DISTRIBUTED DATA ANALYSIS WITH DOCKER SWARM USING R

Aishvarya Sivaram
M.Sc Software Systems,
Department of Computing,
Coimbatore Institute of Technology,
Coimbatore, India.

Surya Balamurugan
M.Sc Software Systems,
Department of Computing,
Coimbatore Institute of Technology,
Coimbatore, India.

Mrs.A.Kannammal
Professor,
Department of Computing,
Coimbatore Institute of Technology,
Coimbatore, India

Mrs.P.Aruna
Assistant Professor,
Department of Computing,
Coimbatore Institute of Technology,
Coimbatore, India

Abstract— *The Objective of the exploration work is to propose Docker in R Platform for Distributed Data Analysis with Docker Swarm. Examining substantial volume of dataset is a major test for Data Analysts which can overcome by utilizing Docker Swarm to dissect information in appropriated mold. Docker gives offices to construct, disperse and run applications in a compact, lightweight runtime and bundling instrument, known as Docker Engine. It gives reproducible research strategy to catch the code setting and make it accessible for sometime later. There is no equipment copying required for Docker not at all like Virtual Machines. Factual calculation and investigation of information should be possible by creating R venture inside a Docker compartment which thusly known as Rocker.*

Keywords—Docker, Containers, Virtual Machines, Docker swarm, Rocker, R

I. INTRODUCTION

Virtualization is the technology or approach for utilizing logical version of the physical resources. The goal of the Virtualization is to utilize resources such as storage, processor and network collaboratively to maximum level. Deploying multiple isolated services and resource utilization in a single platform can be achieved by two ways: Containers and Virtual Machines. Hypervisors are used to manage virtual machines (VMs). It runs VMs which have their own operating system using hardware VM support whereas container's system uses basic services provided by underlying operating system to all of the containerized applications using virtual memory support for isolation[4].Container-based virtualization is the technique which uses Linux Containers(LXC) to run a multiple processes, each in their own isolated environment. Therefore the performance of container-based virtualization is better than hypervisor-virtualization because of potentially reduced overheads and thus improves the utilization of data centers. The Combination of LXC with Docker and CoreOS provides fully featured lightweight virtualization for isolating application infrastructure [3]. Therefore Docker is the Operating System level virtualization technology.

Docker is an open source platform to develop ship and run an application in an isolated infrastructure. With the help of this methodology the user can significantly reduce the delay between writing code and running it in a production [6]. Docker will have its own Dockerized application design whereas software architectural designs have been successful for non-Docker applications.Because Docker container provides the ability to package and run an application in an isolated environment. Many different containers can run in a single kernel where the isolation is implemented entirely within that single kernel. Containers are lightweight because there is no hypervisor layer needed in between the isolated task and the host machine kernel to run an application. Containers are an active instance of static images. To launch a container the user must either download the public image or create a new image. Every image consists of one or more file system layers. Each layer states the build step to create an image. The file which contains source code to create an image is called Docker files. Docker Hub is the public cloud where images are placed so that the user can pull or push the images whenever needed. It also provides the feature to distribute the images to particular area which is called as private registry [1] [2].

Computational analysis has made a movement from open source code to reproducible research. It is the methodology to share code, record results and make it available for future use. In order to reuse the existing code, the user has to reconfigure the operating system environment and all the dependencies of the code. This can be achieved through Docker platform where Docker image captures the code context. R is the powerful language for statistical computation and graphics. It is completely audible unlike other GUI analysis programs. Rocker [7] is the collection of Dockerfiles and pre-built Docker images for running R programs using Docker containers

Big data is the large volume of data which consists of either homogenous or heterogeneous data. Storage space, CPU speed, I/O availability are the resources to be considered for analyzing large dataset whose size is beyond the capability of commonly used Software tools to capture, manage and process information. One of the important features of container technology is clustering which promotes parallel processing of containers in different Docker hosts and redundancy to overcome one time failure of containers. Docker swarm is the tool for clustering and scheduling Docker containers. It consists of multiple Docker hosts which run in swarm mode.

II. DOCKER AS A DEVELOPMENT ENVIRONMENT

Docker simplifies the workflow and communication, since it directly runs on the underlying host Machine's operating system. Therefore it is called as operating system level virtualization. Docker can be installed in Linux, MacOS as well as Windows. The Linux command to install Docker and to check its status is:

sudo apt-get install docker.io
sudo systemctl status docker

If the installation is done properly it will show the status of the docker as active in terminal after running the above commands.

