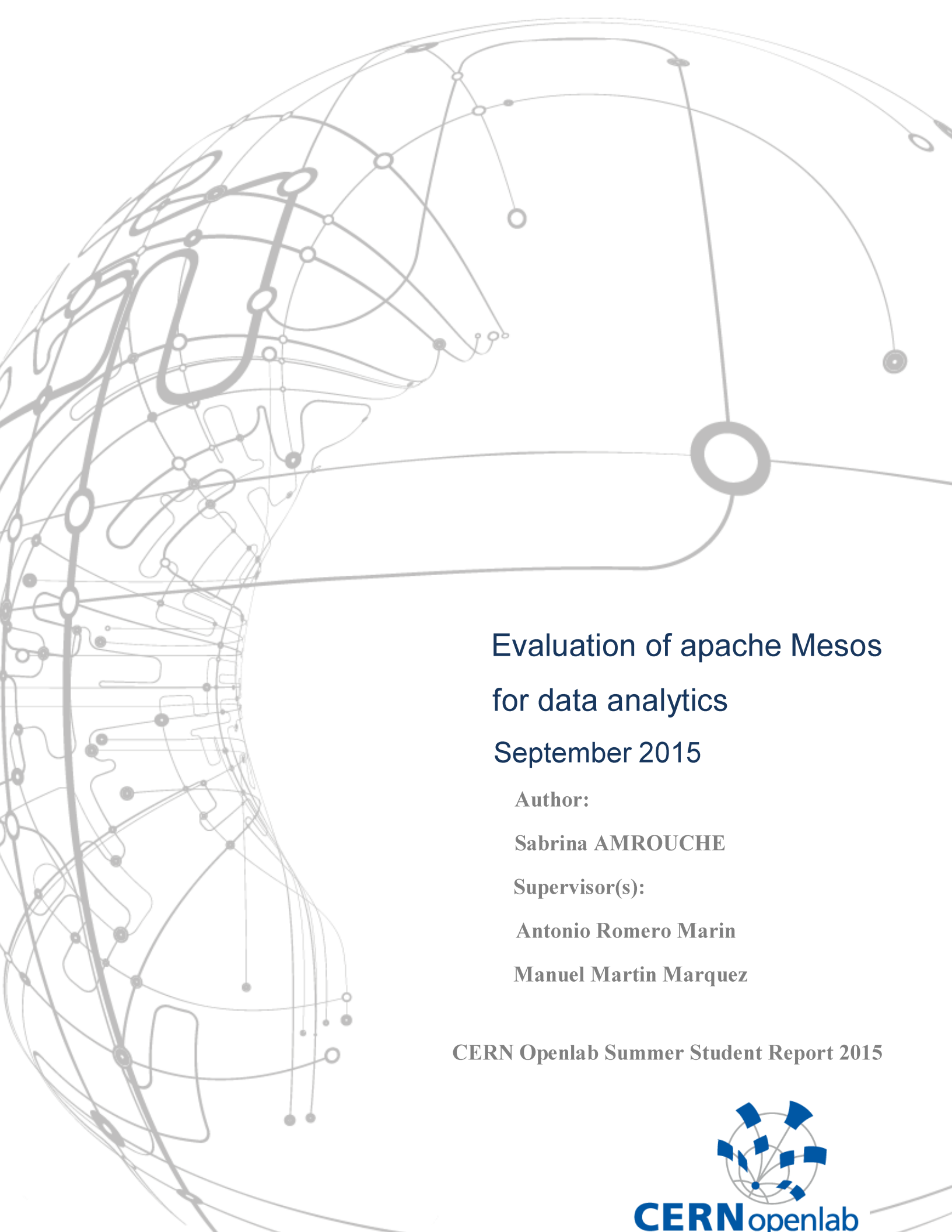

Evaluation of apache Mesos for data analytics

September 2015

Author:

Sabrina AMROUCHE

Supervisor(s):

Antonio Romero Marin

Manuel Martin Marquez

CERN Openlab Summer Student Report 2015

CERN openlab

Project Specification

This project involves the following steps

1. Install, configure and run Apache Mesos on a development cluster and extend it to a stable cluster
2. Test and evaluate data analysis frameworks such as Hadoop and Spark
3. Test and evaluate containers features such as Docker and Marathon
4. Evaluate the stability of Apache Mesos through various benchmarking.

