Quantifying the Noisy Neighbor Problem in Openstack

Ajay Gulati, Nodir Kodirov, Gautam Kulkarni

OpenStack Summit - Austin, TX - April 28, 2016

Motivation

- Private clouds are meant to run diverse workloads
- A cloud requires consolidation of various resources
  - Shared storage (distributed or over SAN/NAS)
  - Shared networking (host and switch level)
- It gets more critical as over-commitment increases

Application level performance is the ultimate objective

- We try to answer three questions
  - Is there a contention problem in a cloud environment?
  - How quickly does it appear?
  - What are the best practices to reduce contention?

Scope of this Talk

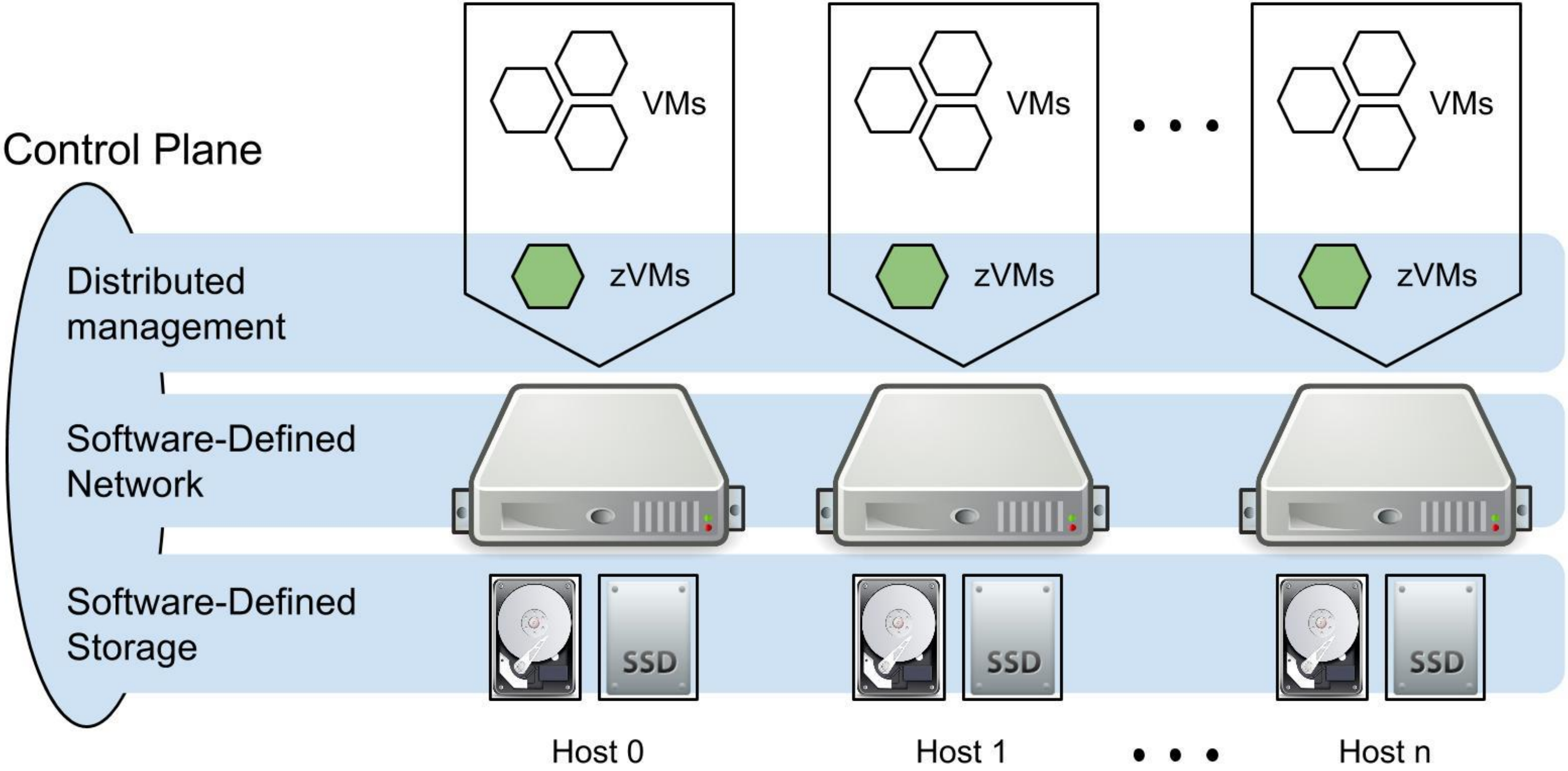
- We used an Openstack cloud deployment with
  - Local and distributed/shared storage
  - Networking using Neutron and OVS
- We did micro- and macro-evaluation to study workload contention
  - Micro-benchmarks
    - Network
    - Storage
  - Macro-benchmarks: enterprise workloads
    - Hadoop Terasort
    - Jenkins job to compile Linux kernel
- Control plane performance

