

Computing Clusters and Management

Course Title: Computing clusters, grids and clouds

Supervised by: Andrey Y. Shevel

Submitted by:
Md Anisul Islam
anisul.iut@gmail.com

6th June, 2018

Table of Contents

Introduction	3
Why We Need Cluster?	3
History	4
Components of a Cluster	4
Types of Computing Clusters	4
Failover Clusters.....	5
High-performance Clusters (HPC)	5
Load Balancing Clusters	6
Advantages of Clusters	6
Cluster Management	6
Apache Mesos: Architecture.....	6
Apache Mesos: Resource Offer.....	7
Demonstration: Google Kubernetes	8

Introduction

A computer cluster is a distributed or parallel computer system. To meet the requirement to be a computing cluster, there should be following **three** aspects:

- Collection of interconnected standalone computers
- Under one administrator
- Working together as a single integrated computing resource

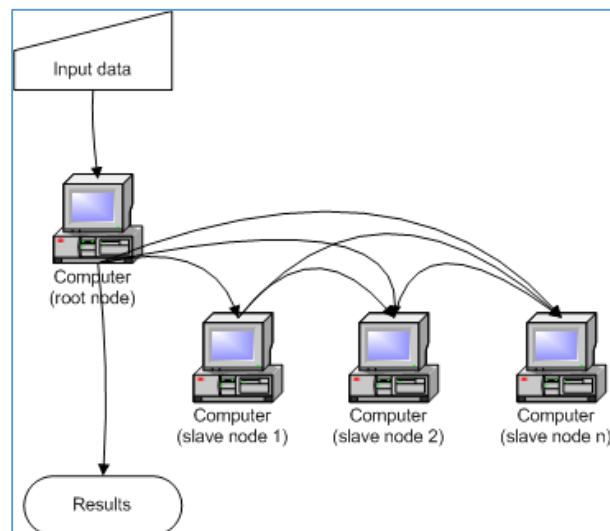


Fig 1: Example of a computing cluster

Fig-1 describes layout of a simple cluster. The cluster consists of one Master node/Root node. The incoming tasks (to be performed) from the user first comes to master node and then master distributes the tasks among all the slave nodes.

Why We Need Cluster?

Clusters are used when content is critical or when services have to be processed as quickly as possible. The Beowulf clusters are used in science, engineering and finance to work on projects

of protein folding, fluid dynamics, neural networks, genetic analysis, statistics, economics, and astrophysics among others. Researchers, organizations and companies are using clusters because they need to increase their scalability, resource management, availability or processing to supercomputing at an affordable price level. The parallel clusters are heavily involved in rendering high quality graphics and animations.

History

In 1960s, IBM developed an alternative of linking large mainframes to provide cost optimization in form of commercial parallelism. The first commodity clustering product was ARCnet, developed by Datapoint in 1977. The original PC cluster project- Beowulf was started at the Center of Excellence in Space Data and Information Sciences NASA in early 1994. Beowulf consists of one master or server node, and one or more client nodes connected together via Ethernet.

Components of a Cluster

Components of a standard cluster computing system are:

- Multiple stand-alone computer
- Operating system
- High-performance interconnects
- Middleware

Types of Computing Clusters

There are three basic types of computing clusters:

1. Failover clusters
2. High-performance clusters
3. Load balancing clusters

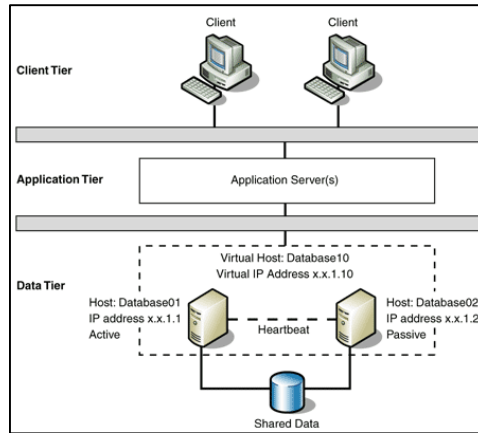


Fig. Failover Cluster

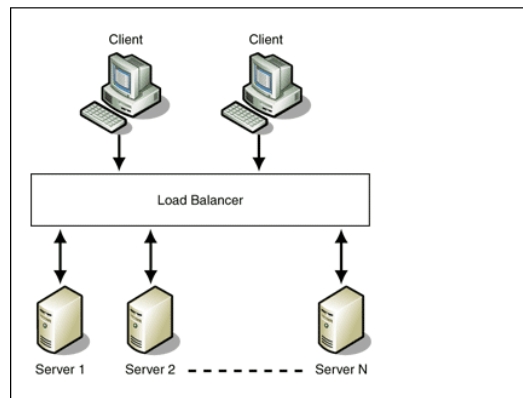


Fig. Load balancing cluster

Failover Clusters

These clusters are designed to provide uninterrupted availability of services to the end-user. With a high-availability cluster, nodes can be taken out-of-service for maintenance or repairs. Additionally, if a node fails, the service can be restored without affecting the availability of the services provided by the cluster.

High-performance Clusters (HPC)

High-performance clusters are designed to exploit the parallel processing power of multiple stand-alone computers/nodes. Applications like data mining, parallel processing, weather modeling, simulations etc. uses high-performance cluster. Beowulf cluster is also a good example of HPC

Load Balancing Clusters

Load Balancing Cluster distributes tasks among multiple nodes. If a node fails, tasks are redistributed between the remaining available nodes. This type of distribution is typically seen in a web-hosting environment, web servers with large client base.

Advantages of Clusters

- Better availability and reliability
- Scalability
- Enhanced network performance
- Easy troubleshooting

Cluster Management

A typical cluster management includes following tasks:

- Monitoring nodes
- Resource management
- Failure Recovery

Popular existing containers with cluster management tool are: Swarm, Fleet, Google Kubernetes and Apache Mesos etc.

Apache Mesos: Architecture

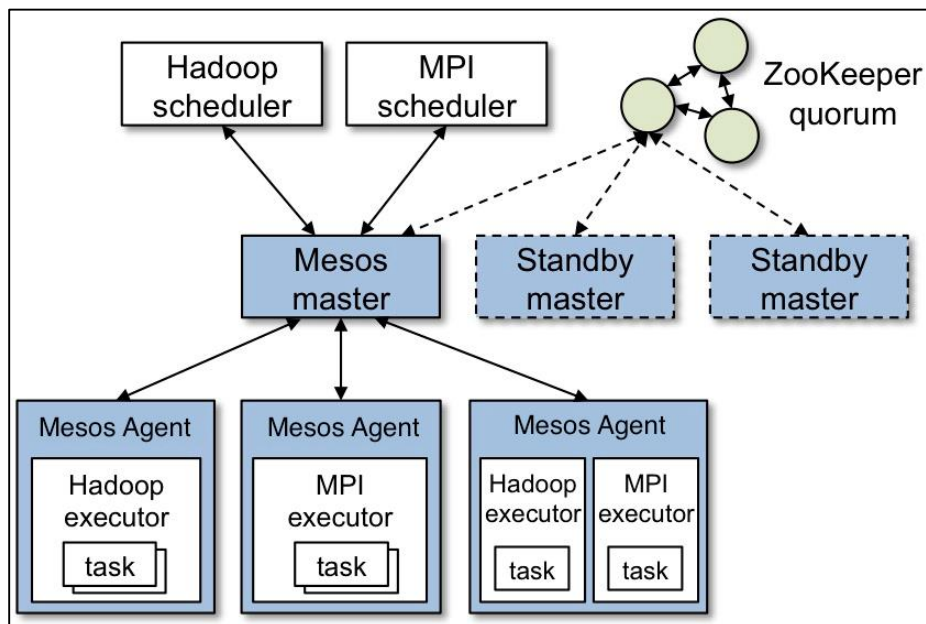


Fig. Architecture of Apache Mesos

Master daemons manages agent-daemon running in nodes. Master decides how many resources to offer to each framework according to a given organizational policy, such as fair sharing or strict priority. A framework running on top of Mesos consists of 2 components: a scheduler and an executor. Scheduler registers with the master to be offered resources and executor process that is launched on agent nodes to run the framework's tasks. Master determines how many resources are offered to each framework, the frameworks' schedulers select which of the offered resources to use.

