taken by data transfer is OUT of CONTROL: Setting another data transfer network to avoid it? => decreasing the network resource usage again

Taking advantage of stateful service build-in data-replication feature

- Scenarios：
  - Multiple stateful services are supporting data-replication: Cassandra, MariaDB Galera, MongoDB, etc.
  - Cassandra on Mesos: https://github.com/mesosphere/cassandra-mesos
  - MariaDB Galera on Mesos: http://sttts.github.io/galera/mesos/2015/03/04/galera-on-mesos.html
- Pros:
  - Network is in control via task build-in data replication
  - Stateful service itself is responsible for network delay
- Cons:
  - Persistent disk is outside of cluster still

in a nutshell

To persistent data, we have to store data outside of the cluster currently

Disk Isolation and Monitoring

- To keep other tasks running, Mesos is restricting disk quota per task

Mesos disk is a GENERIC disk

- Separated Filesystem
    - Created by physical disk/LVM etc.
    - Signal ENOSPC triggered once data size expanded
    - Task is terminated
    - Hard enforcement
    - Adapt to production environment
- Shared Filesystem
    - Sharing file directory with other tasks
    - Monitoring “disk” usage by executing “du” periodically
    - Tolerating data size expansion
    - Soft enforcement

Shared Filesystem needs disk isolator

- Support Mesos build-in container only.
- Map container path to slave host path by command “mount -n --bind”. Kernel will umount it automatically after task finished, at the same time.
- Use docker volume mapper for docker container

Disk IO resource

- Heavy-disk-IO task is throating other tasks maybe
- Try “Cgroups blkio controller” for disk IO isolation in future

Persistent Volumes

- Data in persistent volumes WON’T be GC after task completed
- Another task can restore last finished task data
- Data survives even slave info/id changed or rebooted
- Belonging to Mesos cluster resource, handled by Mesos cluster

Regular Resource VS. Persistent Resource

- Regular resource
    - is renamed from current 0.22 resource
    - match stateless task
    - CPU, RAM, Disk will be GC after task completed
- Persistent resource
    - Besides persistent volumes, CPU, RAM is included also. WHY?
    - match stateful service
    - data in persistent volume is reserved after task completed
    - be validated by google Borg

Resource Offer with persistent resource

```
{“id” : { “value” : “offerid-123456789”},
 “framework_id” : “MYID”,
 “slave_id” : “slaveid-123456789”,
 “hostname” : “hostname1.prod.twttr.net”
 “resources” : [
  // Regular resources.
  { “name” : “cpu”, “type” : SCALAR, “scalar” : 10 }
  { “name” : “mem”, “type” : SCALAR, “scalar” : 12GB }
  { “name” : “disk”, “type” : SCALAR, “scalar” : 90GB }
  // Persistent resources.
  { “name” : “cpu”, “type” : SCALAR, “scalar” : 2,
    “persistence” : { “framework_id” : “MYID”, “handle” : “uuid-123456789” } }
  { “name” : “mem”, “type” : SCALAR, “scalar” : 2GB,
    “persistence” : { “framework_id” : “MYID”, “handle” : “uuid-123456789” } }
  { “name” : “disk”, “type” : SCALAR, “scalar” : 10GB,
    “persistence” : { “framework_id” : “MYID”, “handle” : “uuid-123456789” } }
 ]
 ...
}
```

Shared persistent data

- MySQL framework maybe launch 2 tasks accessing the same persistent data, for example
  - mysqld server
  - data backup periodically
- under discussion still

Dynamic Reservations

- Why reservations? Make sure specified framework running on specified slave(s)
- What reserved? CPU/RAM/DISK/Network
- How to reserve? Static reserve by setting slave role currently

Dynamic Reservations

- Framework can broadcast reserving resource dynamically, whatever regular or persistent resource, by setting “reserved_role”, when launch a task
- The “reserved_role” can be reoffered to the same framework after task completed
- Framework can release the dynamic reserved resource by itself, while can NOT release the before static one
- Very different from the before static reserve by setting slave role

What we can contribute for Mesos persistent Storage

- acentric or node-equal stateful service
  - Cassandra, for example
  - effortless: dockerize cassandra program => distribute to the multiple specified slaves over scheduler
- master-slave or leader-follower stateful service
  - HDFS, MongoDB, MySQL, for example
  - name node, data node, config node
  - develop different framework for different service
  - Let the framework solve fault-tolerant, backup issue

What we can contribute for Mesos persistent Storage

- know more the stateful service content
  - mail-list
  - source code
  - doc
- try to develop the framework or dockerize the program
- solve the fault-tolerate, backup issue
- avoid to re-invent wheel

Q&A

We are hiring

- Python
- Django
- AngularJS
- Mesos
- Docker
- Linux
- Git
- opensource
- shell
- 3+ years
- 望京
- line 14/15 subway
- work with googler, red hatter, 活力四射er
- more geek, more money

mailto:jjyan@dataman-inc.com

Thanks