The figure 1 describes the general infrastructure of the Apache Mesos cluster. The figure 2 illustrates the different frameworks and applications running on the top of the cluster manager Apache Mesos.

The aim of the project is to evaluate Apache Mesos in Data analysis as a service perspective.

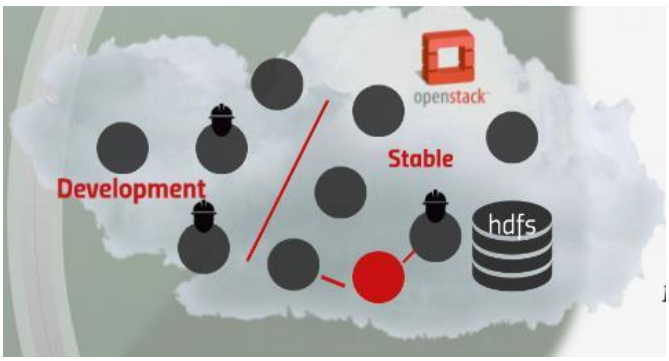

Figure 1 Cluster to build

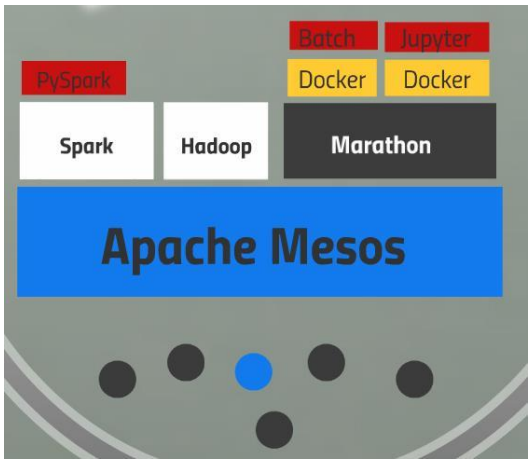

Figure 2 Different applications to run on Mesos

Abstract

Apache Mesos is a popular open source cluster manager for data centers and cloud environments, aimed at providing efficient resource isolation and sharing across distributed applications. Apache Mesos is used in multiple domains as it provides frameworks to run distributed applications for Big Data processing such as Hadoop, Spark and Storm, for batch scheduling like Chronos, data storage, PaaS and more.

This report is about the evaluation of Apache Mesos in the context of data analytics and distributed applications. The evaluation consists mainly of the installation and configuration of an Apache Mesos cluster as well as the analysis of the performance and management of resources in a data analytics context.

Table of Contents

Project Specification....2

Abstract....3

1 Introduction....5

2 Apache Mesos....5

3 Spark for data analysis....7

3.1 Forest Cover type dataset....8

3.2 HIGGS dataset....8

3.3 Monitoring CPU and memory....9

4 Containers features for data analysis....10

4.1 Docker....10

4.2 Marathon....11

5 Technical deployment....12

5.1 Apache Mesos:....12

5.2 Hadoop....12

5.3 Spark....12

5.4 Docker and Marathon....13

6 Conclusion....13

1 Introduction

At CERN a lot of data analytics is being performed every single day. Beyond data analytics for physics discoveries, scientists are analysing the Meta-data to improve the systems: monitoring and logging data to predict failure and prevent them, predict popularity of a dataset to make it more accessible and available, parallelize learning algorithm across a cluster to gain accuracy and efficiency and finally the test of new technologies to speed up data analytics like the use of containers. To make available such services with performant and global tools defines Data Analytics As Service.

Performing data analysis can be done with different types: Batch processing, Stream processing, user-facing services, graph processing and complex event processing. To deal with all of these types scientist have very different tastes on the right tool to use. To give every scientist the possibility to work with his favourite tool one more layer is added to manage all the data analysis tools. This report is a brief evaluation of Apache Mesos, a popular cluster manager for data analytics.

2 Apache Mesos

Mesos is described as a distributed systems kernel for a cluster. It is an open source project developed at the University of California, Berkeley and presented first in 2009.

The Mesos kernel runs on all of the cluster nodes providing various types of applications named frameworks as Spark, Hadoop, MPI, Storm... for data analysis, Marathon, Aurora as long running services and batch scheduling and data storage tools like Cronos, Jenkins, Cassandra and Elastic Search (see Figure 1).

