MesosCon Asia 2016

Getting Rid of Zookeeper

Jay Guo

Software Developer @IBM
guojiannan@cn.ibm.com

Kapil Arya

Mesos Committer @Mesosphere
kapil@mesosphere.io

Motivation

Zookeeper is...

- Mature
- Feature-rich
- ...

But!

- Primitive K/V store
  - Provide your own tooling for other abstractions!
- Heavy
- Hard dependencies
- Language binding instead of RESTfull API
- ...

It’s all about having options!

- **Chocolate**
- **Strawberry**
- **Vanilla**
- …

Mesos HA:
An Overview

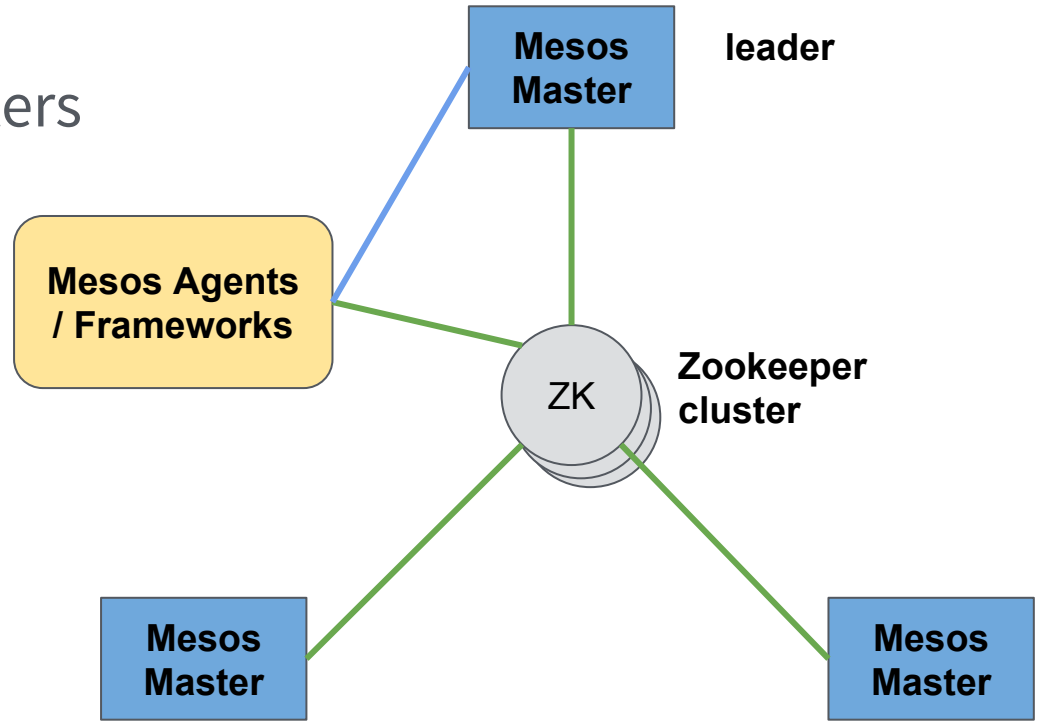
High Availability

First of all, we need a Distributed **Key-Value storage...**

Mesos HA

- At least three Mesos Masters
- One leading Master
  - Leader **election**
  - Leader **detection**

