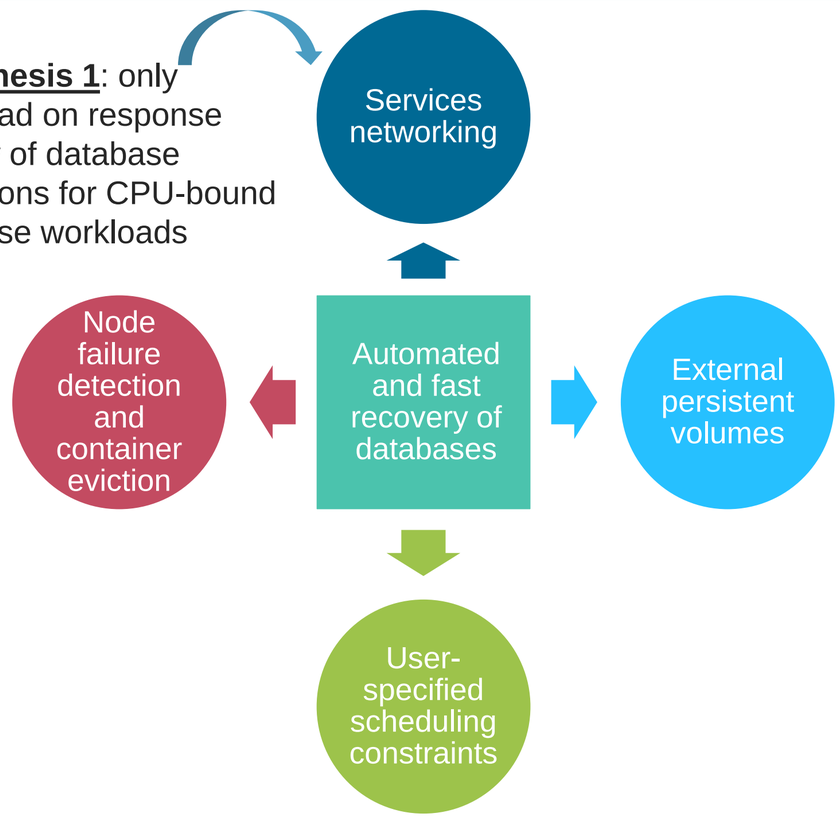
Evaluation of container orchestration systems for deploying and managing NoSQL database clusters

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

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



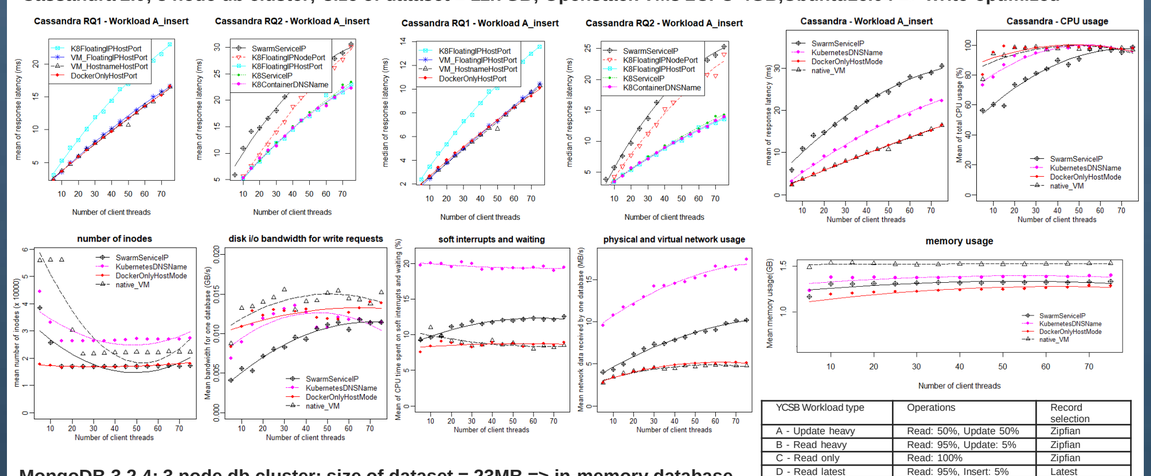
Hypothesis 2: no overhead on response latency for bottom 3 features