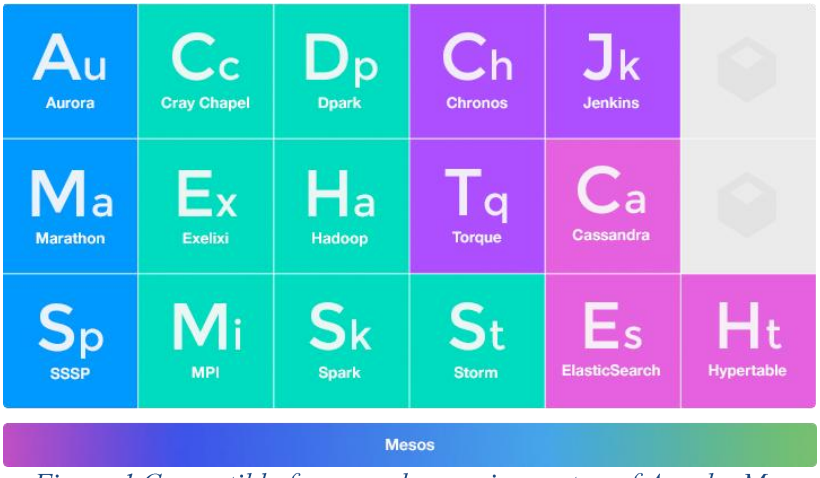

Figure 1 Compatible frameworks running on top of Apache Mesos

Mesos aims to give frameworks a common interface for accessing the cluster resources. Its approach is to delegate control over scheduling to the frameworks by creating resource offers.

*M*esos aims to give frameworks a common interface for accessing the cluster resources. Its approach is to delegate control over scheduling to the frameworks by creating resource offers.

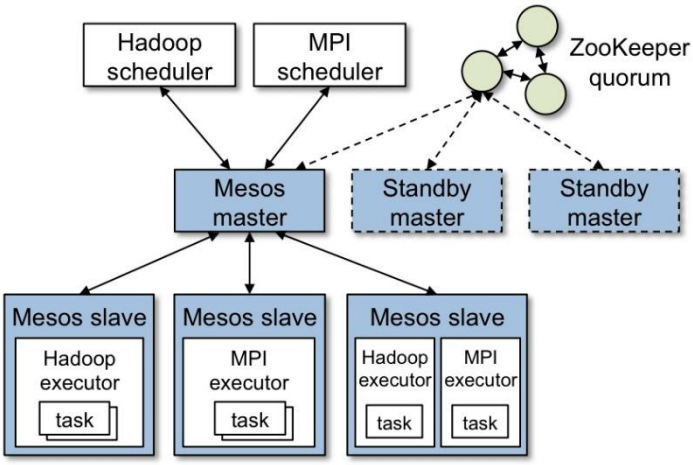

Figure 2 Resources offers mechanism

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**.** Organizational policies can be defined via a pluggable allocation module.
A framework running on Mesos consists of a *scheduler* and an *executor* as shown in **figure 2.** More than the frameworks already running on the top of Mesos, the development of new frameworks (scheduler and executor) is described in the App/Framework development guide .

Mesos Apache Project Features

As mentioned in the mesos.apache.org Mesos apache has some appealing features for data analysis jobs as

- Scalability to 10,000s of nodes
- Fault-tolerant replicated master and slaves using Zookeeper
- Support for Docker containers
- Native isolation between tasks with Linux Containers
- Multi-resource scheduling (memory, CPU, disk, and ports)
- Java, Python and C++ APIs for developing new parallel applications
- Web UI for viewing cluster state

Adding that apache Mesos provides a Json object containing all sort of metrics to follow in real time the jobs execution. Monitoring is about resources, system, master, slaves, frameworks, tasks, messages… and also basic alerts can be set.

