Optimizing CMS Build Infrastructure using Docker and Mesos

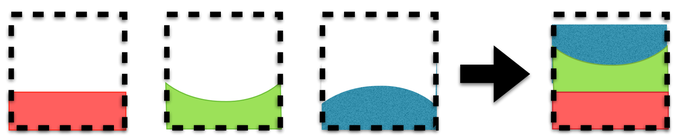
Giulio Eulisse[1], Shahzad Muzaffar[1], David Abdurachmanov[1], Peter Elmer[2], David Mendez[3], Alessandro Degano[4]
[1] FNAL (USA), [2] Princeton University (USA), [3] Universidad de los Andes (Colombia), [4] Università di Torino (Italy)

CMS Build Infrastructure

CMS Build Infrastructure is responsible for release integration, building, testing and deployment of the CMS Offline Software. In busy periods, it needs to handle up to 40 releases per day, on a variety of different platforms and compiler setups.
In order to do so we rely on a cluster of roughly 250 cores, running on Scientific Linux 6, composed of virtual machines running in the CERN OpenStack setup. To drive the whole process we use Jenkins, https://jenkins-ci.org, a commonly used continuous integration system which we use as a glorified batch system for build jobs. The code repository itself and the reports on what is happening are hosted on Github, a well known code / webpage hosting provider. The jobs being executed are extremely non homogeneous, ranging from small batch shell script which generate static web pages, longer jobs performing release builds and validation.
In particular, we use Elasticsearch to provide live feed of what is happening in the build infrastructure and log stash to analyze build machine / web server logs.

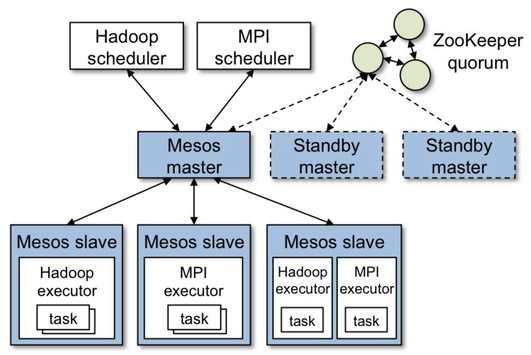
A more dynamic partitioning

While virtual machines provide a nice way to simplify cluster management, they are not necessarily the correct granularity at which a computing system works. In particular our loads and payloads have quite different weights and load distribution over time: for example there is no point in reserving a 24 core machine for a simple cleanup script. On the other hand we do not want to have too small virtual machines, because that makes them useless for larger tasks and increase maintenance due to a larger number of entities. What we want to is make sure we can run any task dynamically on large machines, including long running services.



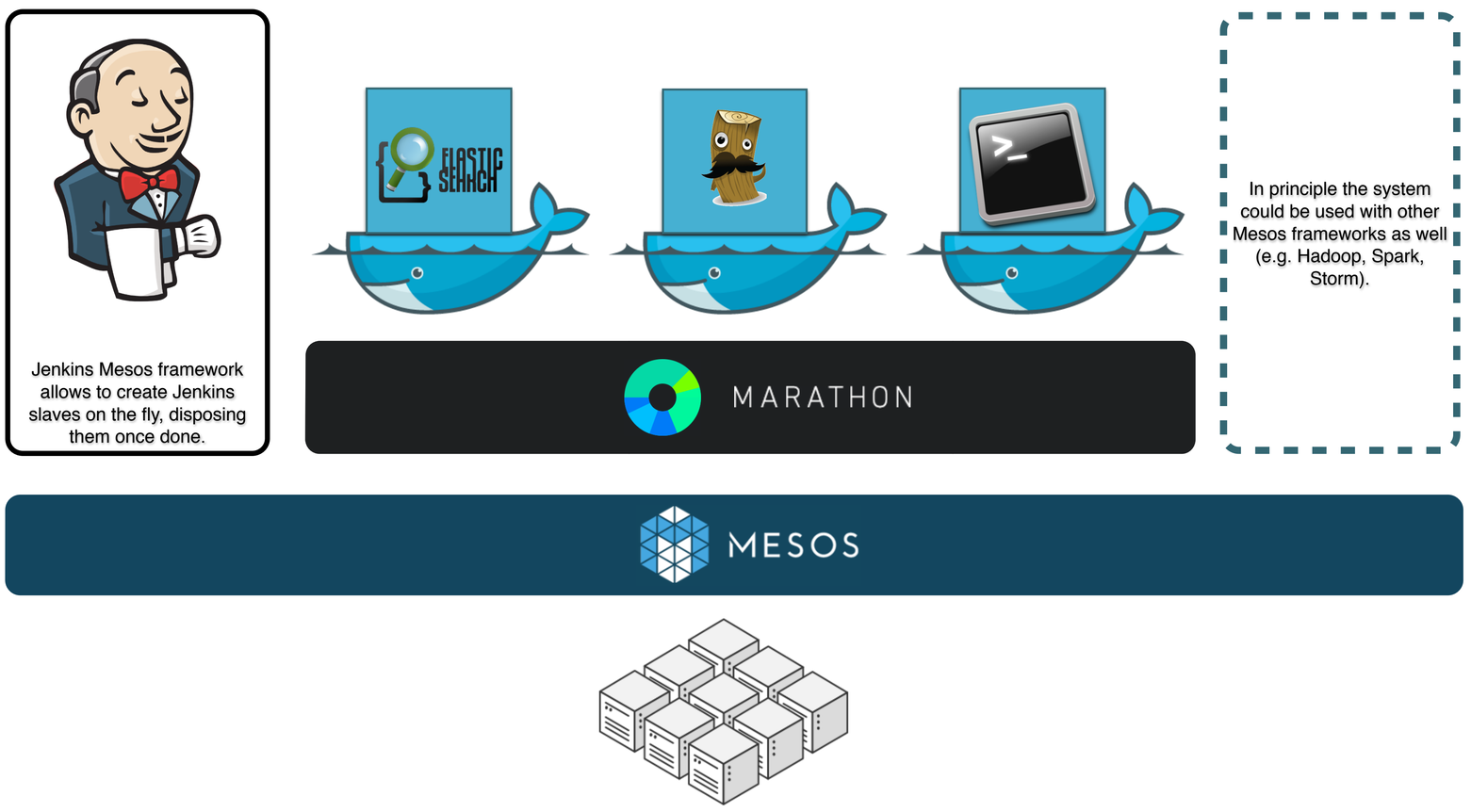
Of course, this is not a novel problem and big players in industry have the exact same need, we therefore looked at what was available there. In particular we looked at how things are done by Google and to an OpenSource reimplementation of their cluster application framework, Mesos.

