Container Runtime Performance Evaluation of Kubernetes and OpenShift

Preprint · August 2019

DOI: 10.13140/RG.2.2.12310.37441

1 author:

Prashanth Jakkula
National College of Ireland
3 PUBLICATIONS **1** CITATION

SEE PROFILE

Container Runtime Performance Evaluation of Kubernetes and OpenShift

Prashanth Jakkula
Msc. Cloud Computing
National College of Ireland
x18167063

***Abstract*—Virtualization in the field of cloud computing has opened doors to innovation and new technologies. With the help of virtualization, different applications are able to cater different types of workloads. Containerization is one such type of virtualization which enabled isolation of application with its own operating system. This improved the portability of applications. This paper aims to evaluate and compare two open source application containerization tools, Kubernetes and OpenShift and further understand the differences between them.**

Keywords—Virtualization, Virtualization Infrastructure, Containerization, Kubernetes, OpenShift

I. INTRODUCTION

With the present day technologies extending into the cloud ecosystem, virtualization has become one of the vital components and an ever growing technology in IT. It is enabling the market leading organizations to consolidate the workloads to achieve scalability. Virtualization at different levels is offering different solutions. Virtualization can be defined as a combination of software and hardware that enables resource consolidation to create virtual machines thorough which multiple operating systems can be launched on a single system [4].

The capacity of Multi-Tenant Operating Systems:

As the NIST definition suggests, Cloud computing provides a platform for resource pooling. This implies the utilization of several resources by consumers in a multitenant fashion where in various physical and virtual resources are assigned dynamically as per user requirements. The most prominently used technologies to achieve this are the virtualization and the hypervisors.

Each type of virtualization can be implemented on different hypervisor. Hypervisor helps in providing an interface to the virtual machine with the bare metal resources of the system the virtual machine is installed on. Type 1 hypervisor has Virtual machines launched directly on the system resources. In other words, it can also be called as a hardware embedded hypervisor since it runs on baremetal. Type 2 hypervisor is a hypervisor which runs on a host operating system. This paper compares application level virtualizations using containers on type 2 hypervisor.

There are different types of virtualizations depending on the resources needed. They are as follows:

- Storage Virtualization
- Server Virtualization
- Network Virtualization
- Application Virtualization
- Memory Virtualization
- Hardware Virtualization
- Desktop Virtualization

Containerization is similar to hypervisors but the abstraction layer is at the application. This approach wraps the application in an abstraction layer known to be a container which provides isolation. Isolated containers act as an independent system capable of addressing user requested tasks. This technology is considered highly flexible as migration of applications in the form of containers involves less complex procedures than migrating virtual machines. This paper aims to understand the working and architecture of Kubernetes and OpenShift. The later section of the paper suggests a method evaluate the performance of Container Runtime.

A. Hypervisors and Containers

Like discussed previously, Hypervisors are the most vital component for cloud computing to achieve Virtualization. This enables hardware consolidation for cloud solutions. Based on the type of hypervisor, there are various market offerings of hypervisors. EXSi is such example for type 1 hypervisor which can be considered as one of the early implementations in private clouds. Public cloud services such as Rackspace Amazon Web Services are based on Xen Hypervisor which is another example for type 1. Other examples for type 2 include VM Ware WorkStation, Virtual Box etc.

The main difference between a hypervisor and a container is based on the hypervisor. For example, in an environment where multiple operating systems are required a hypervisor is essential (Linux, Debian, Windows, RHEL Linux etc.). Hhardware abstraction must be provided at the Virtual Machine level to allow running multiple operating systems or operating systems of different versions.

In a container deployment, a single operating system is shared. The operating system's libraries and binaries are shared with applications wherever required. And hence, unlike virtual machines, these deployments are considerably smaller in size. Because of lesser footprint of containers, a physical machine can accommodate hundreds of them which is not possible with Virtual machines because of their size.

Since containers does not have an operating system, restarting or shutting down a container does not require shutting down the physical machine. There's an incredible level of parallel application density (Loosely coupled) among Linux variations, with libraries sporadically required to finish the Portability. Therefore, it is ideal to have one compartment pack-age that will run on all Linux-based mists.