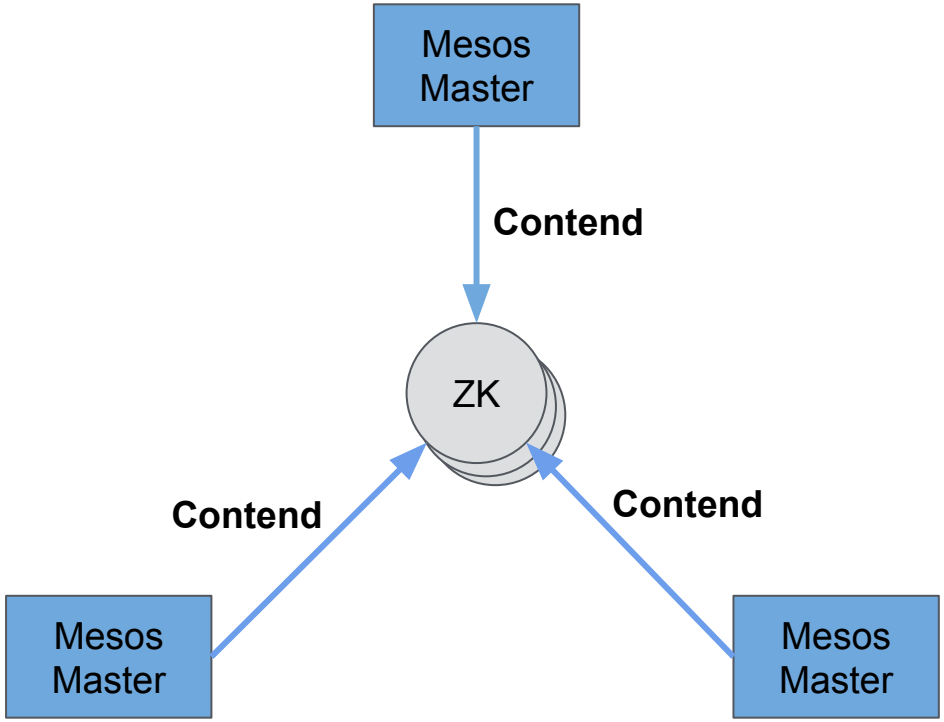
- **Replicated Log**



Zookeeper as the distributed key-value store

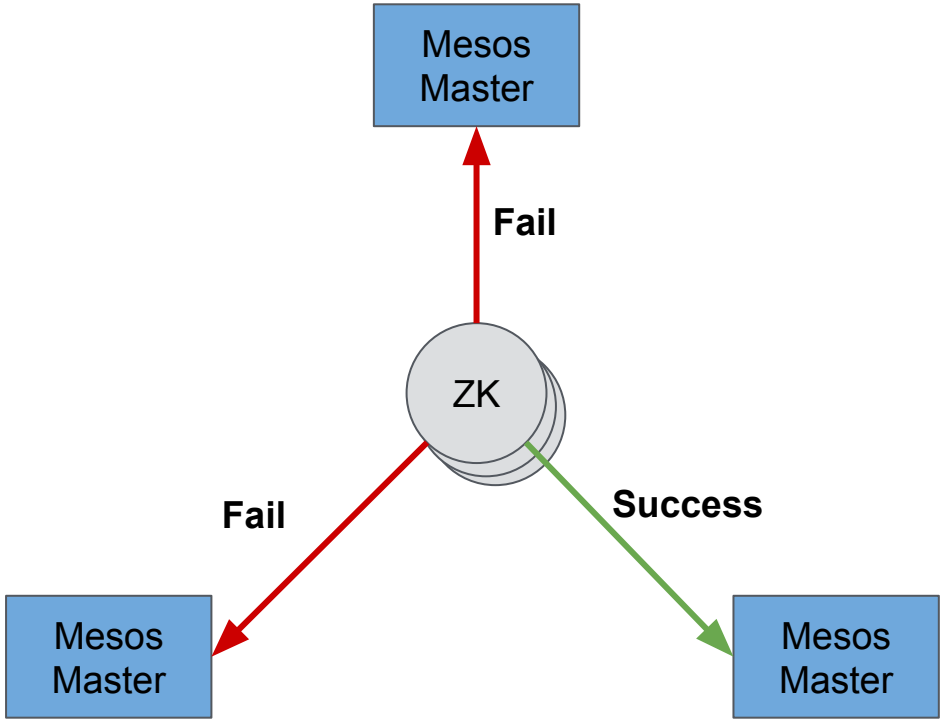
Mesos HA: Leader Election

- All Masters “contend” to be the leader!



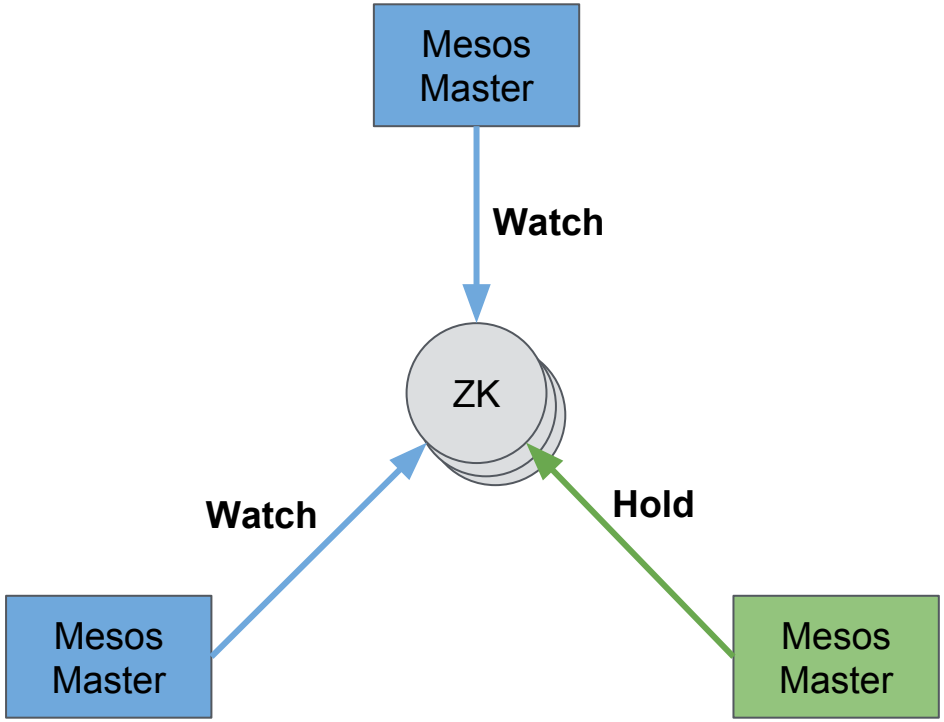
Mesos HA: Leader Election

- All Masters “contend” to be the leader!
- Only **one** succeeds; others **fail**



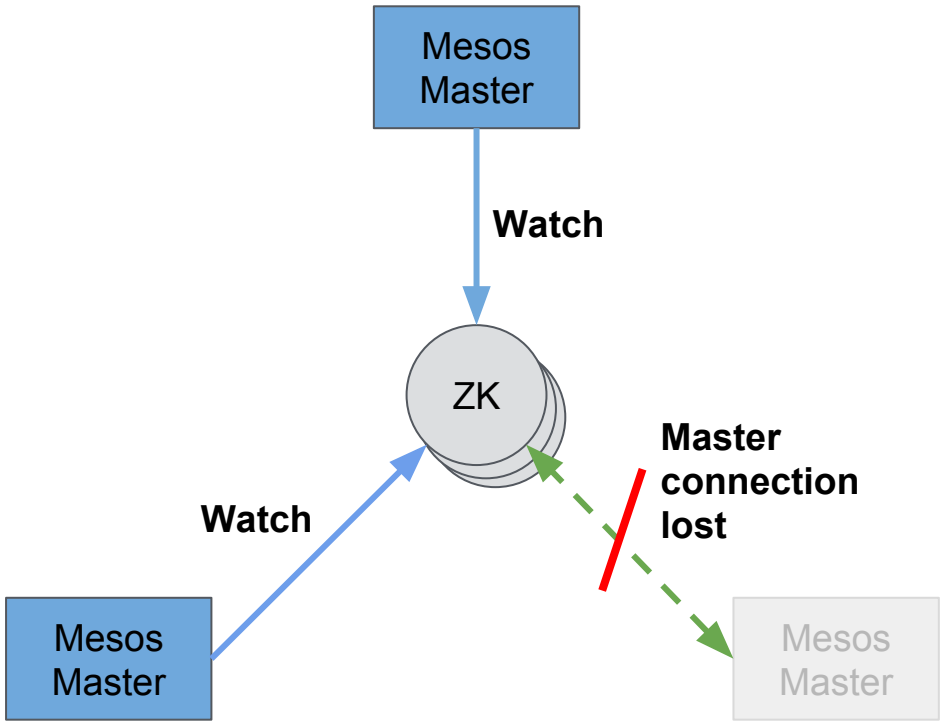
Mesos HA: Leader Election

- All Masters “contend” to be the leader!
- Only **one** succeeds; others **fail**
- We have a leading Masters!