Outline

- Experimental setup
- Stress tool: design and implementation
- Storage, network performance evaluation
- Application performance evaluation
  - Hadoop Terasort
  - Jenkins job to compile Linux kernel
- Control plane performance

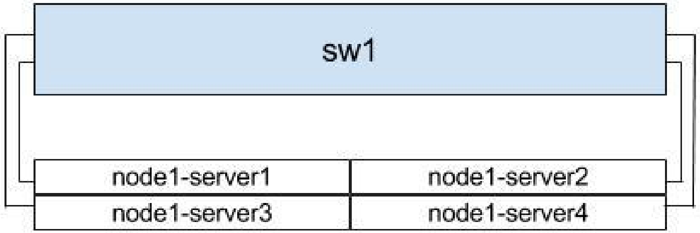
Experimental setup

ZeroStack: Controller-less Architecture



Experimental setup

- Minimum building block is 2U node
- Each 2U node has 4 servers
- Each server has
  - 2 sockets with 8 core Intel Xeon E5-2600
  - 4 x 1 TB HDD
  - 2 x 800 GB SSDs
  - 2 x 10Gbps NICs
    (but we used one NIC in this study)
- OpenStack cloud on Kilo



Symmetric hardware and cloud architecture makes results translate linearly

Stress Tool

- ZeroStack has an OpenStack client in Golang
- Designed and implemented a stress tool using the Golang client
- The tool uses Openstack APIs to set up rich test configurations
- For example
  - Create VMs across different hosts, with same or different subnets
  - Support diverse network topologies
  - Support volume creation across different storage pools/backends
  - Run benchmarks (iperf, ioblazer, fio) within VMs
  - Collect results, analyze and plot them in an automated manner
  - Measure API call performance
- Use Heat Orchestration Template for deploying workloads (Hadoop, Jenkins)