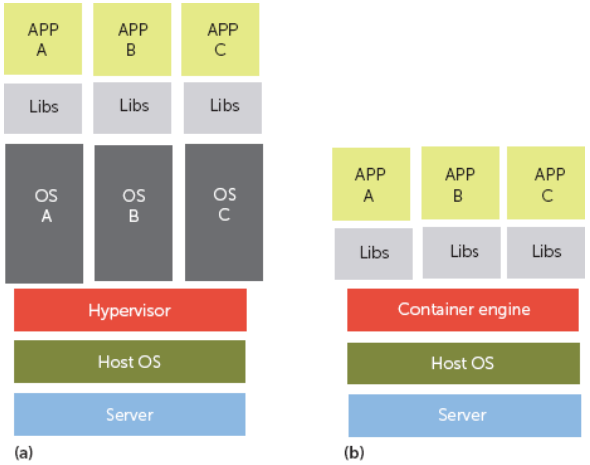

Figure 1. Hypervisor vs Container

The figure compares (a). Hypervisor deployment with a container deployment. It can be seen that a normal VM set up has multiple Oss sitting above a host OS that has a Hypervisor. Here abstraction for hardware resources is provided by the hypervisor over a host OS. (b). In Containers, instead of a hypervisor a container engine is deployed over a host OS. This provides all the libraries and binaries required for the applications.

There is an array of different containerization tools currently available in the market. With growth in different cloud providers for containers such as Dockers, Amazon's Container Service etc., innovation and research in the field of containers has perceived a significant growth. Specific to this study, Kubernetes and Red Hat's OpenShift have been considered. The motivation behind narrowing down to these two tools is that they are open source tools, free to use and they are open to the research community. However, there are few cloud providers such as IBM and Google are noted to be running the infrastructures on containers.

II. KUBERNETES

Kubernetes is a production grade container orchestration service. It is an open source tool to automate container deployment, management and scaling across host clusters. It works with a wide range of containers such as Docker. As discussed previously many cloud service providers offer Kubernetes based PaaS. And also Kubernetes is being offered as services by cloud vendors as their own distribution. It clusters containers that forms an application into logical units. This makes it easier to manage and discover. And in the case of a container failure Kubernetes launches another container automatically or assigns another one. This is why container orchestration is needed. In production environments or in cloud services, it is vital to follow the SLAs and container orchestration helps in managing without complexity. Kubernetes provides different levels of orchestration functionalities, some of them are as follows:

i. Load Balancing and Service Discovery:
Kubernetes is capable of identifying a container by DNS name or IP address. And in case of high traffic inflow to a particular container, Kubernetes is capable of balancing the load by distributing the network traffic to stabilize the deployment.

ii. Storage Orchestration:
Kubernetes allows mounting of storage systems automatically to any container. It allows to mount user specific storage such as public cloud storage and local storage.

iii. Rollbacks and Rollouts:
Kubernetes allows users to define a specific state of a container to roll back in case of a failure. It can also change the state of a container to a desired state. As an example if a new container is created old containers can be removed and all their resources can be assigned to the new one.

iv. Automated bin packing:
Kubernetes automatically assigns the user specified CPU and the memory required for each container. It assigns the resources to the containers based on the resources requests.

v. Configuration Management and secrets:
Kubernetes allows management of sensitive information such as authentication tokens, passwords, ssh keys etc. Application configurations and sensitive information can be updated without relaunching of container images and without exposing the stack configuration.

Unlike traditional Platform as a Service system, Kubernetes works at container level. Though it does not run at hardware layer, it provides some features similar to Platform as a service such as scaling, deployment, load balancing, monitoring and logging. However, these features are optional to Kubernetes as it is a non-monolithic service.