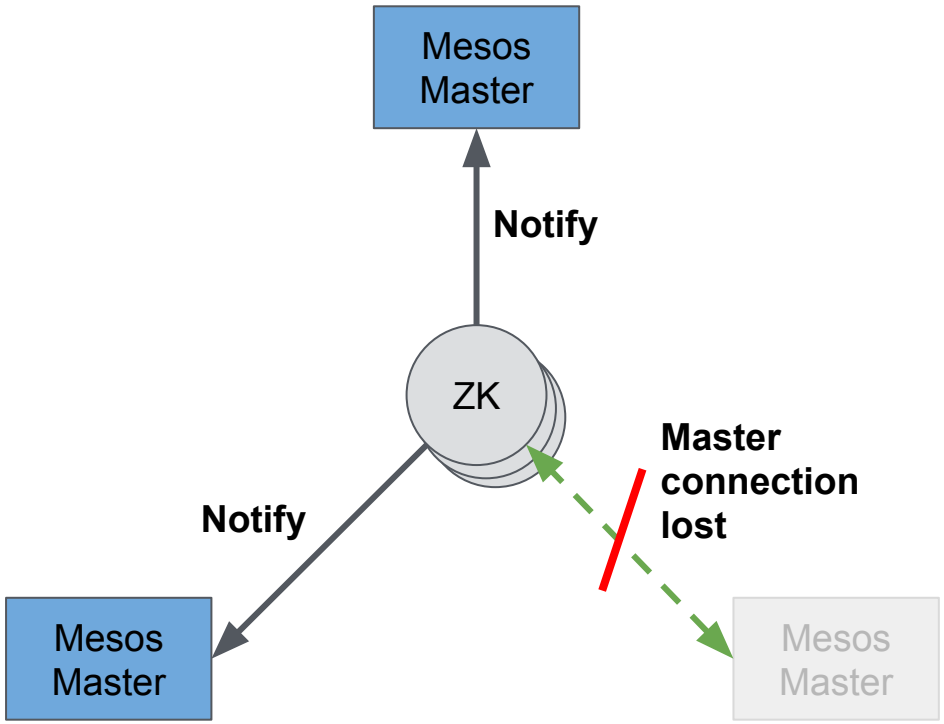
Mesos HA: Losing a Leader

- Suppose the leading Master is "**lost**"



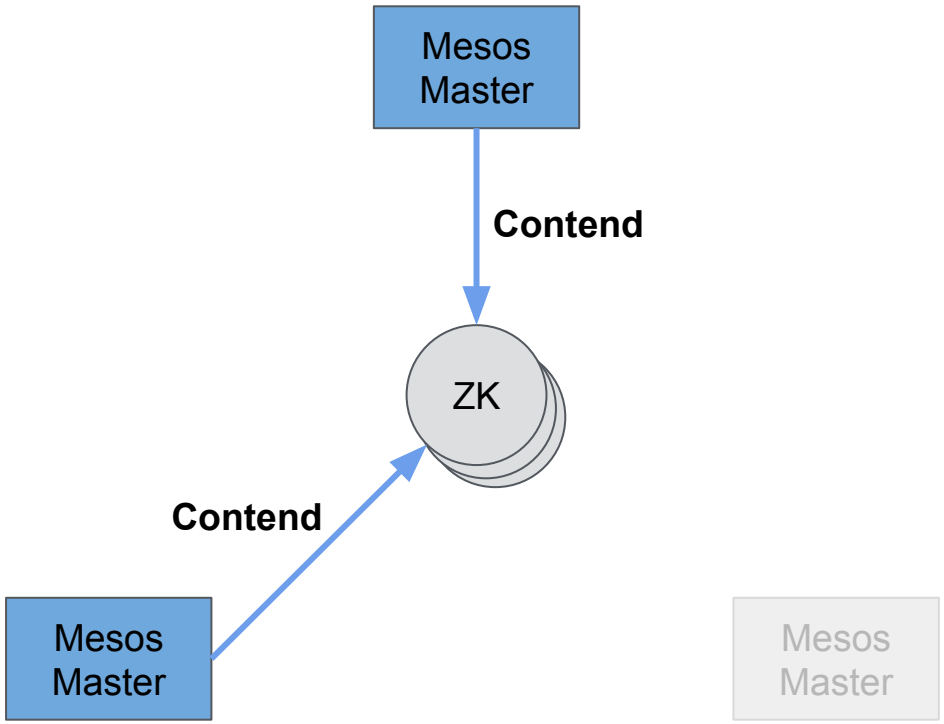
Mesos HA: Losing a Leader

- Suppose the leading Master is "**lost**"
- All other Masters are notified



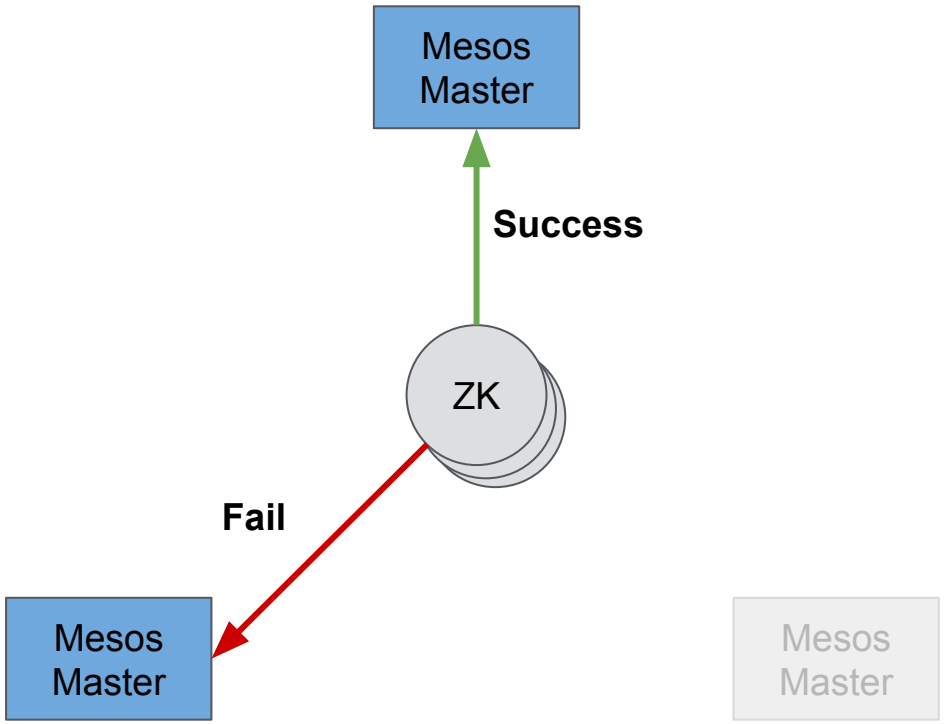
Mesos HA: Losing a Leader

- Suppose the leading Master is "**lost**"
- All other Masters are notified
- The remaining Masters contend again



