Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center

Mesos is a platform for sharing clusters between multiple and diverse cluster computing framework. Mesos shares resources in a fine-grained manner, allowing frameworks to achieve data locality by taking turns reading data stored on each machine. To support the sophisticated schedulers of today's frameworks, Mesos introduces a distributed two-level scheduling mechanism called resource offers. Mesos takes an approach of delegating control over scheduling to the frameworks. Resource offers encapsulates a bundle of resources that a framework can allocate on a cluster node to run tasks. Mesos decides how many resources to offer each framework, based on an organizational policy such as fair sharing, while frameworks decide which resources to accept and which tasks to run on them.

Each framework running on 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. While the master determines how many resources to offer to each framework, the frameworks' schedulers select which of the offered resources to use. When a framework accepts offered resources, it passes Mesos a description of the tasks it wants to launch on them. To achieve the goal of allowing multiple tasks to run on the same set of slave nodes, Mesos uses what it calls isolation modules to allow a number of application and process isolation mechanisms to be used for running those tasks. It's probably no surprise that although an isolation module could be written to use virtual machines for isolation, the current supported modules are for containers. Mesos delegates allocation decisions to a pluggable allocation module, so that organizations can tailor allocation to their needs.

Since all the frameworks depend on the Mesos master, it is critical to make the master fault-tolerant. This is achieved by designing master to be soft state. A new master can completely reconstruct its internal state from information held by the slaves and the framework schedulers. In particular, the master's only state is the list of active slaves, active frameworks, and running tasks. This information is sufficient to compute how many resources each framework is using and run the allocation policy. To deal with scheduler failures, Mesos allows a framework to register multiple schedulers such that when one fails, another one is notified by the Mesos master to take over. Frameworks must use their own mechanisms to share state between their schedulers.

Mesos was evaluated through a series of experiments on the Amazon Elastic Compute Cloud (EC2). Authors begin with a macrobenchmark that evaluates how the system shares resources between four workloads, and go on to present a series of smaller experiments designed to evaluate overhead, decentralized scheduling, specialized framework (Spark), scalability, and failure recovery.

Mesos is built around two design elements: a fine grained sharing model at the level of tasks, and a distributed scheduling mechanism called resource offers that delegates scheduling decisions to the frameworks. Together, these elements let Mesos achieve high utilization, respond quickly to workload changes, and cater to diverse frameworks while remaining scalable and robust. Mesos enables the development of specialized frameworks providing major performance gains, such as Spark, and Mesos's simple design allows the system to be fault tolerant and to scale to 50,000 nodes.