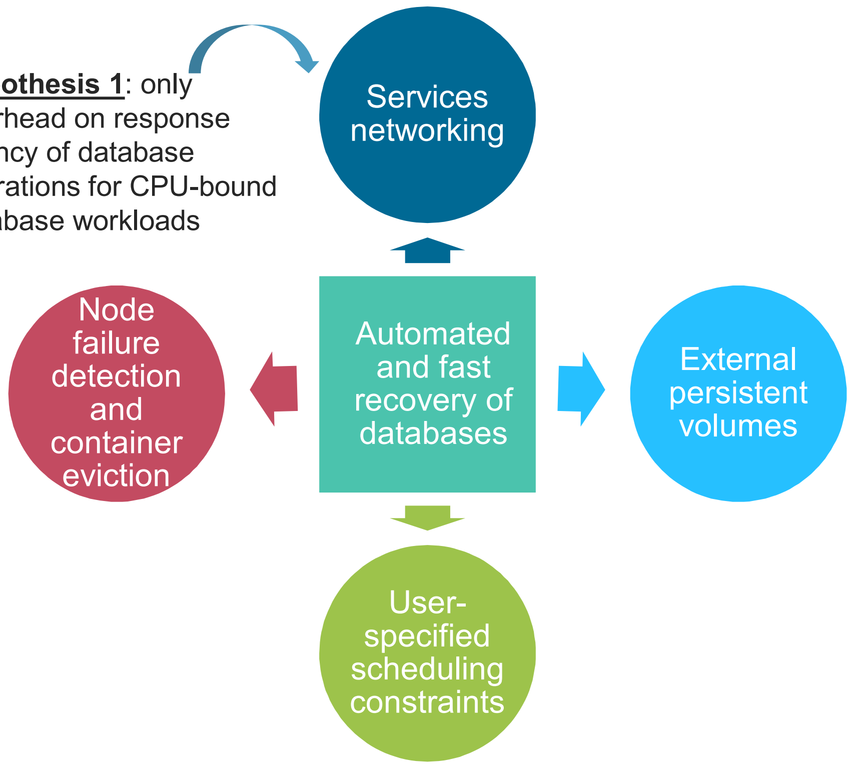
Evaluation of container orchestration systems for deploying and managing NoSQL database clusters

Container orchestration (CO) systems, such as Docker Swarm, Kubernetes, Mesos Marathon and DC/OS, have been initially designed for stateless services. However, they have also been used for running database clusters due to improved automated management. This work evaluates the performance overhead of Docker Swarm and Kubernetes when running NoSQL database clusters, with MongoDB and Cassandra as case study.

https://github.com/eddytruyen/containers_on_openstack/blob/master/README.md
IEEE International Conference on Cloud Computing (IEEE Cloud 2018),
Workshop 2: Cloud Infrastructure, Tuesday July 3, 8:30am

Common CO features for database clusters


Services networking
Automated and fast recovery of databases
Node failure detection and container eviction
External persistent volumes
User-specified scheduling constraints

Hypothesis 1: only overhead on response latency of database operations for CPU-bound database workloads

Hypothesis 2: no overhead on response latency for bottom 3 features

Three services networking approaches

Service networking approach	*Common features among CO systems*	Swarm	Kubernetes	Marathon	DC/OS
Routing mesh with service node ports	*Distributed Layer 4 Load balancer (with ipvs)*	✓	✓		
	Central L4-L7 load balancer (without ipvs)	✓	✓	✓	✓
Virtual network with service IPs	*Distributed Layer4 load balancer (with ipvs)*	✓	✓		✓
	with stable DNS name for services	✓	✓		✓
	IP per container	✓	✓	✓	✓
Host ports networking with VM IPs	*Mapping container port to host port*	✓	✓	✓	✓
	Automated allocation of host ports	✓		✓	✓
	Static host port conflict management	✓	✓		