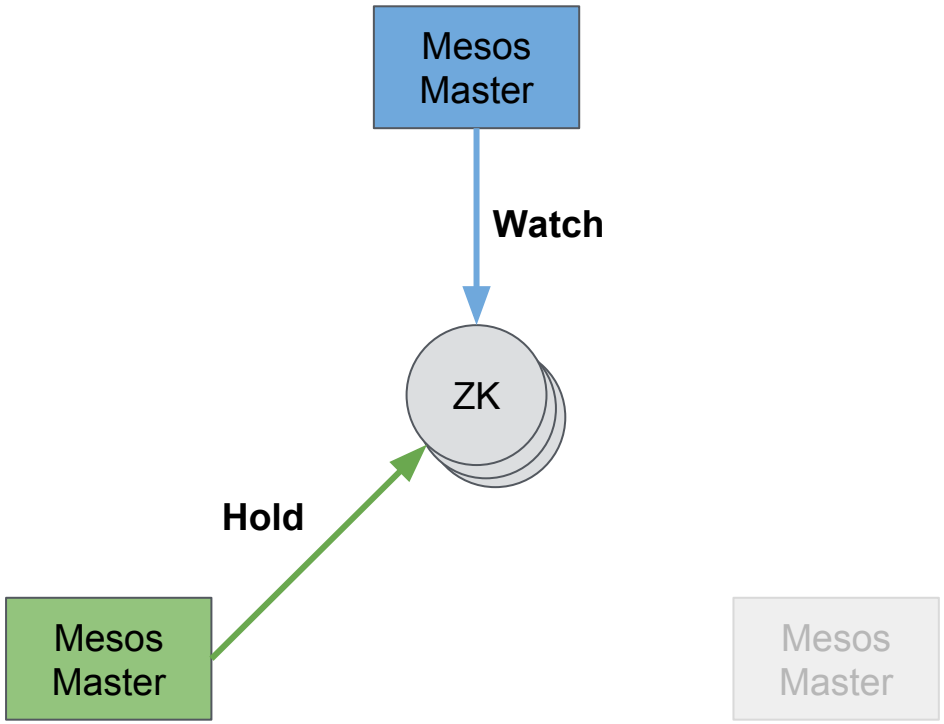
Mesos HA: Losing a Leader

- Suppose the leading Master is "**lost**"
- All other Masters are notified
- The remaining Masters contend again
- One of them succeeds



Mesos HA: Losing a Leader

- Suppose the leading Master is "**lost**"
- All other Masters are notified
- The remaining Masters contend again
- One of them succeeds

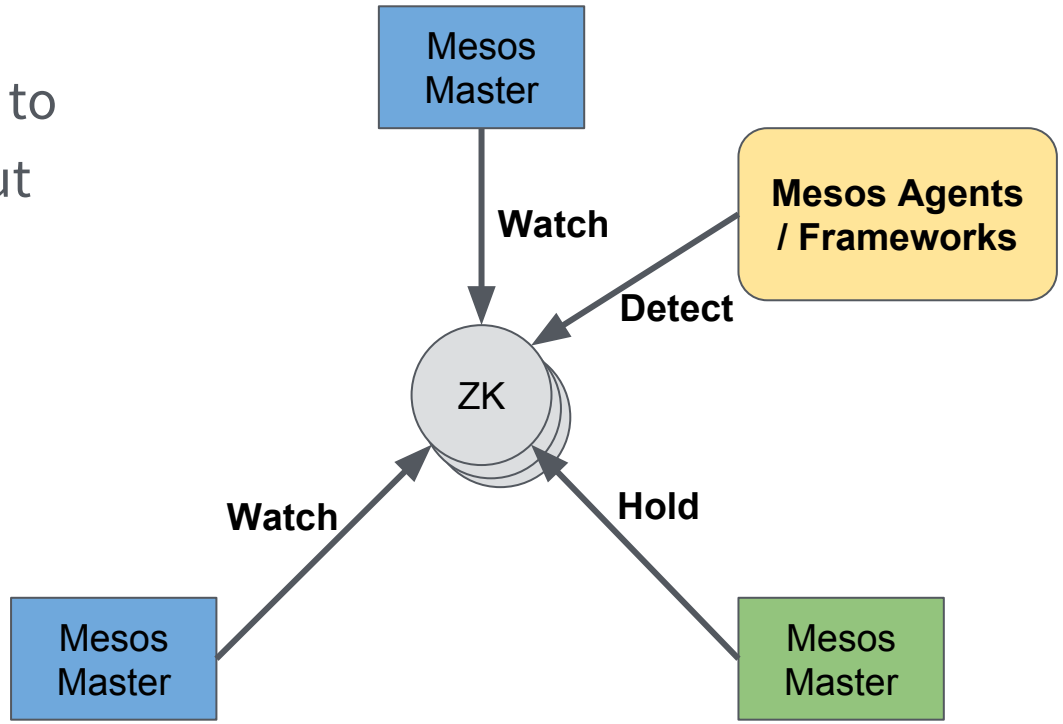


- A new leader is elected!

What about Agents/Frameworks?

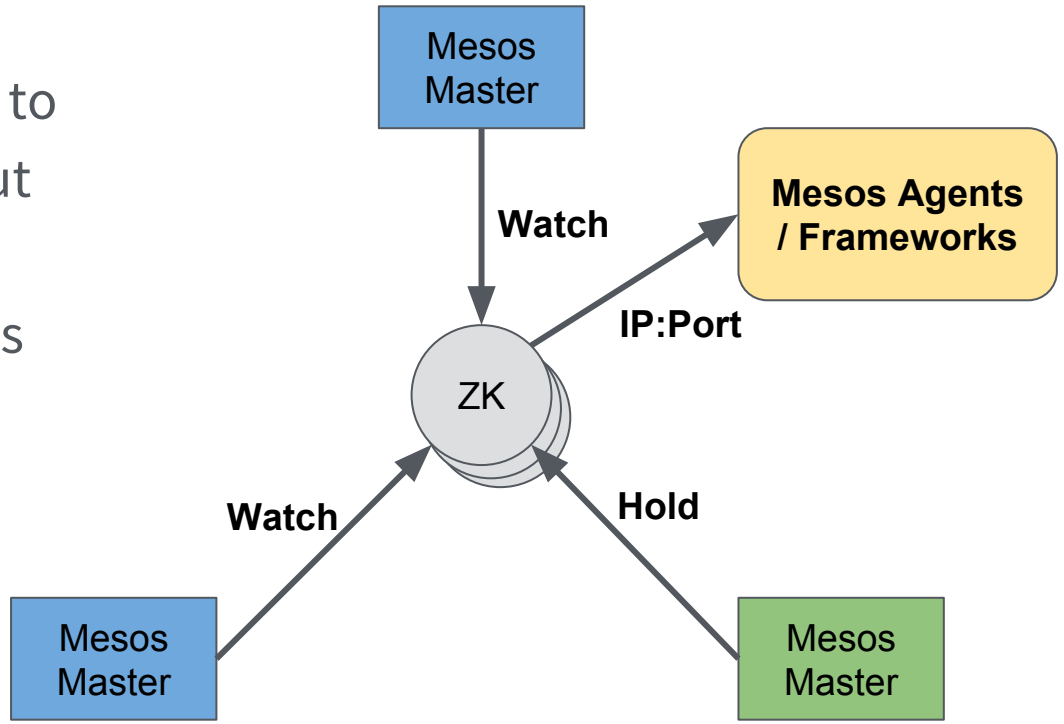
Mesos HA: Leader Detection

- Framework/Agent connects to Zookeeper to "**detect**" about the current leading Master



