



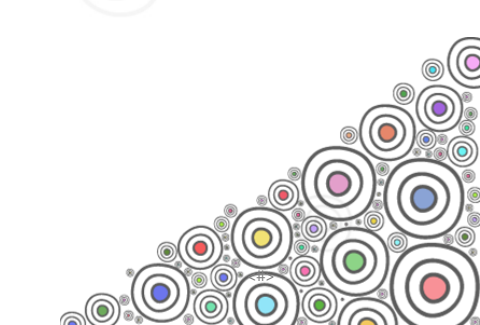
# **PERFORMANCE OVERHEAD OF CO FRAMEWORKS FOR MULTI-TENANT DATABASE DEPLOYMENTS**

**PRESENTED BY INDRANIL GHOSH**

## **Introduction**

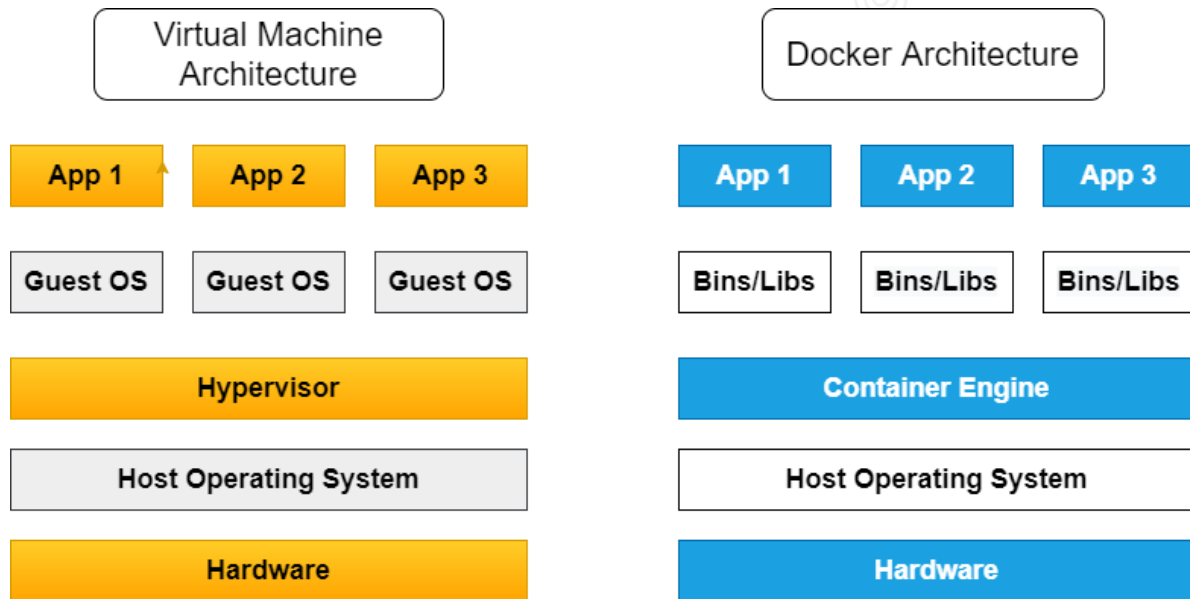
**This paper has discussed on the viability of the light-weight container technology such as Docker for running high-end performance database workloads.**

**The paper looks into the production assessment of Docker engine in the current accentuation and industry acquisition of Container Orchestration( CO) framework. Like Docker Swarm and Kubernetes for the motive of spontaneous placement of cloud based features. Specifically when to manage a CPU bound CPU-bound Cassandra workload in Open Stack.**



# COMPARISON BETWEEN DOCKER ENGINE AND VIRTUAL MACHINE

- ✎ In Docker, the containers running share the host OS kernel. On the other hand VMs, server hardware is virtualized. Each VM has Operating system (OS) & apps. It shares hardware resource from the host.
- ✎ A Dockerized application is just a process that runs on your system. It doesn't require running a Hypervisor , which means there's no guest operating system.
- ✎ reduced size,less code to transfer, migrate and upload workloads.<sup>2</sup>





# USED CO FRAMEWORKS

## 1. Docker Swarm

**Docker Swarm is Docker's orchestration technology that focuses on clustering for Docker containers—tightly integrated into the Docker ecosystem and using its own API.**

