On demand YARN clusters that have network, compute and storage isolation from each other (Docker Hack day 3)

By Swapnil Daingade, Sarjeet Singh and Mitra Kaseebhotla @ MapR Technologies

1 Introduction

Businesses constantly strive for higher utilization and better returns from their datacenter investments. This is causing businesses to explore consolidating their infrastructure into one giant cluster instead of several siloed ones.

2 Problem

Businesses currently have several Hadoop/YARN clusters that need to be separate for various reasons (security, performance etc). Is it possible to provide YARN clusters on shared infrastructure which have the same security and performance?

3 Solution

A recent project called Apache Myriad (Incubating) attempts to solve this problem. It uses Apache Mesos as the underlying resource scheduler. It currently supports running a single YARN cluster on top of a mesos cluster. Businesses can share resources between YARN and other applications like web servers, spark etc. As active contributors to Myriad we are exploring options to provide multi tenancy and tenant isolation in Myriad. We have tried to use the experimental networking plugin for creating overlay networks for network isolation. With containers, compute isolation comes for free. For storage we have tried to look at multiple options. 1 Providing each cluster their own storage (loopback or raw devices attached to containers) 2. Providing isolation via file system permissions to single shared instance of DFS.

4 Implementation details

4.1 A little bit about YARN

YARN consists of a central daemon called ResourceManager (RM). Applications running on YARN can request resources from the RM in the form of yarn containers (blocks of cpu, mem). Each node in the cluster has a daemon that manages resource on that node called the NodeManager (NM)

4.2 A little bit about Mesos

Mesos is a light weight resource scheduler. Applications running on mesos are called frameworks. Unlike YARN, it is Mesos that makes resource (cpu, mem) offers to frameworks. If a framework can utilize an offer, it accepts the offer and launches a mesos task (a process) on the node on which the resource was offered. The mesos task is not complete until the process terminates.

4.3 A little bit about Apache Myriad (runs YARN on mesos)
In Myriad the RM is launched using a meta framework like Marathon (open source). Marathon accepts an offer from Mesos and launches RM as a mesos tasks. Myriad has code running inside the RM process that registers with Mesos as another framework. We call this code Myriad Scheduler. Note that since RM has now registered as a mesos framework. It can launch NM's after accepting offers from Mesos.

Myriad currently does not support docker containers for RM and NM. It also does not support multi-tenancy or isolation from other non YARN applications (like web servers etc). Following are details of what we tried as part of the hackathon. Mesos is deployed on the physical cluster. RM and NM's are in containers. RM and NM's belonging to the same YARN cluster can talk to each other. Those that belong to different YARN cluster cannot.

4.4Network Configuration and Isolation

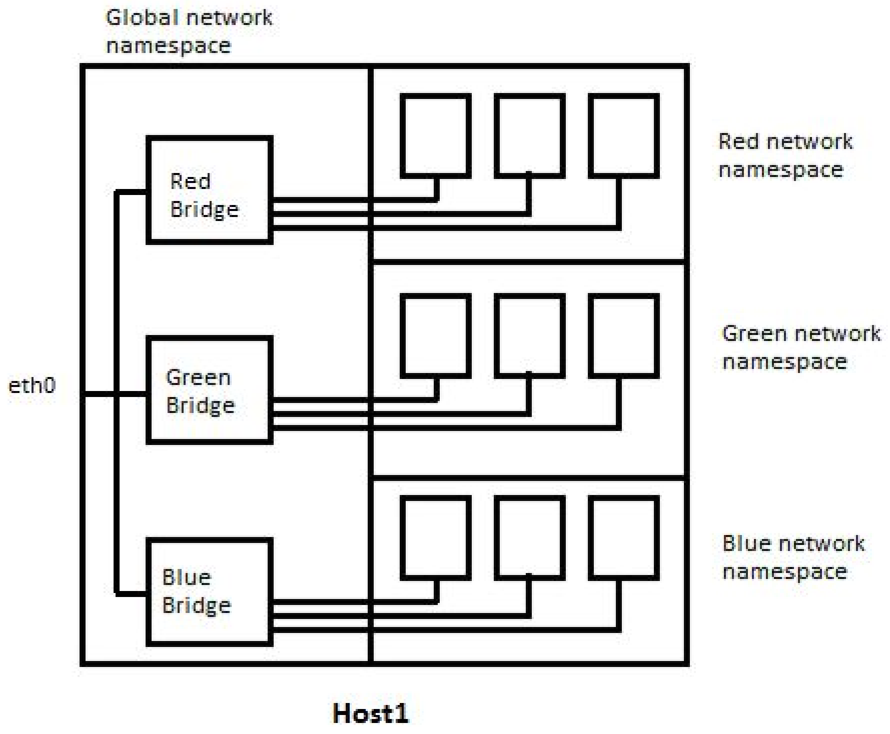

fig.1
Here is how we configured networking on our physical cluster. For demonstration purposes we assume that there are three YARN clusters running on our physical cluster. We create an overlay network per YARN cluster spanning all nodes of the physical cluster. We use docker's overlay network plugin. However membership of an overlay network does not allow a container to talk to the underlay network. This is a problem for us. If RM is to run inside a docker container, it has to have the ability to register with Mesos as a Mesos

framework in order to launch NMs (The Mesos infrastructure is running on the underlay network). Also, we would like to support a configuration where there is only one instance of Distributed File System (HDFS, MapRFS etc) across multiple YARN clusters. The YARN clusters using this single DFS instance are isolated by file system permissions. Hence in order to allow docker containers to access the DFS we created a bridge per cluster on every hosts (just like the docker0 bridge that is present after docker is installed.) Just like on the docker0 bridge, NAT is configured so containers can access the underlay network easily but cannot access containers belonging to another cluster.