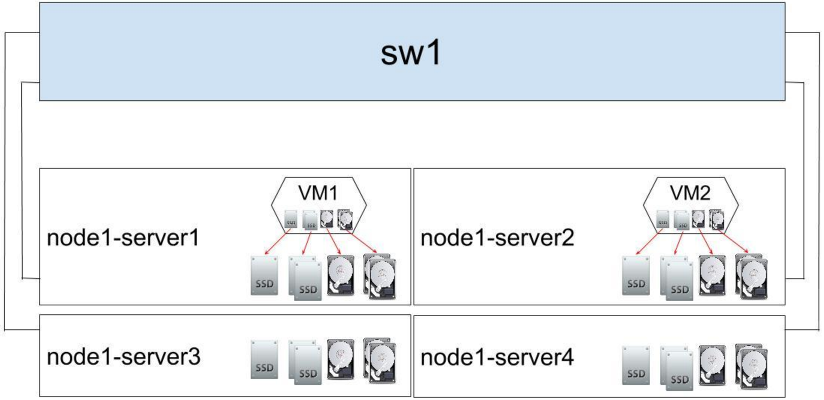
Micro-benchmarking: Storage

Cloud storage pools

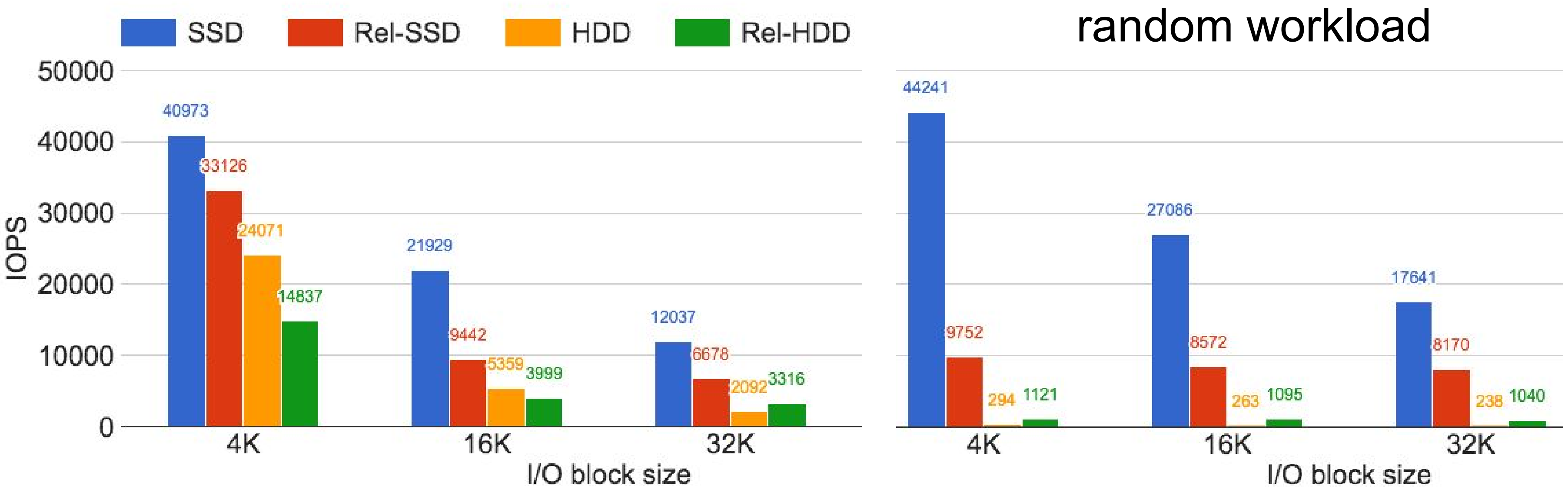
- ZeroStack exposes 4 types storage pools
  - Local SSD
  - Local HDD
  - Reliable SSD
  - Reliable HDD
- Reliable pools: tolerate disk and host failures
  - Default replication factor is 3

Storage Performance Setup

- Used ioblazer, fio, iometer
  - well-suited for virtualized env.
- Benchmark parameters
  - block size (4K, 16K, ..., 64K)
  - queue depth (8, 16, ..., 128)
  - sync/async(buffered)
  - read/write (0, 30, 70, 100%)
  - sequential/random pattern
- Collected over thousand data points
- This talk highlights only some of the data points
- Used X-large KVM VM