The figure 3 describes an example of how Mesos schedules frameworks to run jobs

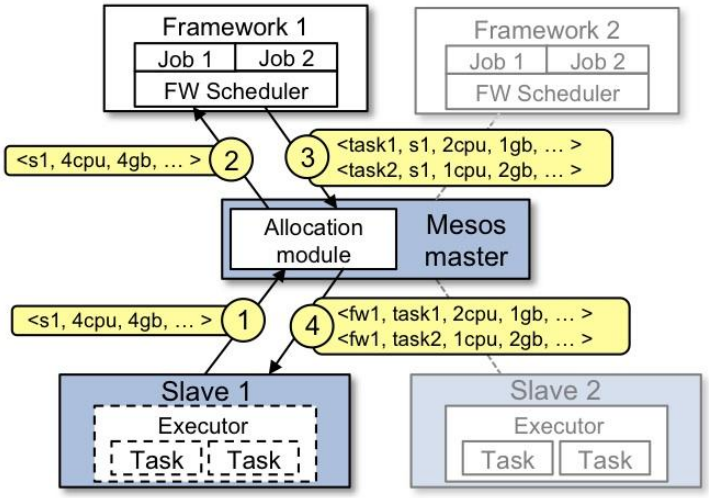

Figure 3 Scheduling an example job

(1) The slaves report to the master all available resources they have. By invoking the allocation policy module, **(2)** the master sends a resource offer describing what is available on slave 1 to framework 1. **(3)** The framework's scheduler replies to the master with information about two tasks to run on the slave, using <2 CPUs, 1 GB RAM> for the first task, and <1 CPUs, 2 GB RAM> for the second task. **(4)** Finally, the master sends the tasks to the slave, which allocates appropriate resources to the framework's executor which launches the two tasks

Apache Mesos has built in software for big data processing, batch scheduling, data storage and long running services. In the following sections we will test the use of Spark as a data analysis framework and Marathon as long running services.

3 Spark for data analysis

Spark is one of the most popular engine for large-scale data processing. Spark runs on Hadoop, Mesos, standalone, or in the cloud. It defines very useful build-in libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. We uses the MLlib to run a random forest classification algorithm on various dataset to evaluate the execution of Spark on Mesos.

3.1 Forest Cover type dataset

The classification algorithm aims to predict forest cover type from cartographic variables.

The dataset is formed of 581012 instances (70 MB) with 54 features. We used 50, 100 and 500 trees in a random forest algorithm and recorded the training time of the predictive model.

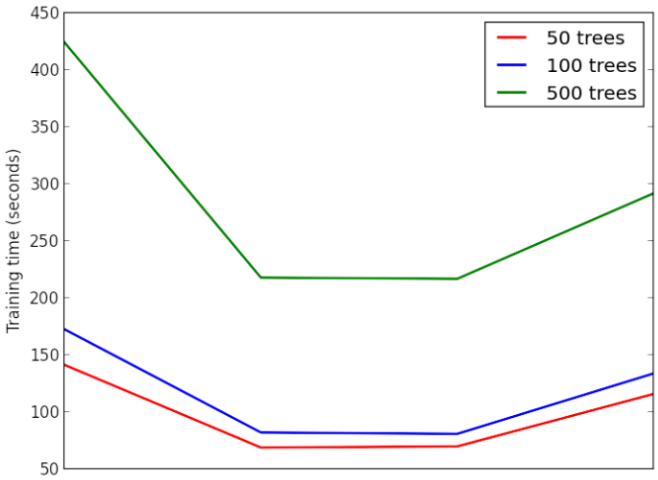

Figure 4 Training time by adding more node with a small dataset

3.2 HIGGS dataset

We used a larger dataset to better evaluate the scaling of the data analysis task. Higgs data describe a classification problem to distinguish between a signal process which produces Higgs bosons and a background process which does not. The dataset contains 11000000 entries (2.5 GB compressed data) with 28 features. We also tested 50 and 500 trees to evaluate the training time.