## 2. Kubernetes

**Kubernetes or k8s (kubernetes) is an open-source container orchestration system for automating deployment, scaling, and managing of applications. A basic difference between Kubernetes and Docker is that Kubernetes is meant to run across a cluster while Docker runs on a single node.**

## Used database management system

### Apache Cassandra

**The advancement of social media made us to realize the importance of the database system like Cassandra that is capable of handling unstructured data unlike relational database management system.**

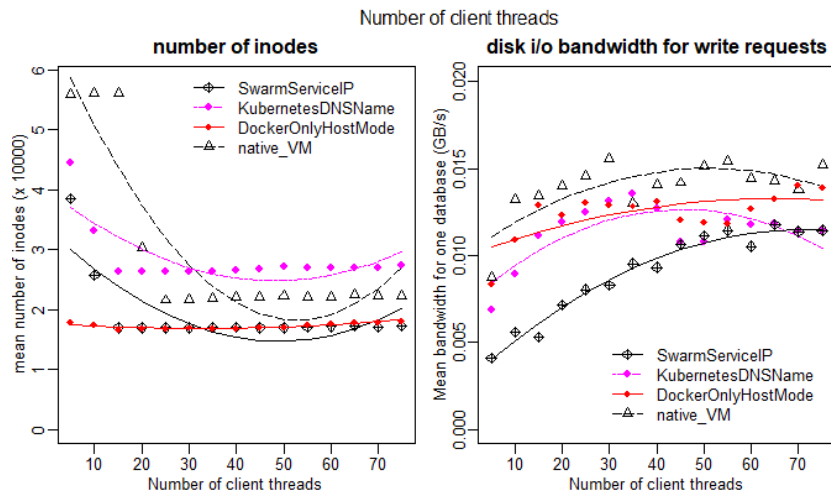
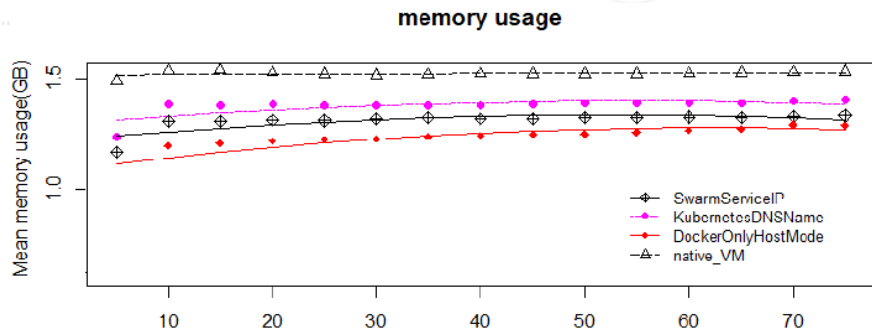


# OVERVIEW OF COMPARED DEPLOYMENTS

- 👉 **A native deployment where Cassandra v2.0.17 is directly installed in an Ubuntu VM.**
- 👉 **A Docker engine deployment where database containers are started from the official cassandra: 2.0 Docker image in similarly sized Ubuntu VMs.**
- 👉 **A Docker Swarm deployment where database containers are inter-connected by means of a stable Service IP address.**
- 👉 **A Kubernetes deployment where database containers discover each other via a DNS name<sup>1</sup>.**