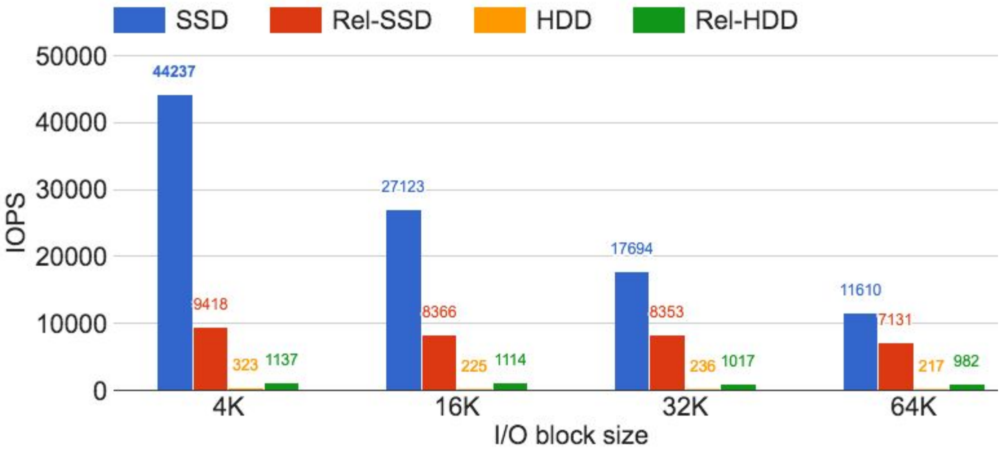


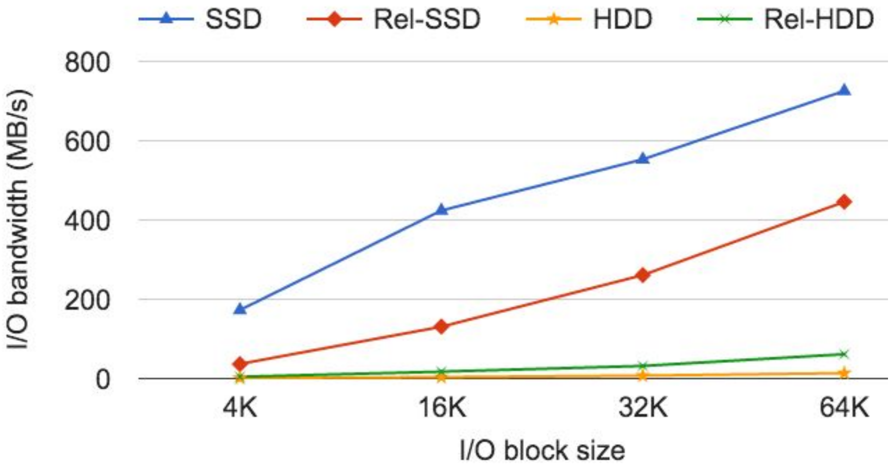
Single VM: sequential vs. random 100% read



Sequential workload: can use either SSD or HDD backend
Random workload: use SSD based pools

Single VM: random 70% read, 30% write





SSD backend should be used for random workloads

Two VMs: random 70% read, 30 % write

Block size	Storage pool type											
	SSD			Replicated SSD			HDD			Replicated HDD		
	w/o inter.	w/ inter.	Δ	w/o inter.	w/ inter.	Δ	w/o inter.	w/ inter.	Δ	w/o inter.	w/ inter.	Δ
4K	44237	32306	**-26%**	9418	9000	-4%	323	277	-14%	1137	1111	-2%
16K	27123	21806	-19%	8366	7983	-4%	225	216	-4%	1114	1056	-5%
32K	17694	15038	-15%	8353	7752	-7%	236	242	+2%	1017	1036	+1%
64K	11610	8939	**-23%**	7131	6704	-5%	217	220	+1%	982	838	-14%

Both VMs get good performance, since storage is not saturated
There is some variance though across hosts: need to control further using storage QoS

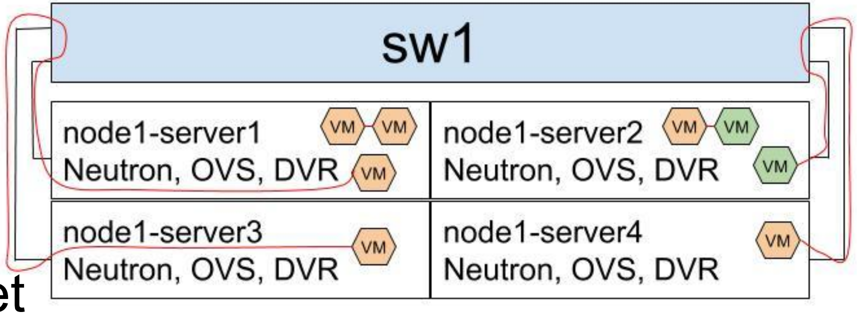
Lessons on storage contention

- Use SSD based pools for random workloads and to avoid VM contention
  - HDD cannot deal well with I/O blender effect
- Have both kinds of pools (local and shared) in your environment
  - No need to use reliable storage for apps with in-built replication e.g. Hadoop, Cassandra
- Always consume local SSD/HDD from the host where VM resides
  - e.g., create nova filter to do it

Micro-benchmarking: Network

Network VM setup

- Combination of different host and OpenStack network/subnet
  - same host, same subnet
  - same host, different subnet
  - different host, same subnet
  - different host, different subnet
- Use iperf by varying
  - message size
  - runtime
  - protocol