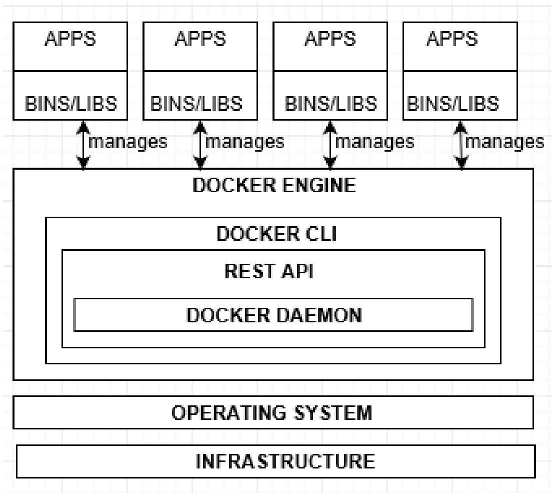


Fig. 1 Docker Container architecture

Fig 1 depicts the architecture of the Docker container. The four important components of Docker are: Docker client and server, Docker images, Docker registries and Docker containers.

A) Docker server and client

Docker follows Client-Server architecture where Docker server and Docker client can resides in same or different machine. Docker server is also known as Docker daemon which is responsible creating, monitoring and managing the states of the Docker containers. Daemon unavailability does not affect container's uptime. It will continue running its application as normal.

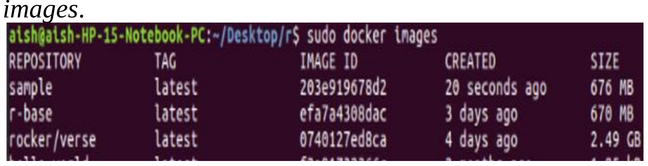
B) Docker engine

Docker engine resides on top of the operating system. It comprises of three major components.

- *Docker Daemon(Server)* – A long running process for managing Docker containers.
- *REST API* – Acts as a medium of communication between Docker client and Docker server.
- *Docker CLI* – A command line interface to execute Docker commands to interact with Docker daemon.

C) Docker images

Docker images are binary archieves of the software. It is an inert, immutable file for running a container. Fig. 2 shows the details such as Repository, tag image id, created date and size of the most recently created images by using the Linux command *sudo docker images*.



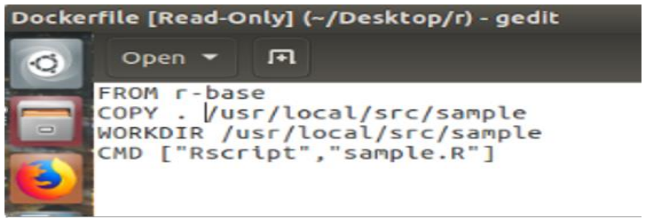
```
aish@aish-HP-15-Notebook-PC:~/Desktop/r$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
sample              latest              203e919678d2        20 seconds ago      676 MB
r-base              latest              efa7a4308dac        3 days ago          670 MB
rocker/verse        latest              0740127ed8ca        4 days ago          2.49 GB
```

Fig. 2 List of recently created images

D) Docker file

Docker file is the text file that uses a specific set of instruction to build a given image. Docker builds the image by reading the instructions from a Dockerfile. Fig 3 shows the Docker file for image named sample of type r-base. Once the Docker file is ready, the user can build the image by using the command:

sudo docker build –t <image name>



```
FROM r-base
COPY . /usr/local/src/sample
WORKDIR /usr/local/src/sample
CMD ["Rscript","sample.R"]
```

Fig 3 Docker file.

E) Docker registries

Docker Registry is used to store built images. Docker provides Docker hub which is the public cloud to place images from where user can pull or push images from single source. Registries are of two types. Public and private registry. Public registry allows everyone to access available images at anytime. Docker hub provides the feature to create private repositories in order to distribute images to particular area.

F) Docker containers

Running instance of the docker image is called Docker container. It runs in an isolated environment by holding all required binary files and libraries for an application. The user can run an image to create a container by:

sudo docker run <image name>

III. DEPLOYING RSTUDIO IN A CONTAINER

Rocker [7] is the collection of Dockerfiles and pre-built Docker images for running R programs using Docker containers. The Rocker project develops the containers such as r-base, r-level and rstudio in the core Rocker repository. Deploying Rocker in a container is shown in a Fig 4.

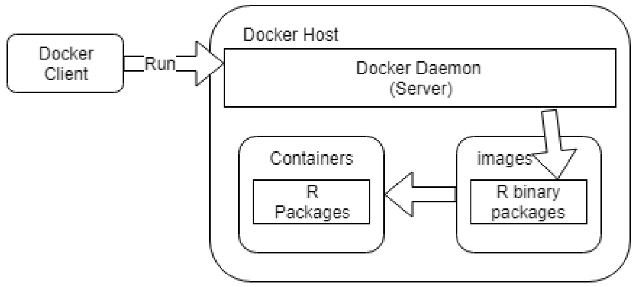


Fig 4 Deploying Rocker

The Docker client can run a Rocker image by interacting with Docker Daemon through Docker CLI. The following command is used to download a Rocker image named rocker/verse.