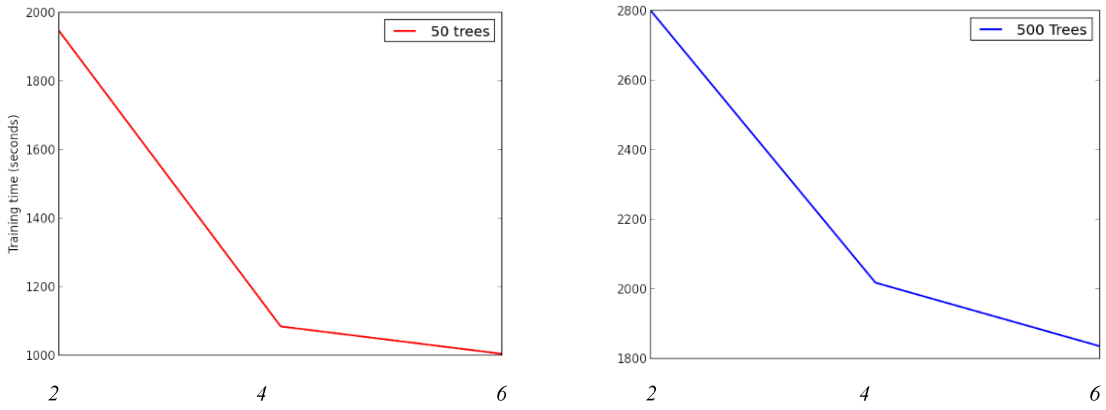

Figure 5 Linear decrease of the training time by adding more nodes on a large dataset

We can clearly see the decrease of the training time by adding more nodes: linear decrease from 2 to 6 nodes.

3.3 Monitoring CPU and memory

We can also follow the CPU and memory usage during the learning process with every dataset. The **figure 6** shows the CPU and memory usage with the forest cover type data. We can see that both the CPU and the memory gradually increase until 8 CPU cores over a total of 24.

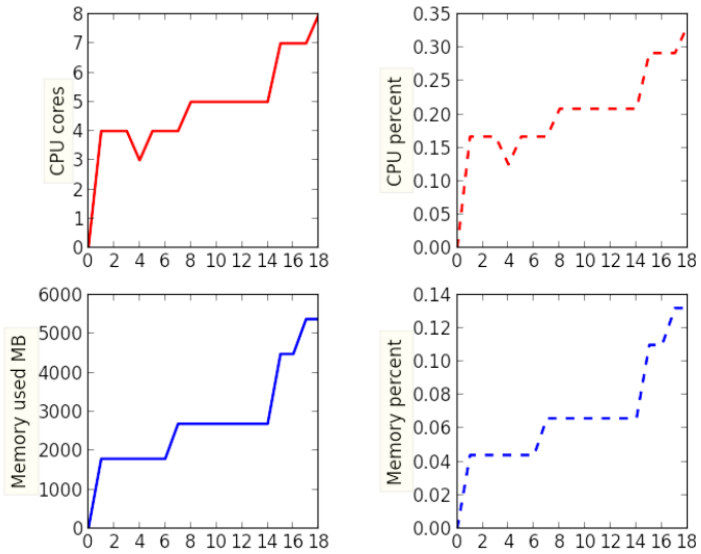

Figure 6 Evolution of the CPU and memory usage with the small dataset

On the HIGGS dataset described by the **figure 7** we can see the much more important usage of the CPU reaching 100% of usage in the learning process. The fluctuations of the usage can be explained by the map and reduce jobs that need to join results at some specific points of the workload and don't require much CPU.

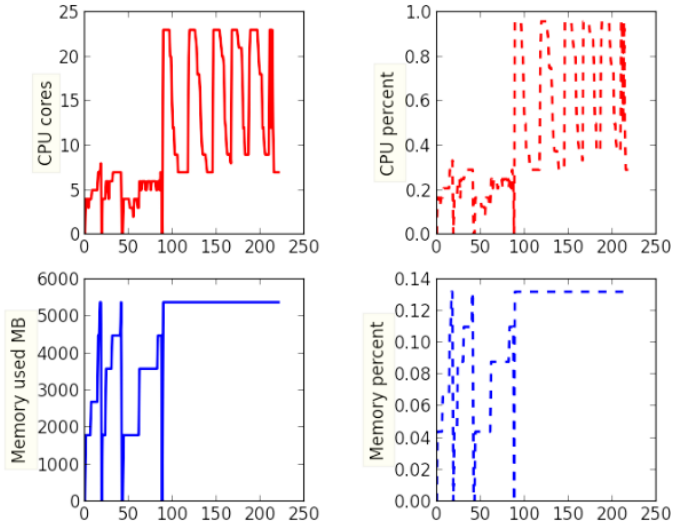

Figure 7 Evolution of the CPU and memory usage with the big dataset

4 Containers features for data analysis

Containers are open platforms for distributed applications for developers and sysadmins. They are mainly used to secure an application by putting it into a “sandbox” or to isolate an application and its data from others customers in a Software as a Service (SaaS).