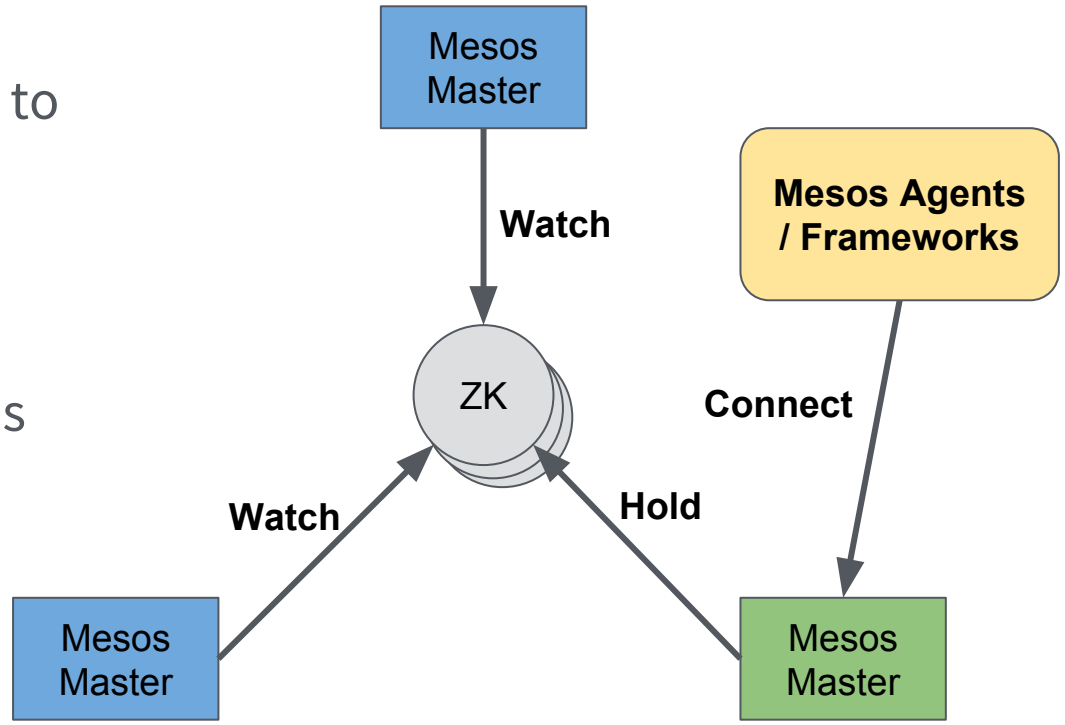
Mesos HA: Leader Detection

- Framework/Agent connects to Zookeeper to "**detect**" about the current leading Master
- Zookeeper provides Master's location
  - I.e. IP:Port



Mesos HA: Leader Detection

- Framework/Agent connects to Zookeeper to "**detect**" the current leading Master
- Zookeeper provides Master's location
- Framework/Agent connects to the "**leader**"

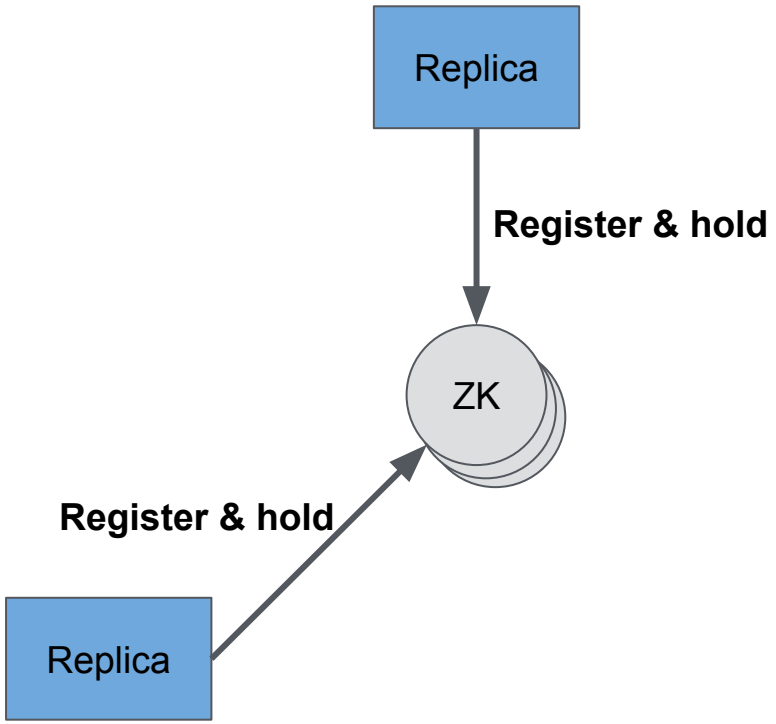


What about Replicated Log?

Replicated Log lets you create replicated fault-tolerant append-only logs. The Mesos master uses Replicated Log to store cluster state in a replicated, durable way.

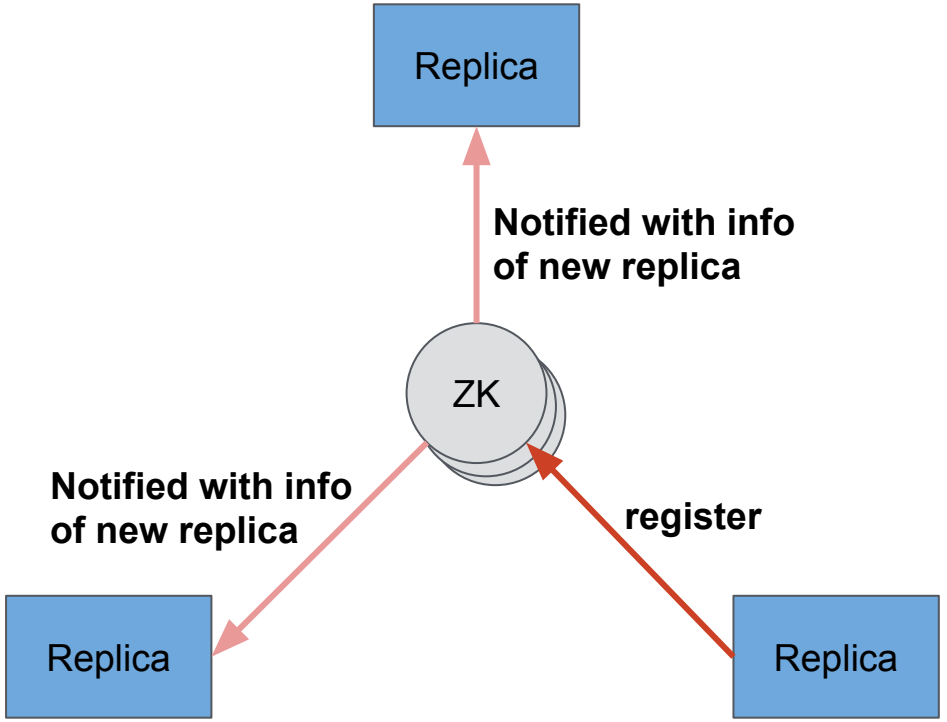
Mesos HA: Replicated Log

- Each replica registers its **pid** into ZK and maintain the presence.