Sudo docker run –rm –p 8787:8787 rocker/verse

The command will launch RStudio-server invisibly. In order to connect to it, open the browser and enter http://localhost:8787. This means it will launch the RStudio-server in the port number 8787.

III. SWARM ARCHITECTURE

Clustering is the process of grouping a set of objects such that inter-dependency between objects in a cluster must be high and intra-dependency between objects in different clusters must be low. The process of forming a cluster with multiple hosts is called Swarm cluster which has the capability of scheduling the container workloads. The swarm cluster is formed when their Docker engines are running together. The purpose of using docker swarm is to establish and manage the cluster nodes as a single virtual system and modify the service configuration that includes networks and volumes of the standalone containers and can restart the service whenever needed. Fig 5 shows the architecture of Swarm.

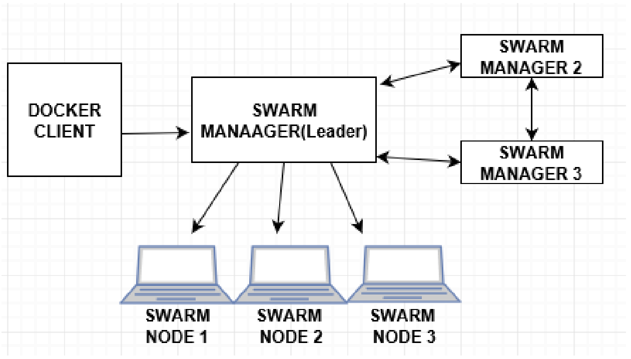


Fig 5 Swarm architecture.

The machines in a swarm are called as nodes. There are two types of nodes: Manager Nodes and Worker nodes.

A)Manager nodes

Every swarm cluster should contain at least one manager node. There can be more than one manager node also. Fig 5 consists of three manager nodes. But there can be only one Manager Leader in case of multiple managers who manages all the worker's services. Manager node is responsible for:

- Adding additional nodes to the cluster.
- Single pool of resources support.
- Scheduling decisions.
- Maintaining and monitoring state of all containers running on different Docker hosts

To set the docker environment for ManagerNodes use the following linux commands:

docker-machine env <manager node name>
docker swarm init

B) Worker nodes

Swarm cluster can have one or more Workers depending on the workload of the service. Each worker has its own optimal state of replicas, storage resources and ports for the services. Docker swarm keeps track of worker's desired state, because once if worker node seems to be unavailable. docker schedules tasks to other nodes. For adding workers in the Swarm cluster use the following linux command:

```
docker-machine env <Worker node name>
```

C) Features of Swarm cluster

Since each worker node is an isolated Docker engine, Docker helps the to configure the system easily and faster which the user to deploy the source code in worker node in less time and effort. Some of the key features of Swarm cluster are

- *Cluster management*– Additional orchestration software is not needed to create or manage swarm. It can be through Docker Engine CLI to manage swarm cluster.
- *Decentralized design* – Entire swarm can be built from a single disk image using Docker engine.
- *Declarative service model* – Docker engine allows the user to describe the various services in the application stack.
- *Scaling* – swarm manager maintains the desired state by adding up or removing tasks.
- *Desired state reconciliation* – Reconcilition can be easily achieved since the manager node continuously monitors the cluster state. Creates new replicas if failure occurs.
- *Multi host networking* – The user can specify an overlay network for application services which can be addressed during initialization or updating of an application
- *Service Discovery* – DNS server is embedded in the swarm through which the user can query any container running in the swarm.
- *Load balancing* – Swarm internally distributes service containers between nodes
- *Rolling updates* – Roll-back feature is also provided in order to move back to previous version if anything goes wrong

IV. DISTRIBUTED ANALYSIS IN DOCKER SWARM

A) Challenges to analyze growing data

The global internet population is growing day by day. Large volume of data is being generated every minute which is floating around these days. As per the statistics, for every 60 seconds, Google receives over 400k search queries, YouTube users upload 71 hours of new videos, Pinterest users pin 3,472 photos, Facebook users share 240K pieces of content, Twitter users 277K tweets, and Apple users download 48K apps. Analyzing such large volume of heterogeneous data is big deal. The main resources to be considered for big data analysis are storage space, CPU speed, I/ O availability which affects the rate of information flow into and out of the system.

B) Map Reduce Vs Docker swarm

Distributed analysis is the powerful methodology for memory-intensive work which affects only data-related tasks. Since the data is distributed across different host machines reading, transforming and computing data needs proper architecture. Map Reduce is the technique used to divide files into smaller parts. Each part is mapped to different slave node by master node. Reduce phase will reduce the result produced by each slave node. But in order to reuse Map Reduce program, each time the user must set up Hadoop environment for deployment which can be overcome by Docker. Once the image is created for Map Reduce program which holds complete environment for deployment, it can be reused for later use. Fig 6 shows how manager distributes 10TBof data in worker nodes.