A. Architecture of Kubernetes:

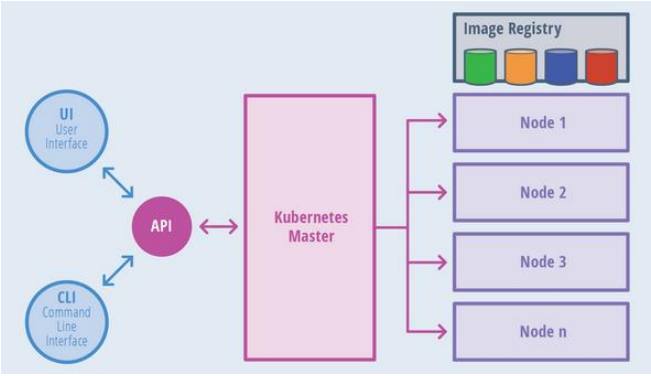

Figure 2. Kubernetes architecture overview

The figure represents the architecture of kubernetes where the kuvbernetes master node is linked to the nodes with image registry and users can access through API calls. Kubernetes architecture provides a loosely coupled and flexible mechanism for service discovery. It follows a master-slave type of architecture and hence it has atleast one master node in the cluster and multiple compute nodes. A master node is capable of controlling the entire cluster. It can operate functions such as scheduling the deployments, defining APIs etc. All the other nodes in the cluster are controlled and managed by master node. Each node in the cluster runs container runtime such as Docker. It also contains an agent which heps in communication with master node. Additional components are also included in the node so as to run operations such as logging and monitoring. Nodes are the viital working components of a cluster they represent netwrok, compute and storage resources to applications. Ndes in this cluster can also be virtual machines that are running on a physical machine or bare metal servers that are in a data center.

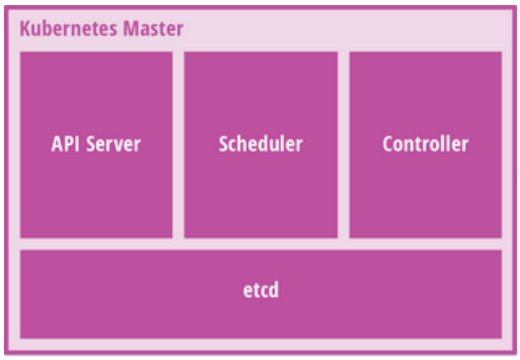

Figure 3. Kubernetes Master

Looking deeper into the architecture, a master node in kubernetes consists of an API server, a scheduler and a controller. It also contains an open source key value database called etcd. The master sends requests to etcd to retrieve the information from nodes such as state of pods, nodes and the containers. This helps kubernetes to be scalable by creating abstraction layer between the underlying infrastructure and the application. Api-Server represents Kubernetes API which is the front end of Kubernetes. Scheduler watches and manages the newly created pods and assigns nodes. Kube-Controller controls different aspects of a cluster. Each controller is a process by itself. It has the following controlleers:

- Node Controller
- Replication controller
- End Point controller and
- Account and token controller

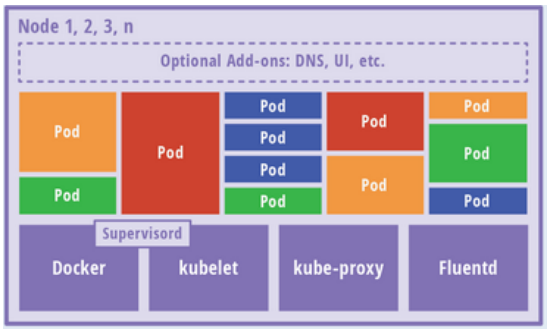

Figure 4. Node in Kubernetes

The above figure represents the components of a node. A kubernetes node is the funtional component where in various components are communicating with each other and the master node. They run the containers in a sets called as pods. The node contains the container runtime, a kubelet, a kube-proxy, entd and the pods.

A **Kublet** is an agent that runs on each node of the cluster. It assures that the containers are running in pods. Kublet monitors the set of pod specifications that are provided in various mechanisms. It makes sure that the specified containers are healthy and running properly. Since it is a part of kubernetes, Kublet cannot manage the containers that are not created in Kubernetes.

Kube-Proxy is similar to a network proxy which runs on each node of the cluster. Applications that run on multiple pods as a network service is a kubernetes service which has Kube-Proxy as the implementing part. The network communication between the pods and the containers is maintained by kube-proxy. It maintains network rules to allow the communication. These rules help to maintain communication frm inside and outside the cluster. If the opersting system packet fltering is available, Kube proxy implements it to the communication.

Fluentd is a logginfg agent that manages logging endpoints for applications and cluster logs. It takes care of cooection of logs, distribution and parsing.