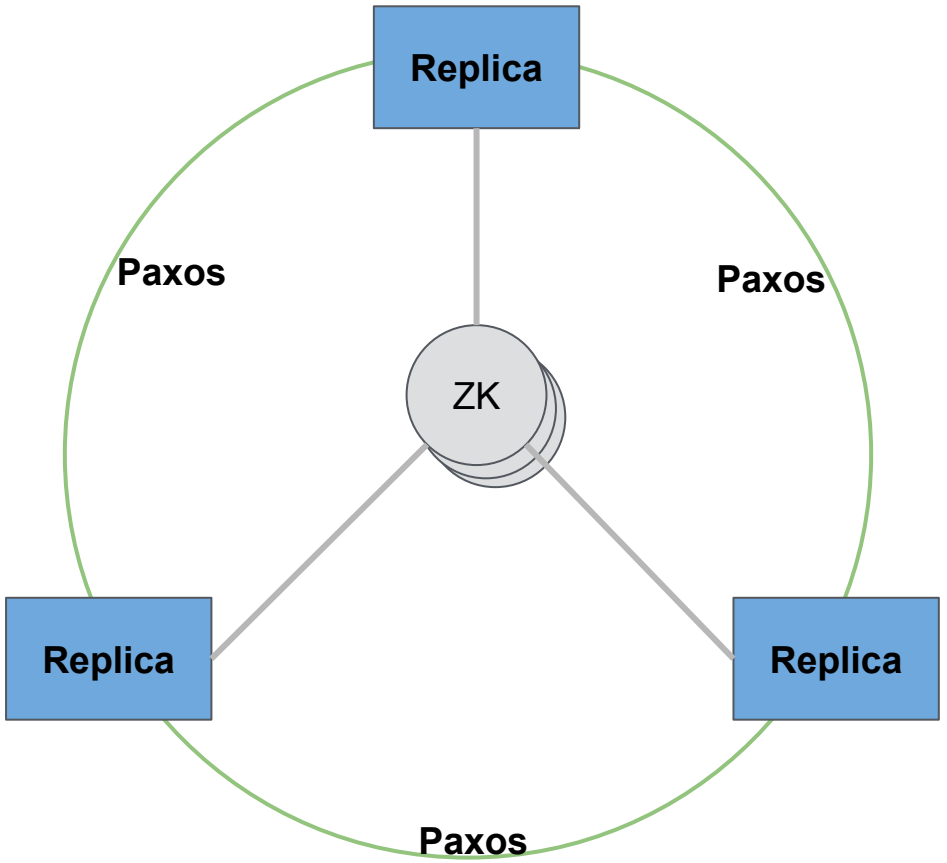
Mesos HA: Replicated Log

- Each replica registers its **pid** into ZK and maintain the presence.
- When new replica joins the cluster, existing ones get notified and get to know the pid of new replica.

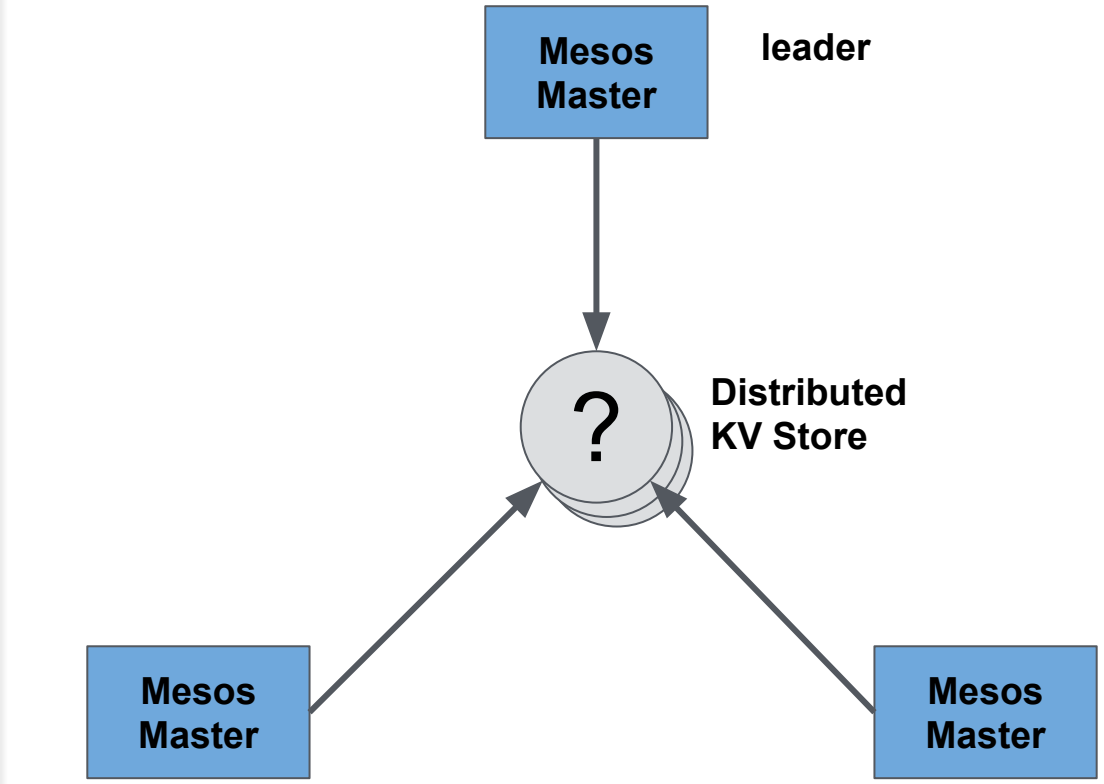


Mesos HA: Replicated Log

- Each replica registers its own **pid** into ZK and maintain the presence.
- When new replica joins the cluster, existing ones get notified and get to know the pid of new replica.
- Every replica knows all nodes in the cluster and do **Paxos**.