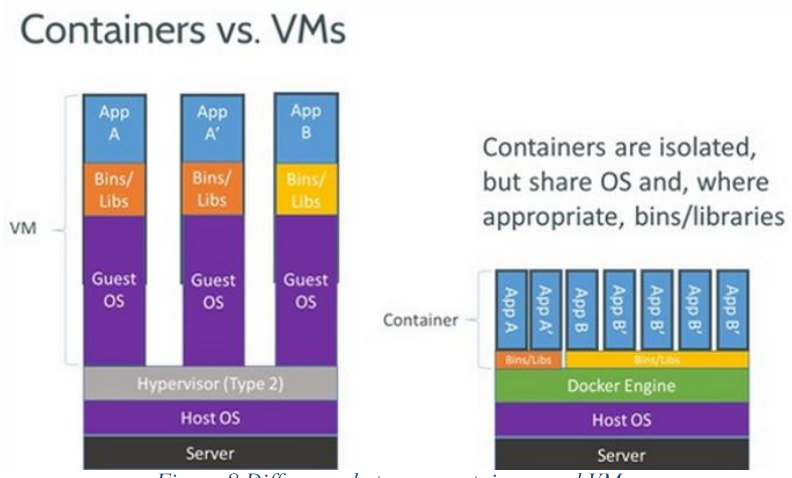

Figure 8 Difference between containers and VM

Containers are an increasingly popular method of separating an application from the operating system and the physical infrastructure.

4.1 Docker

As defined in the Docker official website, Docker is an open platform for building, shipping and running distributed applications. Docker automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux, Mac OS and Windows.

Docker gained in popularity and is now integrated with other tools such as: Ansible, Chef, OpenStack, Puppet, and Salt. Docker is included with RHEL 7.0, OpenShift PaaS, Google Compute Engine (GCE), Deis, and now Amazon Web Services (AWS) Elastic Beanstalk.

4.2 Marathon

Marathon is a private PaaS built on Apache Mesos. It automatically handles hardware or software failures and ensures that an app is “always on”. Instances of various frameworks can be launched and if either of the tasks dies (due to underlying slave crashes or power loss in the cluster), Marathon will re-start a similar instance on another slave. This approach ensures that the exact number of processes are always running on the cluster.

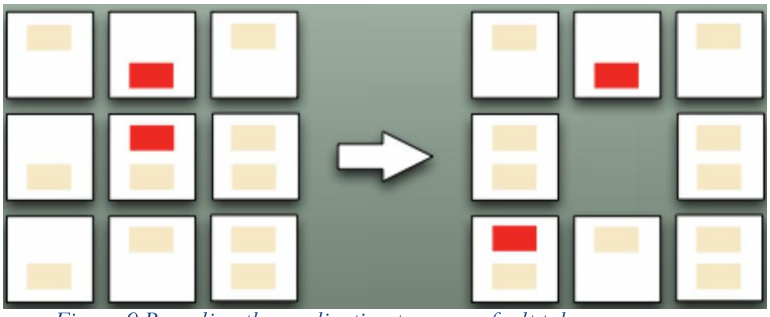

Figure 9 Rescaling the application to ensure fault tolerance

With marathon applications can be easily rescaled over the cluster. Few seconds after scaling the application, new instances are launched on available nodes of the cluster. Marathon offers a Web UI to follow the deployment process and the running of the application. To enable the use of containers such as Docker, a Json object needs to be defined and sent to the running Marathon framework.