III. OpenShift

OpenShift is a cintainer platform service created by RedHat which is based on Docker and Kubernetes. It offers monitoring, logging and service registry of containers. It is an entrprise Kubernetes application platform. It can be used as a production platform for container deployment. There are differetnt distributions of this platform such as OKD and OpenShift Online. OKD is a community driven distribution where as OpenShift Online is a software as a service distribution and OpenShift Dedicated offered as managed service. Since the OpenShift Contaoner platform was based on Kubernetes, it shares the similar workflow.

OpenShift Container Platform is used to deploy hybrid cloud applications by deploying both in public clouds and in data centers. Kubernetes is known to be efficient in managing the applications, but it lacks performnace in platform level deployment processes. OpenShift excels in offering management tools for platform processes. Some of the key features of OpenShift are as follows:

i. Custom Operating System

OpenShift uses Red Hat Enterprise Linux Core OS (RHCOS). This operating system is designed for containers that offers some of the preeminent features of the Core OS. It was designed for container platform and provides operator based management and streamlined upgrades. The operating system also includes **Ignition**, an initial configuaratio for containers. It also includes **cri-o** which is a kubernetes narive container runtime that can integrate with the operating system for optimum efficiency. Another functionality Kubelet is also included which is an agent in Kubernetes. Kubelet is used for launching and minotoring containers.

ii. Simplified Setup and update:

Latest distributions of Openshift allows the usres to deploy clusters for production with the execution of a single command. But with necessary user permissions. And since it supports various cloud platforms, it also allows deployments in various clouds and datacenters. Clusters running with RHCOS have automated updates for all the machines. Since OpenShift has control over the services and systems that run on each machine, updates are designed to become automated including the operating system of the machine.

iii. Cintainer Lifecycle Management

Openshift provides an in built container lifecycle manager calle OPenshift Lifecycle Manager (OLM). It provides OperatorHub to store and distribute the operators to theuser end. Operators are basic fundamental, units of OpenShift. They provide a flexible and convenient way for the user to deply software components and applications. Wit the help of Operators, OpenShift can automate the upgrades of OS and the control Plane.

Other key features include Red Hat Quay which is a Container registry that is capable of providing operators to containers and clusters. It is a Quay.io container registry.

A. Architecture of OpenShift:

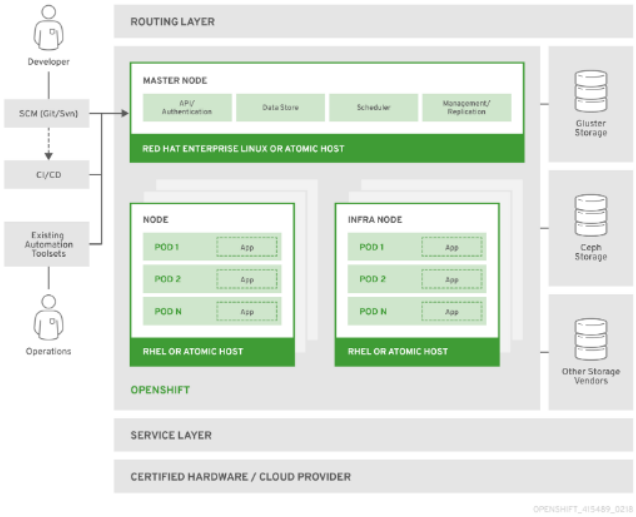

Figure 5. OpenShift Architecture Overview

OpenShift Container platform is a combination of multiple decoupled units that form a songle unit and work together. Since it is based in the Kubertentes engine, it store the objects in etcd which is a key value store. These multiple services are categorised based on functions:

- REST APIs, which represent each object of the core.
- Controllers, read those APIs, modifies objects and reports status

Similar to Kubernetes architecture, OpenShift conatins Master node and nodes. All the nodes are controlled by master node. As shown in figure 5, a master node contains an API server, a data store, Scheduler, a Management server for replication. All the nodes in the cluster are managed and controlled by the master node. Each component of the master node has its functionality. Master node is the control plane of the system. API server is responsible for maintaining API and authentication calls to the nodes and clusters. Data store handles the flow of data from master to nodes and between the nodes and helps in maintaining the state of a cluster or a node. Scheduler is responsible for scheduling processes to pods. In case of failure scheduler assigns jobs to the newly created nodes in the cluster. Manager handles the healtht check of each node in the cluster. It is responsible for managing replication of nodes. It helps in maintaining the self healing property of Kubernetes. It also contains an integrated container registry to maintain service registry and logging functions.