Replacing Zookeeper

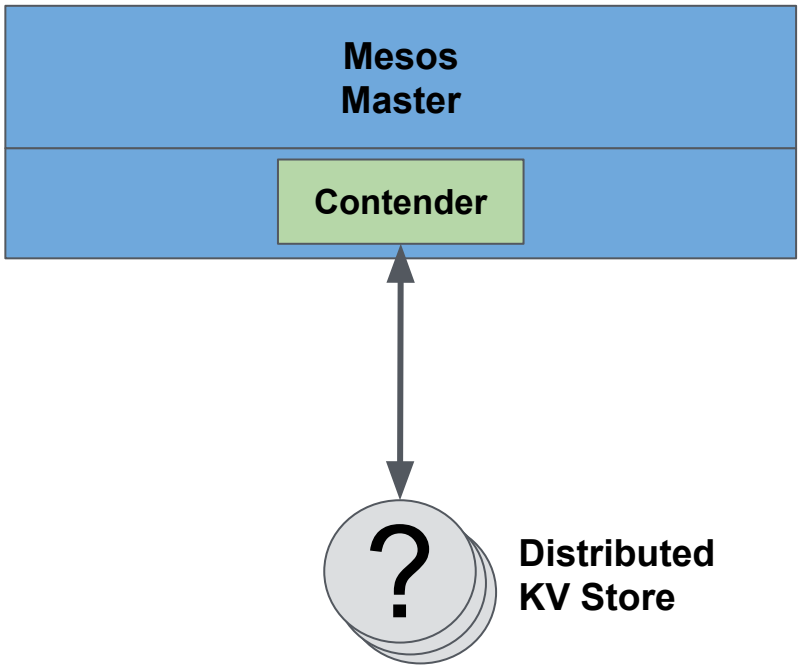


? = ZK || Etcd || Consul ...

Three Key Components

- Master **Contender** for leader election

```
bool contend();
```



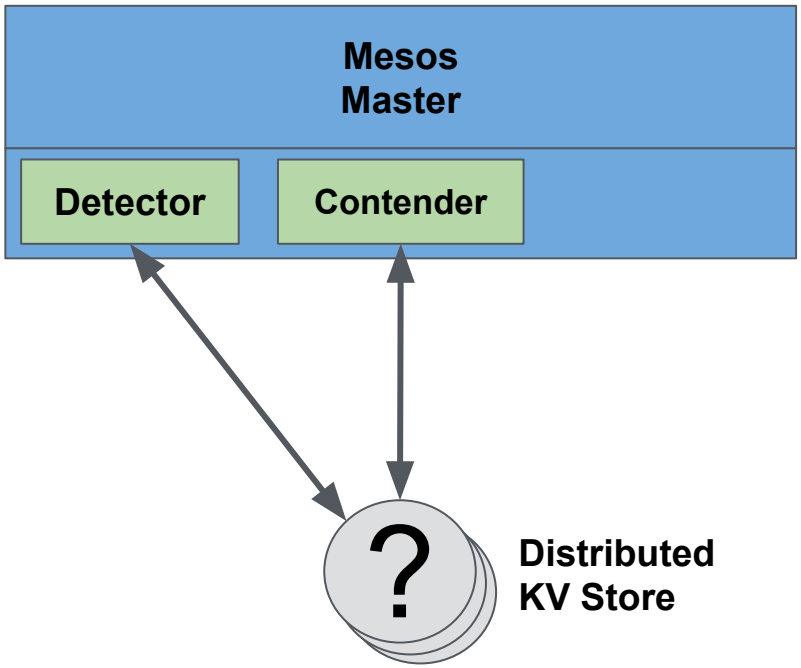
Three Key Components

- Master **Contender** for leader election

```
bool contend();
```

- Master **Detector** for discovery

```
MasterInfo detect(MasterInfo previous);
```



Three Key Components

- Master **Contender** for leader election

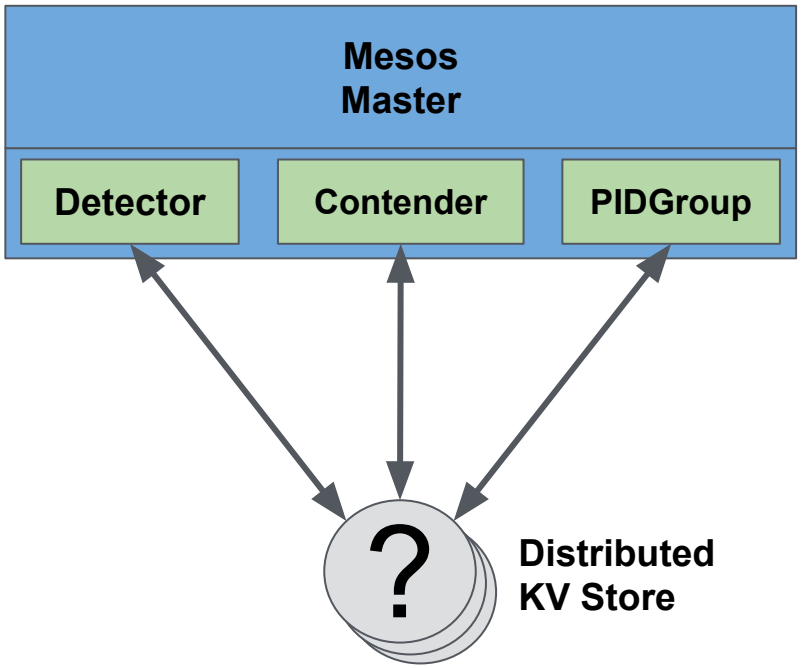
```
bool contend();
```

- Master **Detector** for discovery

```
MasterInfo detect(MasterInfo previous);
```