Apache Mesos: Resource Offer

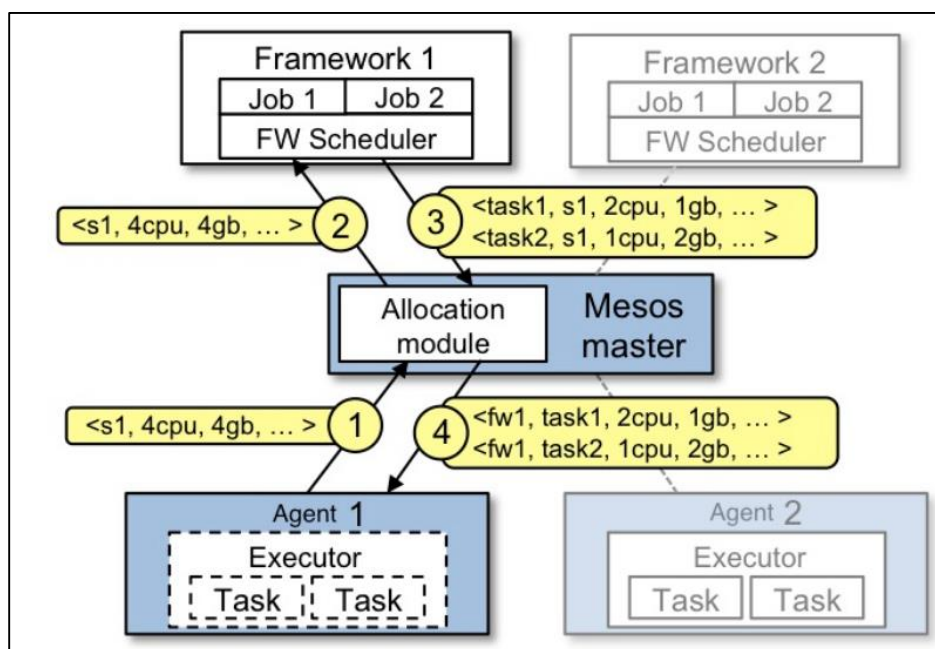


Fig. Resource offering mechanism in Apache Mesos

Agent 1 reports to the master it has 4 CPUs and 4 GB of memory free. Master then invokes the allocation policy module, which tells Framework 1 should be offered all available resources. The master sends a resource offer describing what is available on Agent 1 to Framework 1. The framework's scheduler replies to the master with information about two tasks to run on the agent, using `<2 CPUs, 1 GB RAM>` for the first task, and `<1 CPUs, 2 GB RAM>` for the second task.

Finally, the master sends the tasks to the agent, which allocates appropriate resources to the framework's executor.