Nodes in OpenShift run Red Hat Enterprise Linux (RHEL) to run the applications. Each node contains a set of containers called pods. They can also comprise of container images and pods. Similar to Kubernetes, Pods are considered to be a unit of container orchestration. They also contain requires services to run docker containers, service proxy and a Kubelet. Openshift is responsible for creating nodes from cloud servoce providers, Virtual machines or ohysical machines. Nodes contain node objects which ar ethe representation of those nodes. And these node objects are interacted by Kubertes platfrom. Since the master node is responsible for health cheks, a node is neglected in the cluster unless it passes the health checks.

Similar to Kubernetes, OpenShift nodes also contain Kubelet whichis an agent between the master and the node and a service proxy to run a network proxy to identify the services defined in the node's API. This gives the node the capacity to forward TCP and UDP streams across the system.

IV. KUBERNETES VS OPENSHIFT

OpenShift is also been known as Enterprise Kubernetes as specified by Red Hat. However, there are some differences between OpenShift and Kubernetes. OpenShift is developed based on Kubernetes platform with more features to manage and control the clusters. The basic difference is that Kubernetes is an open source framework on which OpenShift was built and being offered as a product in

different variants. OKD is the open source version of OpenShift.

a. Support:

Since Kubernetes is an opensource community backed framework, it has a self-support model. In case of any failure, it is on the users risk to find a solution. But in OpenShift, since it is a product offering, it has its pay and use support model. However, it depends on the type of distribution being used.

b. Complexity:

OpenShift is entirely dependent on the Red Hat Enterprise Linux. The product requires RHEL to be installed on the host system to run OpenShift. It also requires Red Hat Core OS to be implemented by the master node. This is a requirement to control plane operations. These cannot be installed on other Linux distributions. But Kubernetes can be installed on any Linux distribution such as Ubuntu, Debian etc. Kubernetes also provides many installation tools. Some are useful for cloud deployments and some are more inclined for universal use. But depending on the user specification to deploy clusters, these tools can be chosen.

c. Security

OpenShift is considered to be a more secure platform relatively. This could be because of the target group that OpenShift is addressing. The default policies of OpenShift are mire strict than Kubernetes ones. As it restricts to run a container runtime as root, most of the images on Docker Hub are not operable on OpenShift. However, it can be disabled.

It is important for some environments to have some level of security permissions in real time. OpenShift offers RBAC security as an integral part of the system. But in some deployments users tend to use Kubernetes without any kind if RBAC security. Not only RBAC but also the authentication and authorization model is important. This can be achieved by integrating the deployments with Active Directory. OpenShift makes it easier to integrate and also provides authorization to external applications. OpenShift also allows to use the following features:

- Container Registry
- Logging Stack (Fluentd, ElasticSearch)
- Prometheus Monitoring
- Jenkins

O-Auth can be used to authenticate all of these systems. This makes it easier to manage permissions to the applications. And it also allows additional features like logging from ElasticSearch and Kibana.

V. Design and Procedure

OpenShift and Kubernetes are both container management systems that are capable of deploying multiple clusters both on VMs and PMs. To evaluate the performance of both the deployments using OpenShift and Kubernetes, a test setup is considered. A testing system is created for both the deployments using same configurations and both are deployed on a Linux virtual machine. Single node setup was created using the following tools:

Minikube: It is a system to implement Kubernetes in a local system. It allows to create a single node cluster over different operating systems.

MiniShift: It is a local deployment tool that allows to run a single node OpenShift container on a local system. This is available only with the opensource version of OpenShift i.e. OKD. It makes use of libmachine for VMs and OpenShift Origin is used to run clusters.