- PIDGroup for initialization

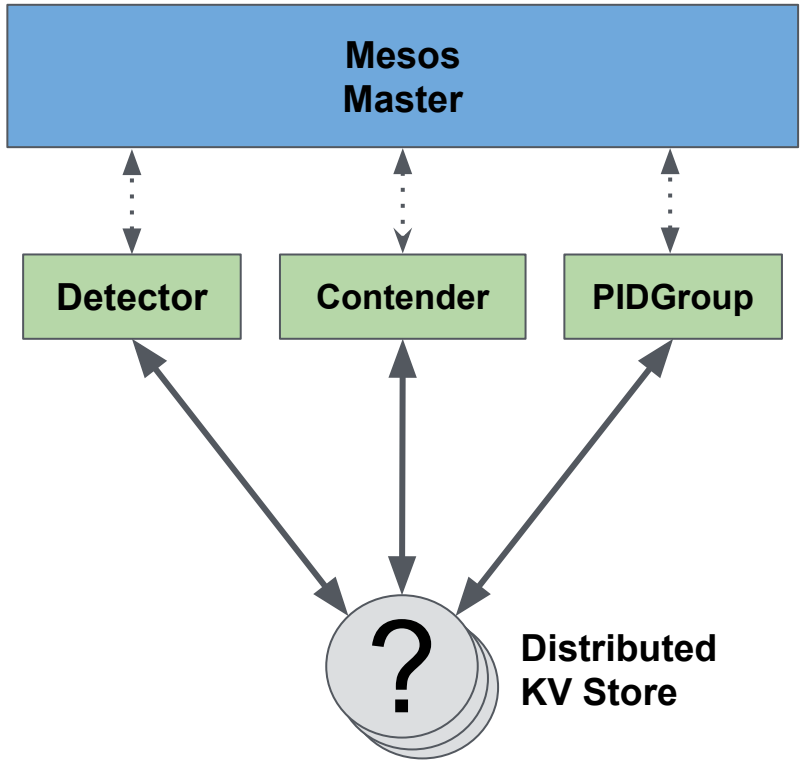
```
void initialize(pid_t pid);
```



A Case for Modularization!

- Already a clear-cut interfaces between:
  - Master and Contender
  - Agent and Detector
  - Framework and Detector

- For new distributed KV store implementation, we just write the module without having to modify Mesos itself!



Let’s Talk about Modules!

Mesos Modules

- Module/Plugin/Extension
- Add/replace a Mesos component
  - Isolators
  - Authenticators
  - …

- Hook modules:
  - **Listen** to interesting events
  - Modify/enhance certain code paths
  - Prepare/enhance task environment
  - ...

How are Modules Used?

- Compiled as shared libraries
  - `E.g., libmesos_network_overlay.so`

- Specified when launching Master/Agent/Framework

```
mesos-agent.sh <master-parameters>
  --modules=file:///path/to/modules.json
  --isolation=”my_isolator”
```

- Gets loaded during initialization
  - E.g., the **`”my_isolator”`** isolator will be loaded into the Agent to provide task isolation

Community Modules

I just wrote a Mesos module that provides a really cute feature.

How do I make it useful for others!

Modules are Tricky!

- Developing
- Building
- Testing
- Using
- Hosting

- How can we make it all better for community?

Writing Modules

- Doesn’t require intimate Mesos knowledge
  - Just the details of the subsystem being implemented (e.g., Isolators)
- Familiarity with Mesos model is required
  - E.g., libprocess, events, futures and promises, etc.
- Closely tied with Mesos version
  - To ensure mutual compatibility

Building Modules: Issues

- Build Mesos first!
  - Install all Mesos dependencies
  - Takes a long time to build
  - Version dependencies

Building Modules: Good News!

- Starting Mesos 1.0 release, pre-compiled Mesos deb/rpm packages contain everything needed to build modules

Testing Modules

I just wrote a simple Mesos module that provide a cute feature and I know how to build it!

Can I write unit tests for it?

Testing Modules

- Key questions:
  - How to get good test coverage?
  - How can we solicit help from community?

- Good news!
  - Efforts on the way to create a “libmesos_test” library that can be used to create/run gmock style tests just like with Mesos itself.

Community-Driven Modules

How do we, as a community, make third-party modules available for general consumption?

While making sure the developers and consumers can seamlessly test/integrate into their environments!

Community Modules: Proposal

- A central registry that contains pointers:
  - E.g., github.com/mesos/modules
  - Each module (or a set of related modules) in its own repository

- Make Mesos version-specific binary rpm/deb modules available
  - E.g., `lib_my_module_<module-version>_<mesos_version>.so`

Module CI: Coming Soon!

- Builds binary packages for every registered module
  - Across a given set of Mesos versions
  - Work-in-progress!
- Automatic build/testing for upcoming Mesos release
  - Catch incompatibilities sooner!
- Run tests!

Let’s take a look at Etcd!

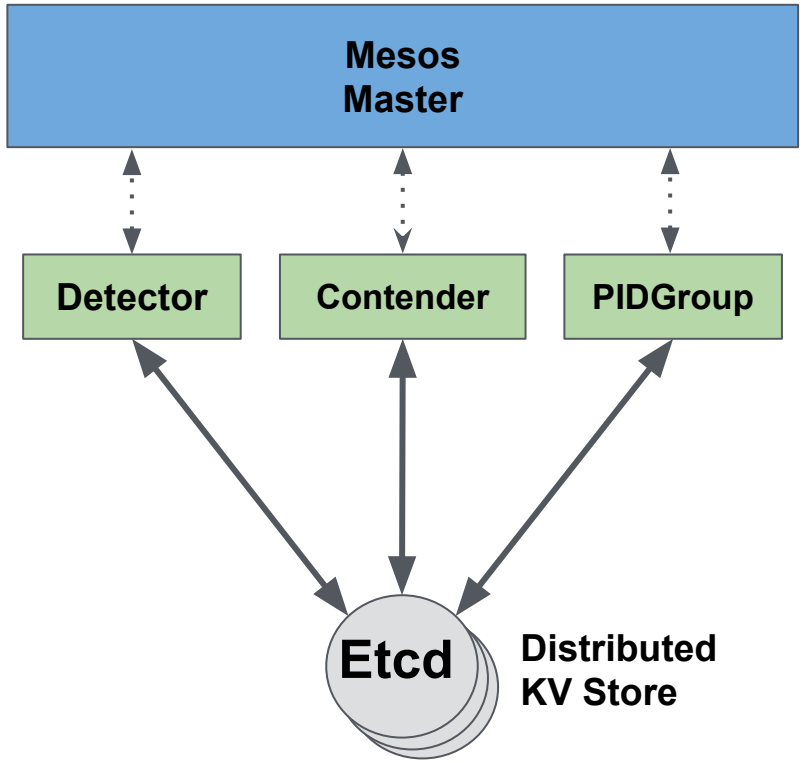
Etcd: A Distributed KV Store

- HTTP API (no language bindings)
- May already exist in your environments

Etcd in a Mesos Cluster

- Create Etcd-specific modules for:
  - Master detector
  - Master Contender
  - PIDGroup

- No need to modify/rebuild Mesos



Again, it's all about having options!

- **Chocolate**
- **Strawberry**
- **Vanilla**
- ...

Again, it's all about having options!

- ~~Chocolate~~
  Zookeeper
- ~~Strawberry~~
  Etcd
- ~~Vanilla~~
  Consul
- ...

Demo!

Module CI:
A Glimpse!

Acknowledgments!

- Shuai Lin
- Cody Maloney
- Benjamin Hindman
- Joseph Wu

Thanks!

- Etcd modules:
  - https://github.com/guoger/mesos-etcd-module/tree/1.1.x
  - https://github.com/guoger/mesos/tree/pid-group-on-1.1.x