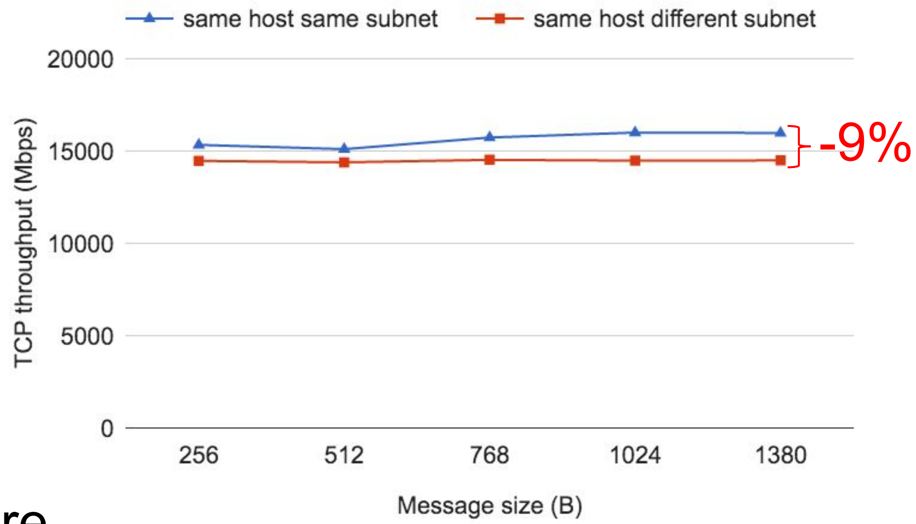


VMs with the same color are on the same network/subnet

VMs with different color are on different network/subnet

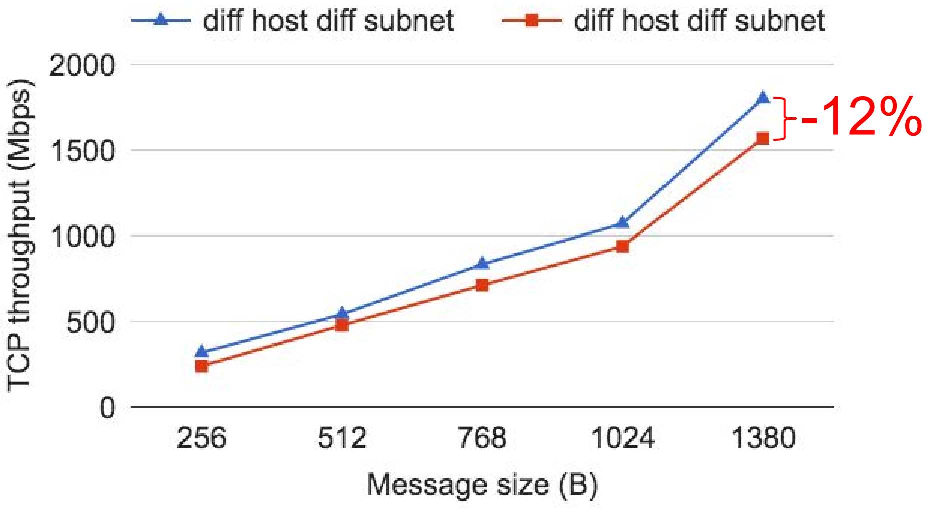
SDN Routing overhead: Same Host

- Neutron with OVS and DVR
- GRE for tenant isolation
- iperf client/server VMs
  - Ubuntu 14.04, 64 bits
  - XLarge (8vCPU, 16 GBRAM)
  - 20 GB local SSD
  - results: mean of 3 runs
- Observations
  - 9% throughput drop due to different OpenStack subnet
  - Virtual router introduces 3 more software hops which consumes more CPU cycles per packet



SDN Routing overhead: Different Hosts

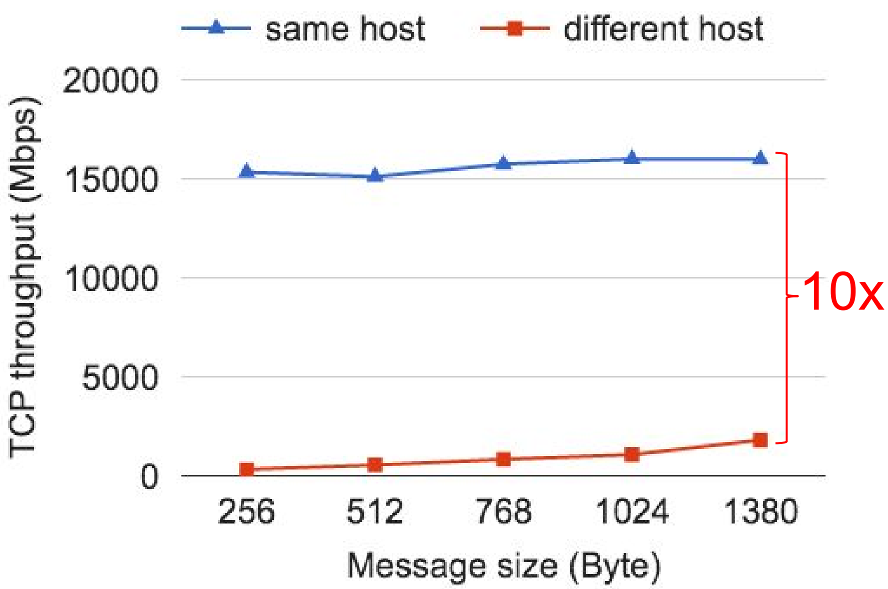
- Similar observations
  - 12% throughput drop due to different OpenStack subnet
- Some suggestions
  - leverage DPDK
  - explore VLAN-based provider but that comes with its own limitations