The test system specifications are:

- Minikube on Linux (amd64)
- VM Specs: 2 VCPUs, Memory 2048MB, Disk 20000MB
- Kubernetes v1.15 on Docker 18.09.5

```
$ minikube start
* minikube v1.2.0 on linux (amd64)
* Creating none VM (CPUs=2, Memory=2048MB, Disk=20000MB) ...
* Configuring environment for Kubernetes v1.15.0 on Docker 18.09.5
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Pulling images ...
* Launching Kubernetes ...
```

Figure 6. System Specifications

The above figure shows the system specifications on which the Minikube was launched. And the cluster information sis given in the following figure.

```
$ kubectl cluster-info
Kubernetes master is running at https://172.17.0.8:8443
KubeDNS is running at https://172.17.0.8:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

Fugure 7. Cluster Information

Similar setup is used tto launch a single node on OpenShift. OpenShift comes with prometheus as an add on and hence there is no requirement to install it. Prometheus is a tool to monitor the aactivity in the cluster. It allows the users to monitor the API calls, the communication between the nodes, pods and the clusters.

VI. Performance Evaluation

Prometheus helps in monitoring the activity of both deployments. Since containers does not provide a whole interactive desktop to run applications, it is considered hard to run benchmarking tools over the cluster deployments. Hence Prometheus was considered. As it can be used in both the deployments, it is considered to monitor the memory and CPU access times in Kubernetes and OpenShift clusters. This approach was previously discussed by Aly in which they have used Heapster, a monitoring tool as a way to run benchmark between both platforms running Hono. They also implemented Wilcoxon test to accept or reject the recordings. For this single node deployment Wilcoxon test is not required as it requires vide range of variables. The throughput of the systems can be the application running on both container deployments. The measurements are taken

from the static workload to increased workloads from different time periods.

A single component metric performance analysis is suggested by Malik in which metrics are taken from a principal component of analysis. Here each component or the recording generated is taken as its performance metric. And it measures how it contributes to the performance analysis pf a system. And hence it was suggested that a comparison for such metrics is necessary for each value of the component. This helped in detecting the regression in the values.
Kraft also suggested the analysis of Virtual machines to find out the performance analysis. He measured the performance of container systems by measuring the overhead of the Virtual Machines. Their study emphasized more on the latency in Virtual machine's disk I/O requests. This approach has been considered to record the disk I/O request-response to find out the performance regression between Kubernetes and OpenShift.

a. Setting up Prometheus:

A configuration is required by Prometheus server that defines the endpoints for scraping the data along with the access frequency. The configuration is set to record data from the end point of Kubernetes container and OpenShift.
The endpoints defined are used to scrape data from container deployments. The two ports are used to pull CPU access rate and the memory access rate respectively. Once the Kubernetes is deployed, the dash board can be accessed. And the endpoint is used to scrape the data.