Demonstration: Google Kubernetes

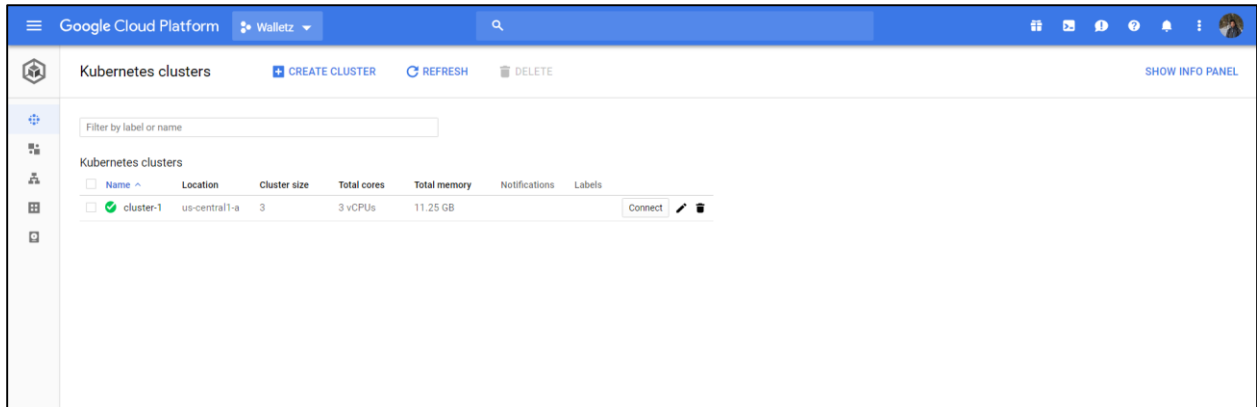


Fig. Kubernetes Cluster initialization on Google Cloud Platform

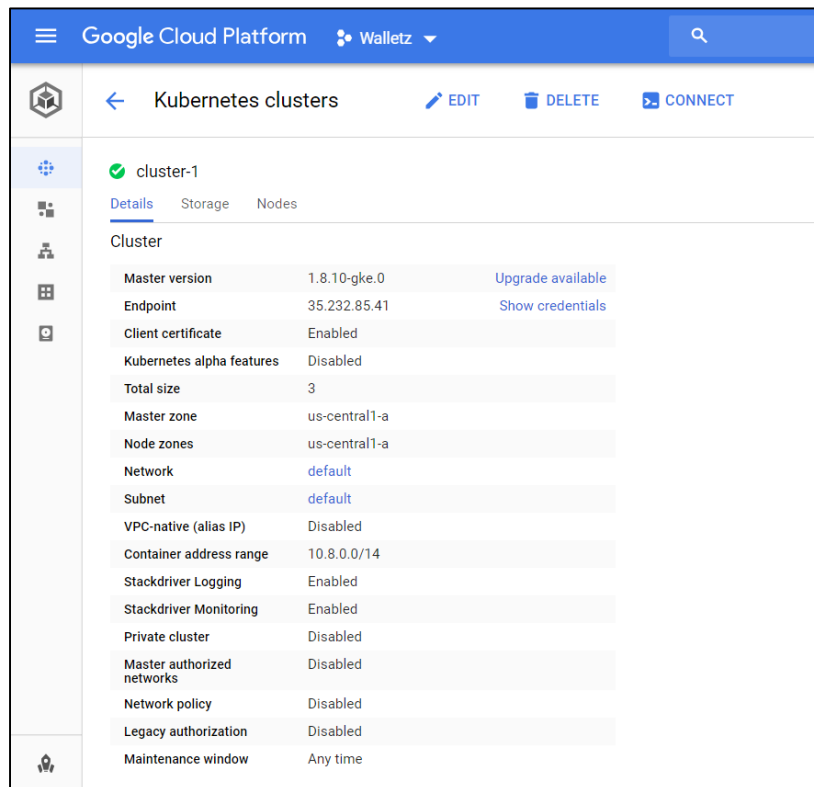


Fig. Attributes of initialized cluster

The top part of the image shows the Google Cloud Platform console interface for a Kubernetes cluster named 'default-pool (3 nodes, version 1.8.10-gke.0)'. The specifications are as follows:

Name	default-pool	
Size	3	
Node version	1.8.10-gke.0	Change
Node image	Container-Optimized OS (cos)	Change
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)	
Total cores	3 vCPUs	
Total memory	11.25 GB	
Automatic node upgrades	Disabled	
Automatic node repair	Enabled	
Autoscaling	Off	
Preemptible nodes	Disabled	
Boot disk type	Standard persistent disk	
Boot disk size in GB (per)	100	

The bottom part of the image shows a terminal window with the following output:

```

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to walletz-1346.
Use "gcloud config set project" to change to a different project.
anis_2803@walletz-1346:~$ gcloud container clusters get-credentials cluster-1 --zone us-central1-a --project walletz-1346
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster-1.
anis_2803@walletz-1346:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
gke-cluster-1-default-pool-51dc4733-21s8    Ready    <none>   10h   v1.8.10-gke.0
gke-cluster-1-default-pool-51dc4733-mhbb    Ready    <none>   10h   v1.8.10-gke.0
gke-cluster-1-default-pool-51dc4733-rc14    Ready    <none>   10h   v1.8.10-gke.0
anis_2803@walletz-1346:~$

```

Fig. Cluster nodes specifications and status

The image shows the Google Cloud Pricing page for Kubernetes Engine. The configuration on the left is as follows:

- Number of nodes: 3
- Instance type: n1-standard-1 (vCPUs: 1, RAM: 3.75 GB)
- Local SSD: 0
- Location: Iowa (us-central1)
- Average hours per day each node is running: 24 hours per day
- Average days per week each node is running: 7
- Persistent Disk Location: Iowa (us-central1)
- Persistent disk storage: GB
- Snapshot storage: GB

The cost estimate on the right is as follows:

- Persistent Disk: Iowa, Storage: 10 GB, \$0.40
- Kubernetes Engine: 3 x, 2,190 total hours per month, Instance type: n1-standard-1, Region: Iowa, GCE Instance Cost: \$72.82, Container Engine Cost: \$0.00
- Sustained Use Discount: 30%
- Effective Hourly Rate: \$0.033
- Estimated Component Cost: \$72.82 per 1 month
- Total Estimated Cost: \$73.22 per 1 month (USD)

Fig. Pricing of Kubernetes Engine