Apache Mesos:
A Fault-Tolerant Cluster Computing Framework

Lyubomir Ivanov
s141736

Antonios Spyropoulos
s141707

Dimitrios Danampasis
s141732

What is Mesos?

A platform for sharing commodity clusters between multiple diverse cluster computing frameworks (e.g. Hadoop and MPI)

What can Mesos do?

Mesos **abstracts** CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily and effectively be built and executed.

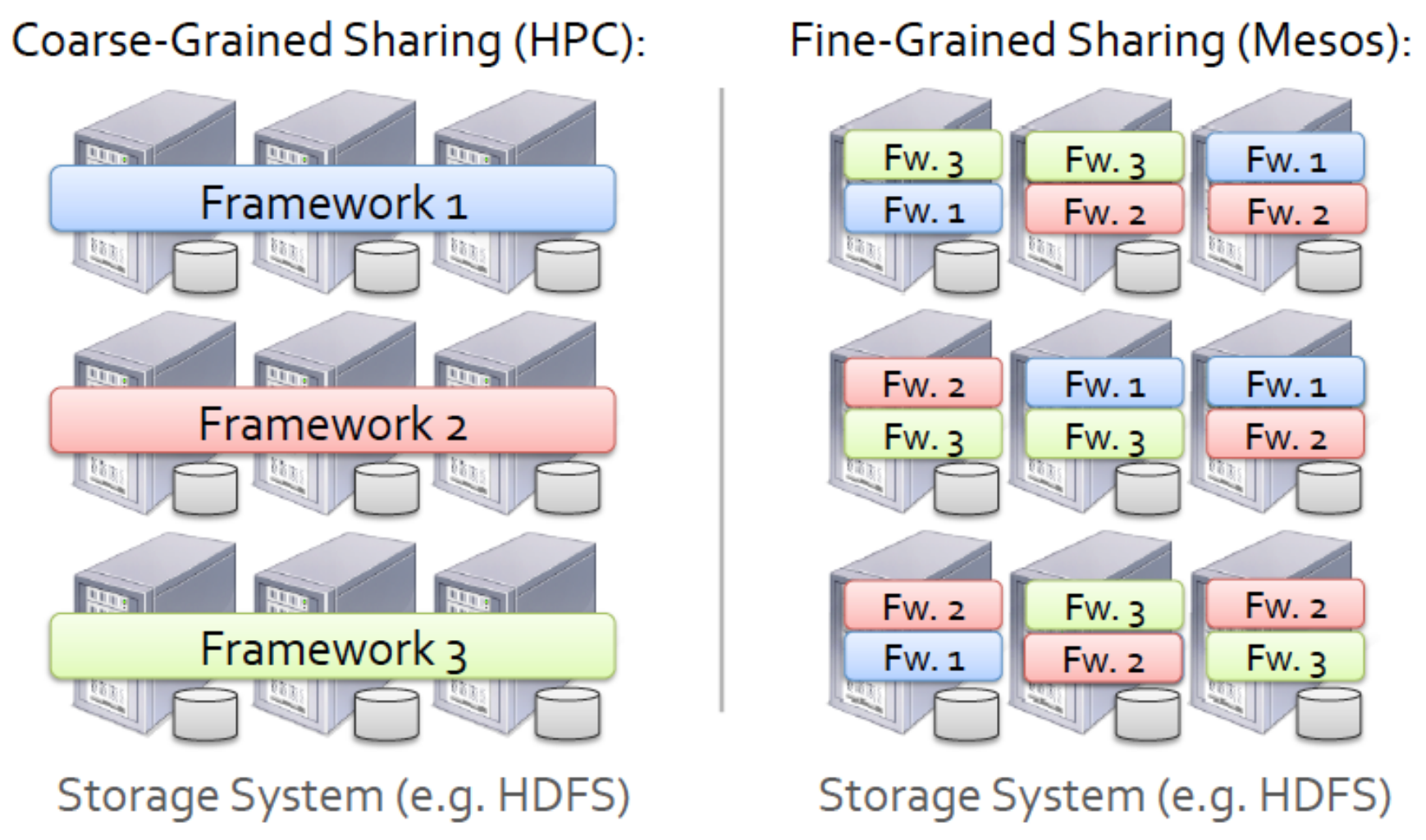

Coarse-Grained Sharing (HPC):
Fine-Grained Sharing (Mesos):
Framework 1
Framework 2
Framework 3
Fw. 1
Fw. 2
Fw. 3
Storage System (e.g. HDFS)
Storage System (e.g. HDFS)

Figure 1. Resource sharing

Architecture

Mesos consists of a **master** daemon that manages **slave** daemons running on each cluster node, and Mesos applications (also called **frameworks**) that run **tasks** on these slaves.