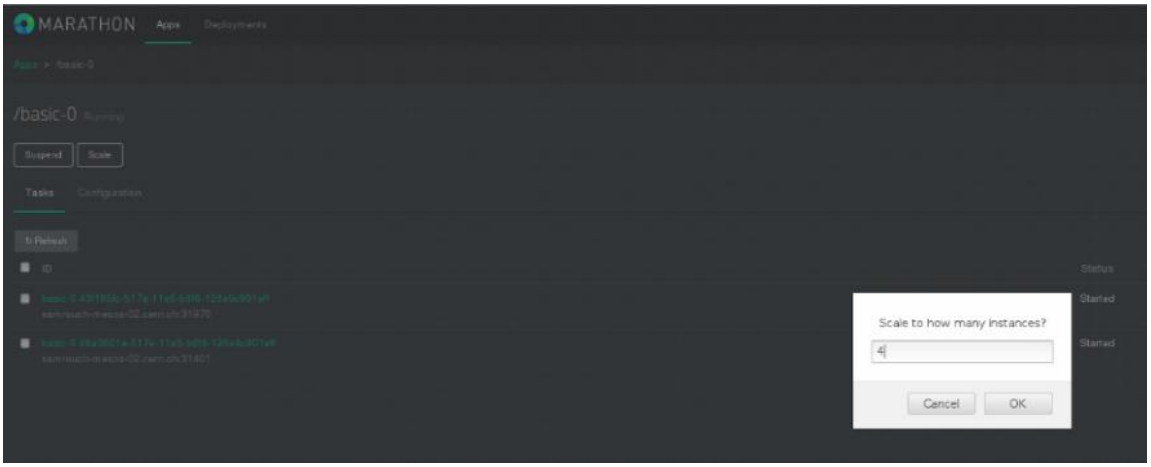

Figure 10 Marathon makes it fast and easy to rescale applications

5 Technical deployment

5.1 Apache Mesos:

Mesos is a recent project very quickly evolving, during the carrying of this study 3 new versions have appeared. For now the most stable version is Mesos 0.26.0 released in December 16, 2015 . This rapid growing makes most documentation outdated and the installation process completely different. Depending on the linux distribution, the installation process is very different. On Ubuntu for example only Mesos dependencies are required before building it. Redhat or Centos require a lot more work with in many cases some issues to solve like the subversion package dependencies. A WANdisco SVN repo must be created to ensure getting the latest version of all packages. To have mesos running the following packages need to be installed/updated along with "Development Tools":

python-devel java-1.7.0-openjdk-devel zlib-devel libcurl-devel openssl-devel cyrus-sasl-devel cyrus-sasl-md5 apr-devel subversion-devel

After successfully installing all packages, simply configure and build Mesos. Start mesos master and slave with **service mesos-master/mesos-slave restart.** It all the process went correctly open the following URL to the web mesos console **http://<master-ip>:5050** to supervise the cluster visually.

An alternative to manually installing packages, dependencies and configuring ,building mesos is to use mesosphere. It has official package repositories which connect directly to the native package management tools of the Linux distribution you are running to install Mesos. The installation process is reduced to a **yum -y install mesos** with the mesos RPM package.

5.2 Hadoop

To install Hadoop on top of mesos the mapred_site, core_site and hdfs_site must be updated with the mesos master address to connect to and the parameters of the file system replicas. Some technical issues are often due to the HDFS not knowing the slaves address/names. It is very important that all the cluster node have access to the same available and updated files. Mesos does not show a specific error on that matter just a standard job stopped/lost at node but mentioned in the mesos-slave.ERROR and /var/log/messages.

5.3 Spark

Spark is a very well documented project. To run it on mesos, the spark-env.sh must contains all necessary variables linking to the mesos cluster manager. To launch jobs Spark will use the mesos manager.

5.4 Docker and Marathon

Mesos supports Docker only after version 0.23.0, in the previous versions it could not create a containerizer. Same goes for marathon which is fully supported with version 0.22.1. The installation of both Docker and marathon on Mesos is very straightforward and it only needs to specify the cluster master.

6 Conclusion

Through the various tests and benchmarking analysis performed we can clearly see the Mesos scalability over the cluster. Mesos is able to dynamically add or remove resources to or from the running jobs. Mesos makes it easy to manage the cluster and thanks to the resources offers mechanism we don't have to worry about the availability of the CPU or the memory.

After the last MesosCon 2015 held in August several technology leaders and bloggers showed interest in Mesos. Early adopters shared their real world experiences using Mesos: stories from the Apple Siri team, Twitter and Verizon. Key attributes that are pointed out to be buzzworthy are: The use of containers, inner support of Spark for data processing, mixed workload capability and the growing community support.

The third point is important to discuss here as many applications are built from a diverse workload mix like demand shifting say from a container hosted API tier to a Spark analytics backend at any moment. As a cluster manager Apache Mesos provides exactly this mixed workload capability that will be a key feature needed in the future.