Three services networking approaches

Service networking approach	*Common features among CO systems*	Docker Swarm	Kubernetes	Marathon	DC/OS
Routing mesh for global service ports	*Layer4 distributed load balancer (ipvs support)*	✓	✓		
	central L4-L7 load balancer (without ipvs)	✓	✓	✓	✓
Virtual IP network for containers	*Layer4 distributed load balancer (with ipvs)*	✓	✓		✓
	with stable DNS name for services	✓	✓		✓
	IP per container	✓	✓	✓	✓
Host ports networking	*Mapping container port to host port*	✓	✓	✓	✓
	Automated allocation of host ports	✓		✓	✓
	Static host port conflict management	✓	✓		

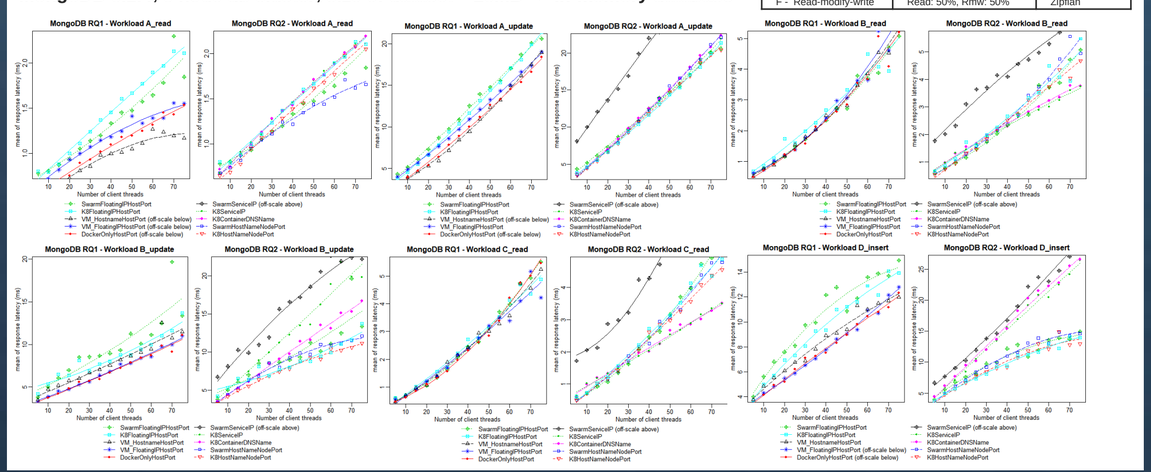
Results for YCSB benchmark[1]

Cassandra 2.0; 3 node db cluster; size of dataset = 11.7GB; Openstack VMs 2CPU-4GB,Ubuntu16.04 => write-optimized



YCSB Workload type	Operations	Record selection
A - Update heavy	Read: 50%, Update 50%	Zipfian
B - Read heavy	Read: 95%, Update: 5%	Zipfian
C - Read only	Read: 100%	Zipfian
D - Read latest	Read: 95%, Insert: 5%	Latest
F - Read-modify-write	Read: 50%, Rmw: 50%	Zipfian

MongoDB 3.2.4; 3 node db cluster; size of dataset = 23MB => in-memory database



Findings for CPU-bound DB workloads

Hypothesis 1:

- Services networking causes overhead due to intermediate virtual network bridge in comparison to DockerOnly without NAT (`--net=host option`)
  - Run containers of services in host mode (beta feature Swarm)
  - Leverage container security policies to improve security isolation (stable feature K8)
- Common rankings between services networking approaches with respect to 95th quantile of response latency for specific combinations of YCSB workload types and operation type

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

each MongoDB deployment is pre-annotated with the mean of the observed 95th quantiles of response latency (ms).

Hypothesis 2:

- Local volume drivers are disk i/o performance bottlenecks in Docker Swarm and in Kubernetes, but not in DockerOnly deployments
  - As a result, in opposition to DockerOnly and VM-based deployments, 100% CPU utilization cannot be achieved

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