Master

The master enables fine-grained sharing of resources (cpu, ram, …) across applications by making them **resource offers**. Each resource offer contains a list of:
<slave ID, resource1: amount1, resource2, amount2, …>

Frameworks

A framework running on top of Mesos consists of two components: a **scheduler** that registers with the master to be offered resources, and an **executor** process that is launched on **slave** nodes to run the framework's **tasks**. Figure 3 illustrates a resource offer example.

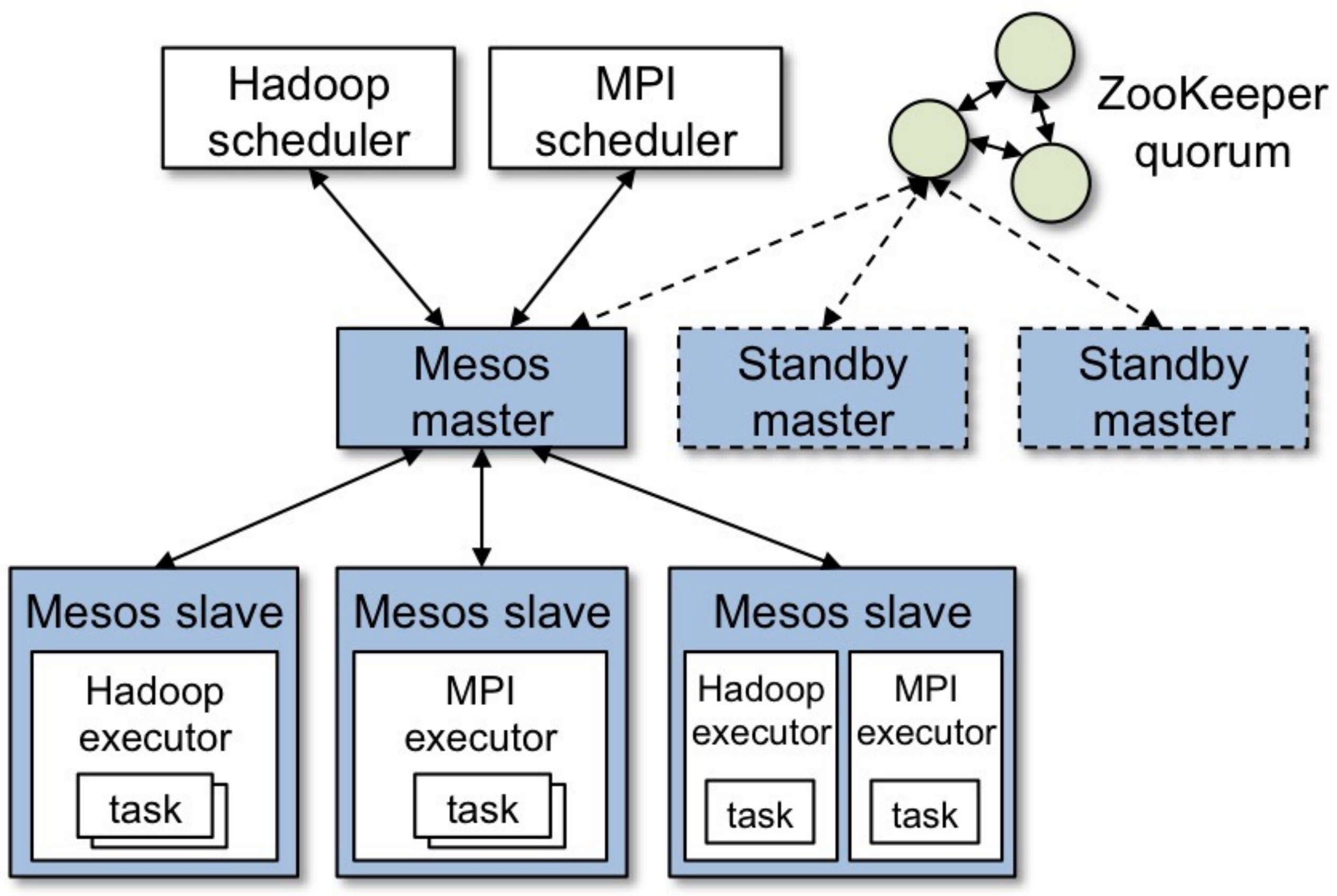


Figure 2. Mesos Architecture

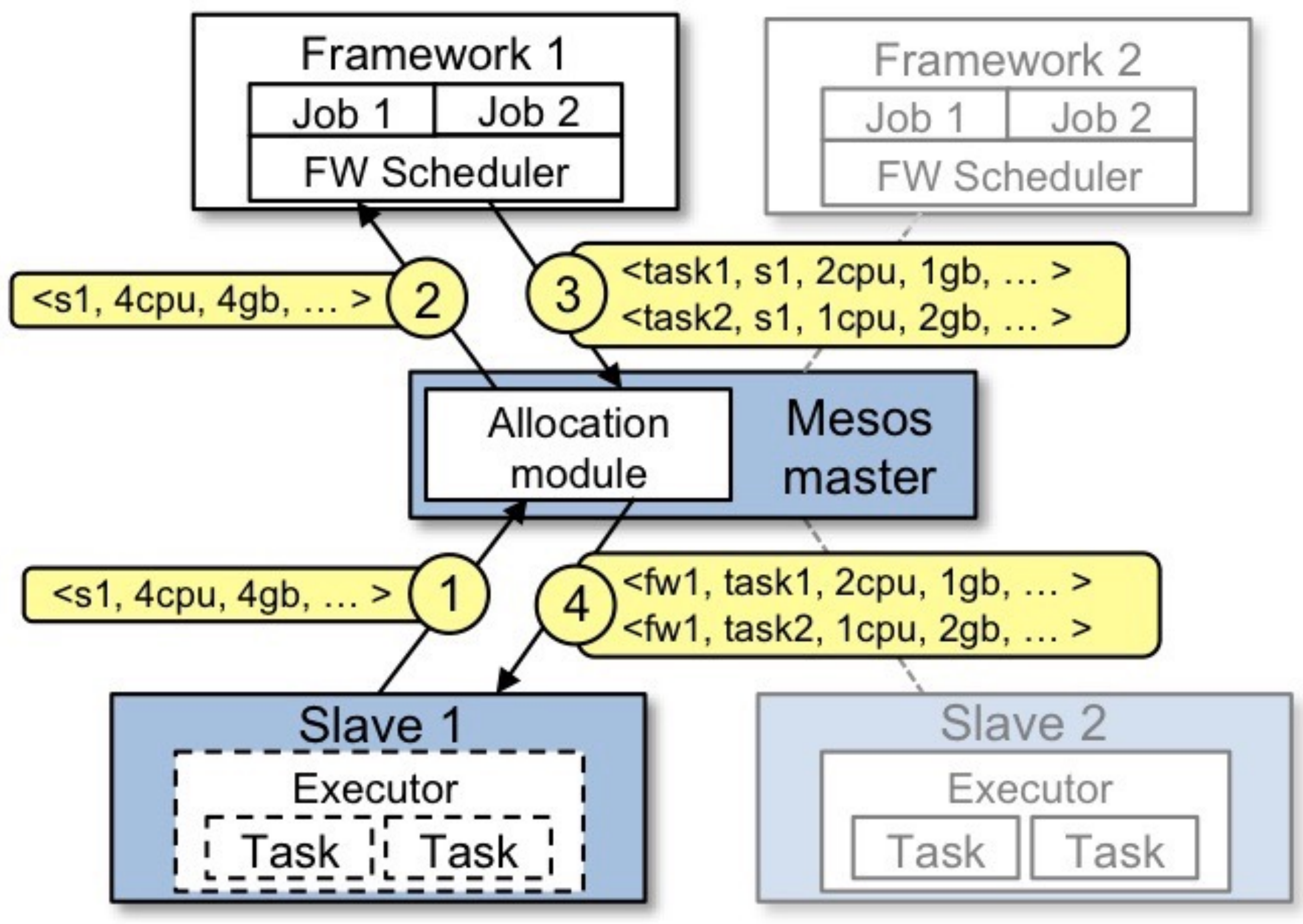


Figure 3. Resource offer example

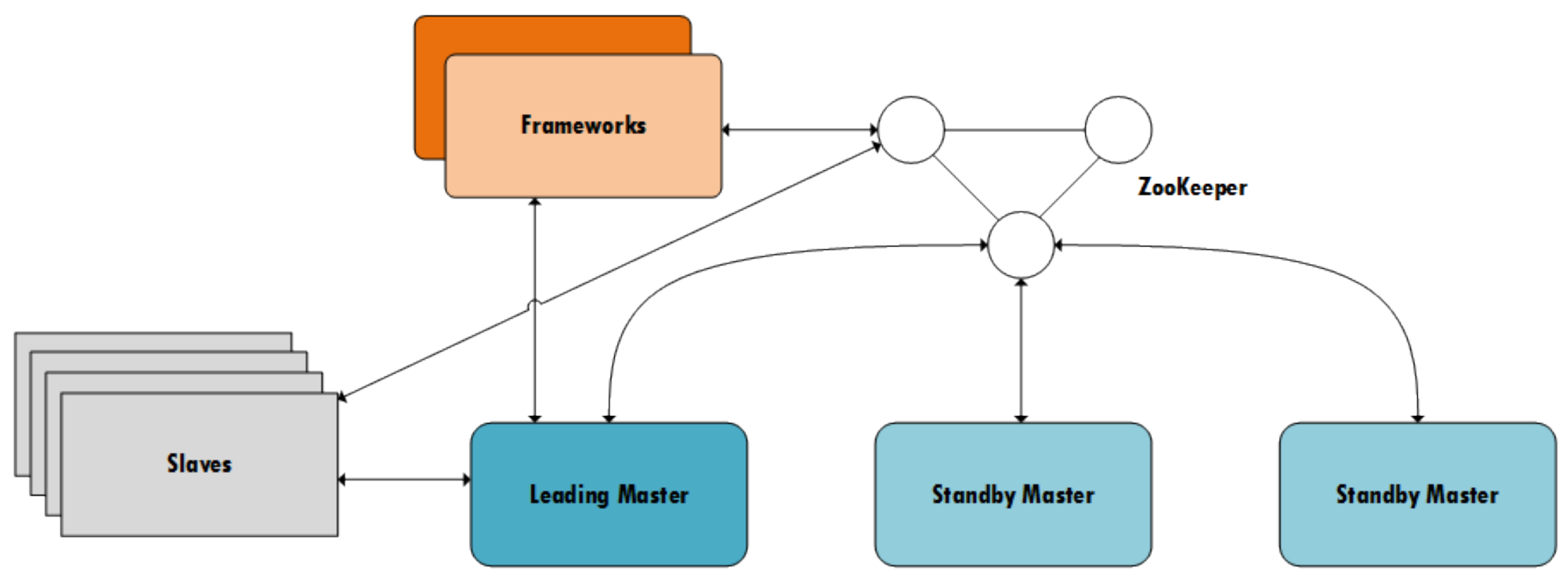


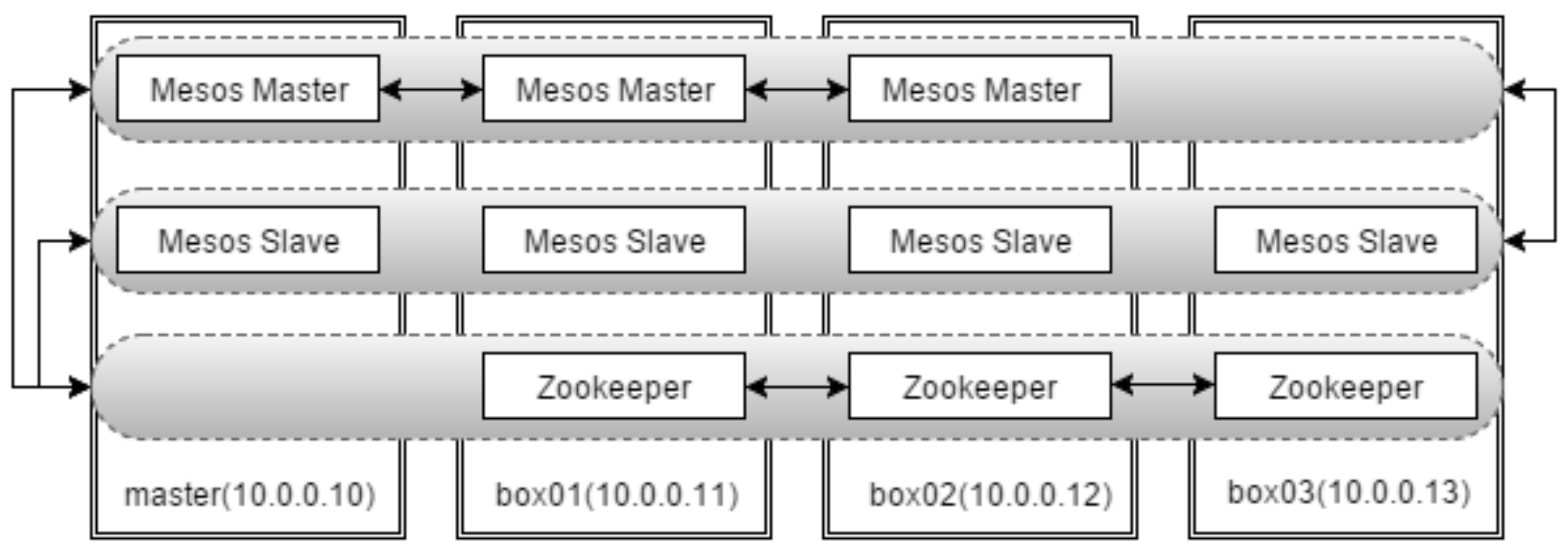
Figure 4. Mesos running in high availability