```
$ kubectl get pods -n kube-system  -w
NAME                                READY   STATUS    RESTARTS   AGE
coredns-5c98db65d4-9ftfc            1/1     Running   1          18m
coredns-5c98db65d4-qd95g            1/1     Running   1          18m
etcd-minikube                       1/1     Running   0          17m
kube-addon-manager-minikube         1/1     Running   0          16m
kube-apiserver-minikube             1/1     Running   0          16m
kube-controller-manager-minikube    1/1     Running   0          17m
kube-proxy-qn4r2                    1/1     Running   0          18m
kube-scheduler-minikube             1/1     Running   0          17m
storage-provisioner                 1/1     Running   0          18m
```

Figure 8. Kubernetes Status

Figure 8 represents the running status of Kubernetes. The status of minikube with components such as controller, scheduler, Kube-proxy etc.
Once the Prometheus server is launched as a container in docker, the targets can be seen in the dash board with all the needed information. All the metrics that are scraped from endpoint are viewable. `node_cpu` shows the CPU perfromance of the node in dashboard in the form of a graph.

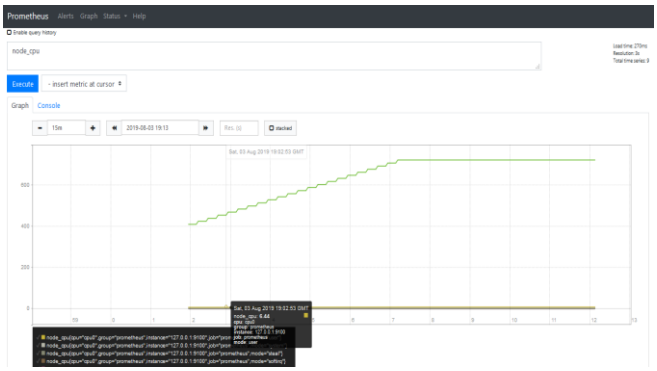

Figure 9. CPU Performance of Node

The graph in figure 9 shows the CPU performance of the node inKubernetes over every 15 seconds interval. Similar test is ran in OpenShift to find the performance of OpenShift

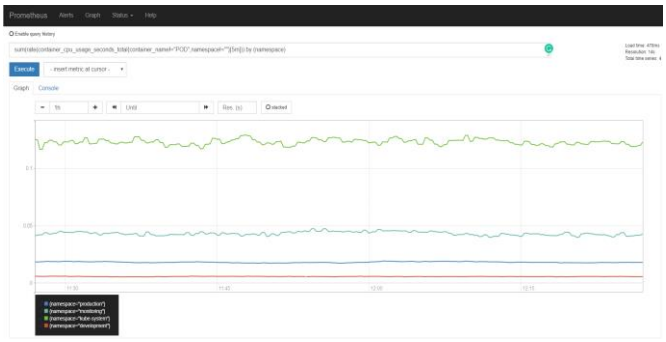

Figure 10. Performance of cpu_node in OpenShift

The graph generated by Prometheus for CPU access time in OpenShift container in shown in the figure 10. It also includes the monitoring of the same node.

VII. Conclusion

This study suggests the difference between the performance of both systems that are launched with same system configurations. It is noted that the discrepancies are found between both the deployments but it depends on the deployment and the throughput given. However, form the data collected it can be seen that both the systems work differently with the same configuration. The CPU performance of the node in Kubernetes increased gradually and stayed constant. But OpenShift had expressed varying performance in Prometheus. Since this adaptation for measuring the container runtime performance is based on empirical studies, there is a scope for a benchmarking tool targeting the container deployments. However, the findings suggest that the discrepancies are equivalent on both platforms.

References

1. D. Bernstein, "*Containers and Cloud: From LXC to Docker to Kubernetes*," in IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014. doi: 10.1109/MCC.2014.51.
2. H. Malik, B. Adams, and A. E. Hassan, "Pinpointing the subsystems responsible for the

performance deviations in a load test," in Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on. IEEE, 2010, pp. 201–210.

3. M. Aly, F. Khomh and S. Yacout, "Kubernetes or OpenShift? Which Technology Best Suits Eclipse Hono IoT Deployments," 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA), Paris, 2018, pp. 113-120. doi: 10.1109/SOCA.2018.00024
4. OKD, The Origin Community Distribution of Kubernetes (Online Source) Available at: https://www.okd.io/ (Accessed on: 27-07-2019)
5. OpenShift Container Platform 3.7, Architecture (Online Source) Available at: https://access.redhat.com/documentation/en-us/openshift_container_platform/3.7/pdf/architecture/OpenShift_Container_Platform-3.7-Architecture-en-US.pdf (Accessed on: 27-07-2019)
6. OpenShift Container Platform3.10, Architecture (Online Source) Available at: https://access.redhat.com/documentation/en-us/openshift_container_platform/3.10/pdf/architecture/OpenShift_Container_Platform-3.10-Architecture-en-US.pdf (Accessed on: 29-07-2019)
7. Tomasz Cholewa, 10 most important differences between OpenShift and Kubernetes, Cloudowski.com (Online Source) Available at: https://cloudowski.com/articles/10-differences-between-openshift-and-kubernetes/ (Accessed on: 29-07-2019)
8. Minikube, Kubernetes.io (Online Source) Available at: https://kubernetes.io/docs/setup/learning-environment/minikube/ (Accessed on: 21-07-2019)
9. MiniShift, OKD (Online Source) Available at: https://www.okd.io/minishift/ (Accessed on: 24-07-2019)
10. Prometheus (Online Source) Available at: https://prometheus.io/ (Accessed on: 01-08-2019)