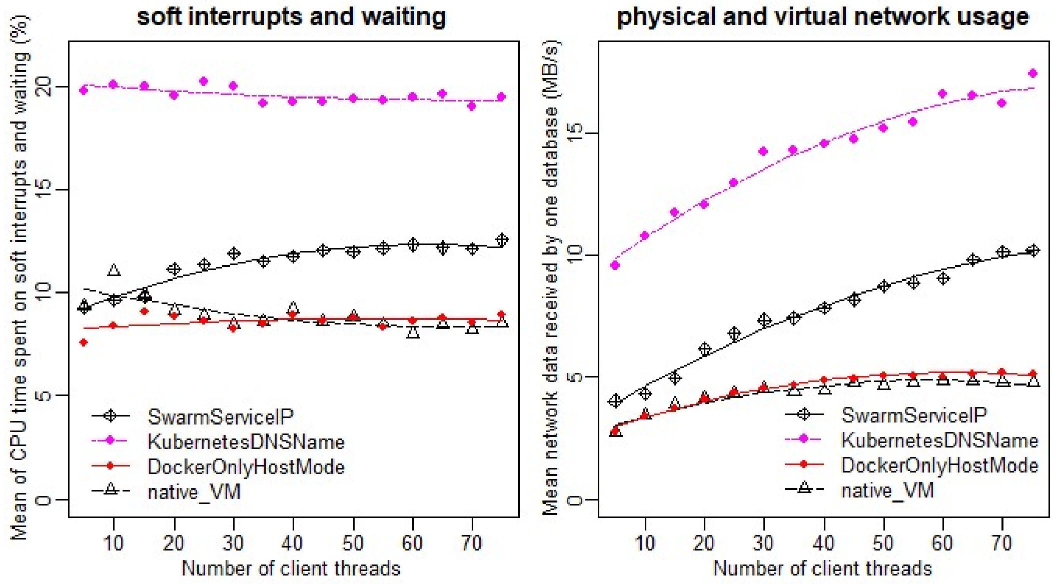
Operation	Update-heavy workloads (A, F)	Read-heavy workloads (B,C,D)
Read	4.8,SwarmFloatingIPHostPort 4.9,SwarmHostNameNodePort 5.5,**K8FloatingIPHostPort** 5.5,**K8ServiceIP** 5.6,**K8HostNameNodePort** 5.7,**K8ContainerDNSName** 23.6,SwarmServiceIP	6.7,**K8ServiceIP** 7.1,**K8ContainerDNSName** 7.5,SwarmFloatingIPHostPort 7.9,**K8HostNameNodePort** 7.9,SwarmHostNameNodePort 8.2,**K8FloatingIPHostPort** 10.6,SwarmServiceIP
Update/ Insert	20.2,SwarmFloatingIPHostPort 21.1,**K8FloatingIPHostPort** 21.3,**K8ServiceIP** 21.8,**K8HostNameNodePort** 22.2,**K8ContainerDNSName** 23.8,SwarmHostNameNodePort 43.8,SwarmServiceIP	18.9,**K8HostNameNodePort** 19.1,**K8FloatingIPHostPort** 20.2,SwarmFloatingIPHostPort 21.3,SwarmHostNameNodePort 26.1,**K8ContainerDNSName** 27,5,**K8ServiceIP** 37.5,SwarmServiceIP

95th quantile of response latencies for MongoDB 3.2.4; 3 node cluster, VMs 2CPU 4GB,Ubuntu 16.04; size of dataset = 23MB; => *in-memory database*

Results for YCSB benchmark[1]

Hypothesis 1:

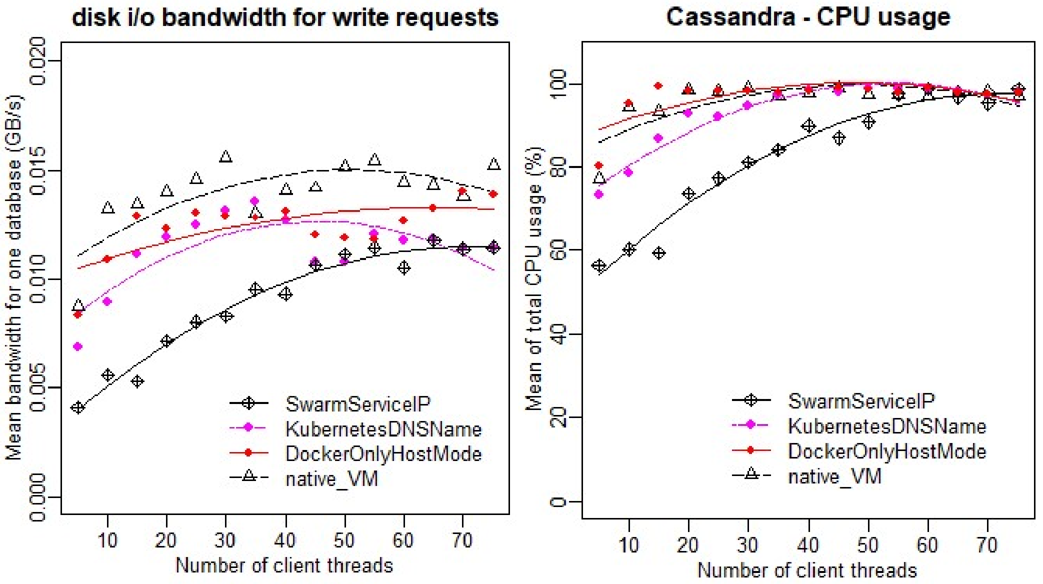
- <u>For CPU-bound Cassandra workloads</u>: services networking causes overhead due to intermediate virtual network bridge in comparison to DockerOnly without NAT (`--net=host option`)



Resource usage graphs for Cassandra 2.0; 3-node cluster, Openstack VMs 2CPU 4GB, Ubuntu 16.04; YCSB workload A, 100% insert operations, size of dataset = 11.7GB => *write-optimized database*

- <u>For CPU-bound MongoDB workloads</u>: Wrt the mean and 95th quantile of response latency metrics, common rankings between services networking approaches are found for specific combinations of YCSB workload type and operation type:

Hypothesis 2: <u>For CPU-bound Cassandra workloads</u>: in comparison to VM-based deployments, local volume plugins for Docker Swarm and Kubernetes are disk i/o performance bottlenecks. However, default volume drivers in DockerOnly deployments are not!



Funded by: IWT-DeCoMAdS

In collaboration with: Red Hat

[1] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with YCSB,” *Proc. 1st ACM Symp. Cloud Comput. - SoCC '10*, pp. 143–154, 2010.

Authors: Eddy Truyen, Matt Bruzek, Dimitri Van Landuyt, Bert Lagaisse, Wouter Joosen
Contact: Eddy.Truyen@cs.kuleuven.be, mbruzek@redhat.com