Implementation

4 VMs with: 2 processors, 512 MB RAM

Total resources for Mesos slaves are offering to the master would be approximately 8 processors and 1 GB of RAM in total.

To simulate fault types, we are going to:
- forcefully power off a virtual machine
- forcefully kill some of the processes in the environment
- forcefully re-segment the network



Fault-Tolerance

Mesos deals with:
- machine failures
- software failures

Components that are resilient to these failures:
- master
- slave
- framework

Detection & Localisation

- Health Checks
- Registry

Handling

- If the elected Master fails, **ZooKeeper** elects new leading master from the standby-s (Figure 4).
- If a slave is separated from ZooKeeper (network segmentation), it ignores elected master messages until reconnected.

Registry

- Adds a minimal amount of persistent state to the master.
- Contains a list with the registered slaves.

Slave Recovery

A slave process can be:
- restarted
- reconnected

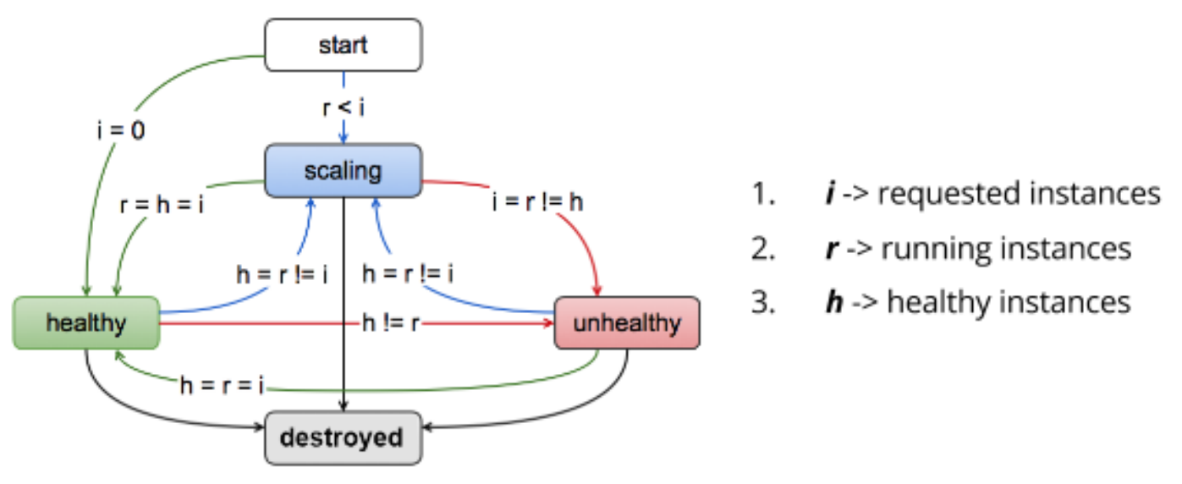
based on checkpoints that are stored in the registry.

Checkpointing

Slave checkpoints store information such as:
- Task Info
- Executor Info
- Status Updates

Health Checks

- Determine if a task is healthy



1. ***i*** -> requested instances
2. ***r*** -> running instances
3. ***h*** -> healthy instances

Figure 5.Health checks