This configuration allows RM (which also acts as a Mesos framework) to register with Mesos running on the underlay network. fig 2.(below) shows how this network configuration looks when seen on a 3 node physical cluster. See scripts OnDemandYARNClusters/scripts/docker-create-master.sh and OnDemandYARNClusters/scripts/docker-create-worker.sh to know how this is configured. When RM (acting as a Mesos framework) decides to launch a NM container, it accepts an offer from Mesos and asks mesos to run the OnDemandYARNClusters/scripts/docker_deploy.sh script (present on each physical server). This script takes 3 parameters

1. The Mesos task id of the NM container being launched
2. The overlay network to connect to. (Remember all overlay networks are available on all physical hosts).
3. The docker image to run for the NM.

This script configures storage and networking (connecting the container to the right bridge so it can talk to the underlay network in addition to the overlay network).

We also modified myriad code so RM (acting as a Mesos framework) could generate the command for NM docker container creation. A patch for myriad (phase1 branch) is provided here OnDemandYARNClusters/myriad-docker-hack-day-3.patch
One can also build myriad with this patch by following these steps
git clone https://github.com/sdaingade/myriad.git
git checkout docker-hack-day
./gradlew clean; ./gradlew build

Finally Dockerfiles for MapR distribution of hadoop are also provided under
OnDemandYARNClusters/nm_docker_centos/Dockerfile
OnDemandYARNClusters/rm_docker_centos/Dockerfile
OnDemandYARNClusters/rm_docker_centos/Dockerfile

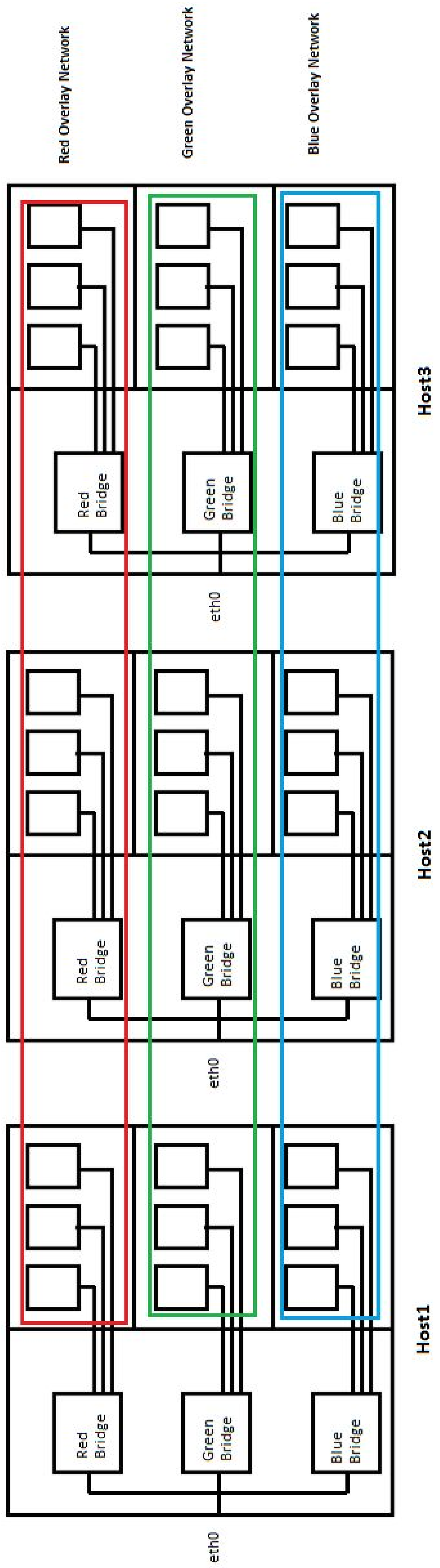
Red Overlay Network
Green Overlay Network
Blue Overlay Network
Host1
Host2
Host3
Red Bridge
Green Bridge
Blue Bridge
eth0

fig2

4.5 Storage

Currently we tried using loopback devices exposed to docker containers for running the DFS i.e. one DFS instance per YARN cluster. In the future we plan to try have a single instance of the DFS shared across multiple YARN clusters and isolation provided using file system permissions.

4.6 Status

Currently were able to execute all the pieces in isolation e.g. networking, RM and NM creation, storage configuration etc. We ran out of time during integration. However we are very confident that with a little more time we’ll be able to get this working end to end.