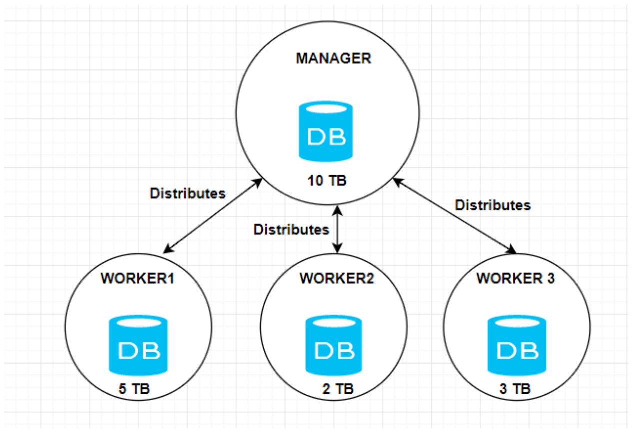


Fig 6 Data distribution in Swarm cluster.

D) Distributed Data Analysis Architecture

Manager node manages and distributes data across multiple worker nodes. Scheduler in manager node is responsible for scheduling the tasks to be assigned to worker nodes. Manager node can also execute containerized applications apart from managing the workers. Though the Docker engine runs in swarm mode, it can also run standalone containers.

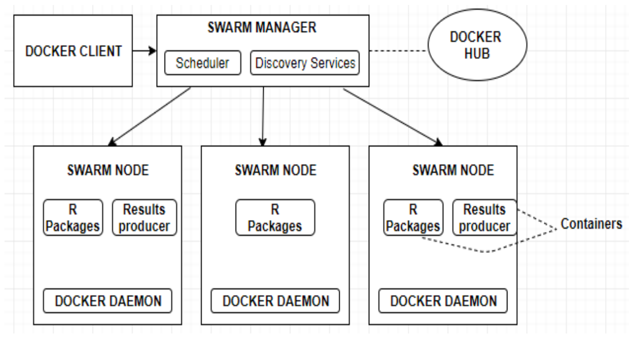


Fig 7 Distributed Data Analysis Architecture

Fig 7 depicts the architecture for distributed data analysis which consists of three worker nodes with R package containers running in it. In the above case manager node maps the data to be processed in different three workers. Worker1 will collect all the results calculated by different workers to produce the final result.

E) High Throughput Achievement

The high rate of producing results can be achieved since each swarm node process the part of the result concurrently in an isolated manner. So that results are produced in a minimal amount. Throughput can calculate as follows:
Without Docker swarm,

$$Throughput = \frac{R}{T}$$

With Docker swarm,

$$Throughput = \sum_{i=0}^{n} \frac{R}{T}$$

Where, R =maximum rate of result production.
T =given amount of time.
n = number of swarm nodes used

F) Optimized Resource Utilization

Resources such as storage space, CPU speed, I/O availability are utilized in an optimized way when data is distributed in swarm nodes whose size cannot be managed to process data in a single host. Utilization of resources can be calculated as follows:

Without Docker swarm,

$$Utilization = \sum_{i=0}^{n} \frac{R}{T} * 100$$

With Docker swarm,

$$Utilization = \frac{\sum_{i=0}^{n} \frac{R}{T} * 100}{m}$$

Where, P = number of resources used
R = number of available resources
n = number of processes
m = number of swarm nodes used.

V. CONCLUSION

Docker automates the application environment when they are containerized. Analysis of large volume of datasets with variety of Data can be done efficiently by running Docker engine in swarm mode with optimized usage of resources.

References

[1] K.N Bala Subramanya Murthy , Manu.A.R, Jitendra Kumar Patel, Shakil Akhtar, V.K Agarwal PhD "Docker Container Security Via Heuristics-Based multilateral Secutity – Conceptual and Pragmatic Study".

[2]Thanh Bui ,Aalto university school of science, "Analysis Of Docker Security".

[3]Mathijis Jeroen Scheepers "Virtualization and Containerization of Application Infrastructure : A Comparison".

[4]Electronic Design - difference between containers and virtual machines http://www.electronicdesign.com/

[5] Mudit Verma, Mohan Dhawan, IBM Research,"Towards a More Reliable and Available Docker-based Container Cloud".

[6]Docker docs - https://docs.docker.com/

[7]ROpenSci –Rocker introduction https://ropensci.org/blog/2014/10/23/introducing-rocker

[8]Karl Matthias and Sean. P . Kane –"Docker up and running – shipping reliable containers in production" O'reilly publications.