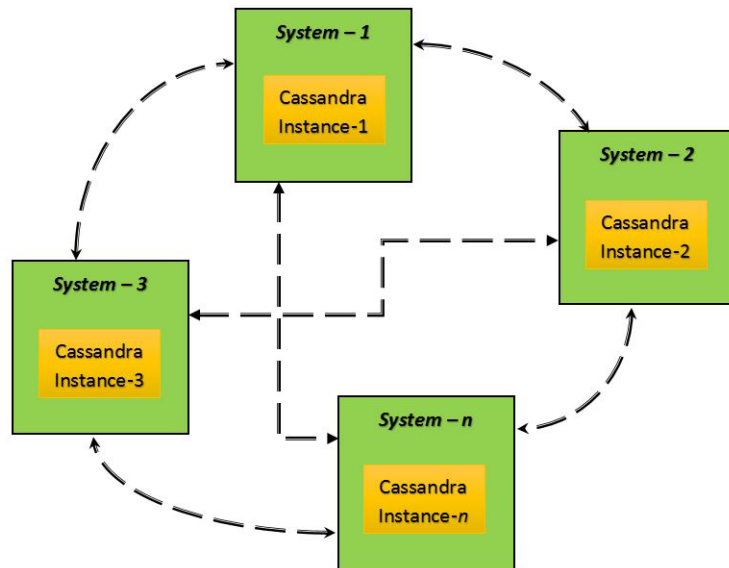
# RESULT

- it has been found that the Docker image with Cassandra 2.0 consumes less memory in comparison to natively installed deployment.
- With respect to response latency Docker performed in much better way. In both the case of Docker Swarm and Kubernetes the network usage appeared high.
- In comparison with Docker engine deployment the disk I/O rate of Docker Swarm has found much lower. Whereas the disk I/O rate of Kubernetes has not decreased like Docker Swarm.<sup>1</sup>



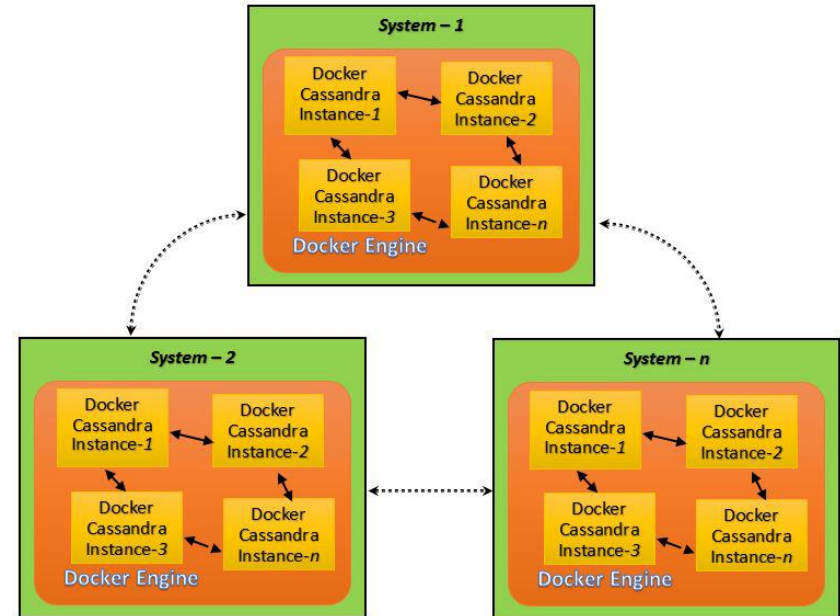
# EXISTING SYSTEM

- 👉 In the present scenario, each server system would host a single instance of Cassandra and clusters can be formed based on the P2P connections established.
- 👉 each system should be connected to at least one other system in the cluster.
- 👉 In the production environment and number of processors per system will not be utilized<sup>2</sup>.



# PROPOSED SYSTEM

- 📞 In this system multiple instances of docker containers have been on the same physical host machine.
- 📞 Cassandra deployment in Docker allows users to define the containers and join with the cluster during runtime by setting up the seeds environment variable by issuing the docker run command.
- 📞 Problem of high availability, in case the physical disk fails.<sup>2</sup>





# CONCLUSION

**In host mode networking the performance overhead of Docker engine has been found relatively small. Though substantial overhead will come into picture for extra CPU consumption for virtual network bridges in service networking solutions of COframeworks. Finally it has been concluded that Co framework should further advance the isolation of container networking approaches that depends on host mode networking and should refine the performance of volume plugins for local persistent storage.**





# REFERENCE

- [1] E. Truyen, D. Van Landuyt, B. Lagaisse, and W. Joosen. "Performance overhead of container orchestration frameworks for management of multi-tenant database deployments". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pp. 156–159.
- [2] A. Jaison, N. Kavitha, and P. Janardhanan. "Docker for optimization of cassandra NoSQL deployments on node limited clusters". In: *2016 International Conference on Emerging Technological Trends (ICETT)*. IEEE. 2016, pp. 1–6.

**Thank you..**