Apache Mesos



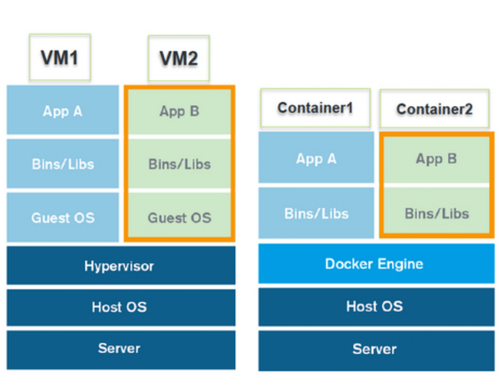
Apache Mesos is a cluster management system initially developed at UC Berkeley and then moved under the Apache Foundation umbrella. It has been adopted by several large software companies, e.g. Twitter. It provides a framework to handle the scheduling of jobs and long running services on a cluster of undifferentiated machines, in a high availability setup by using Apache Zookeeper. Among popular Mesos frameworks we find:

- Mesosphere's Marathon
- the "Mesos Cloud" plugin for Jenkins
- a number of integrations for common "Big Data" heavyweights like Hadoop or Spark.



Docker

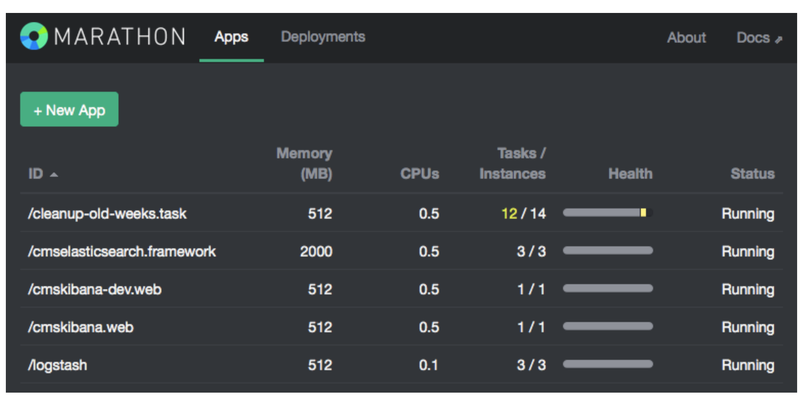
Containers are becoming a widely available feature of modern OSes. They allow processes to run in complete isolation from the rest of the system. They are middle ground between processes and virtual machines offering the isolation advantages of the latter and the ease of management and performance of the former. **Docker** is an open platform which simplifies packaging and distribution of containerized applications. Mesos has native support for Docker containers allowing us to easily deploy containerized jobs on our build infrastructure. This has a two fold advantage:



- the build jobs and services we run on our cluster do not interfere with each other and with the rest of the build infrastructure, simplifying maintenance,
- it allows us to separate updates of the build infrastructure from the job environment, e.g. we can run SLC5 jobs on a SLC6 cluster.

Mesosphere Marathon

Marathon is a PaaS to run long running services on top of a Mesos cluster. It allows to spawn a desired number of processes either running natively or inside a Docker container. It makes sure the services are scheduled on the Mesos cluster given the available resources and the placement constraints (for example access to a specific disk volume). Marathon also monitors the health of the processes and respawns when they die. Configuration and execution of processes can be done either via a web interface or via a REST API. We use the API to generate frontend proxy configuration. Apart from ancillary tasks, in the CMS build infrastructure we use Marathon to spawn Elasticsearch or Kibana instances.



Try it at home

One of the advantages of using Docker is the fact that not only we can try each component in isolation, but we can run newer versions of the software on a restricted setup.
For example you can try out Marathon using:

```
docker run --net=host -it cmssw/zookeeper &
docker run --net=host -it cmssw/mesos-master &
docker run --net=host -it cmssw/marathon &
docker run --net=host \
           -v /var/run/docker.sock:/var/run/docker.sock \
           -v /usr/local/bin/docker:/usr/bin/docker \
           -v /sys:/sys \
           -it \
           cmssw/mesos-slave
```

and then connecting to port 8080 on your own laptop. Using different Zookeeper namespace it's possible to run separate instances of the build infrastructure using the same Mesos cluster or even have a development cluster running on top of the production one. More Docker containers as used by CMS can be found at https://github.com/cms-sw/cms-docker

Acknowledgments

This work was partially supported by the U.S.A. Department of Energy. All trademarks are property of their respective owners and all the non original artwork has been used under the "Fair Use" clause.