Use same subnet as much as possible

VM network throughput on same vs. different host

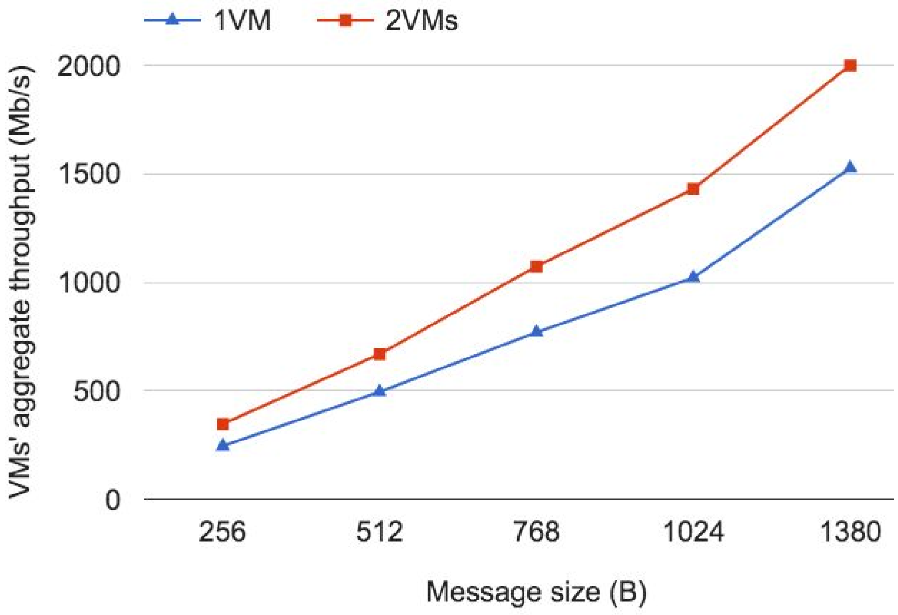
- iperf client/server VMs
  - Ubuntu 14.04, 64 bits
  - X-Large (8vCPU, 16 GB RAM)
  - 20 GB SSD
  - results: mean of 3 runs
- Observation
  - VMs on the same host provide 10x more throughput



Co-locate chatty VMs on the same host using smart placement policies
E.g., Affinity rules (NOT possible on public clouds)

Multi-VM network contention

- Overall network throughput increases as we add mode VMs, but not linearly
- Throughput is OVS bound
- GRE encap/decap consumes high CPU

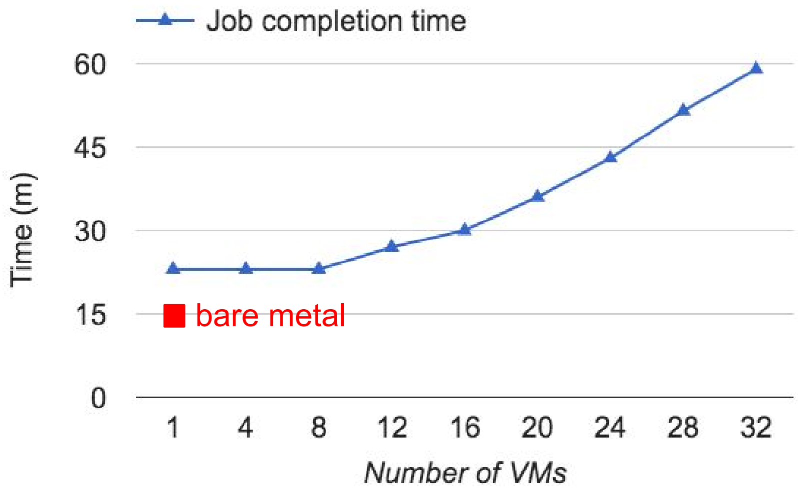


Single VM is not able to achieve 10 Gbps due to CPU saturation
Increase number of VMs for higher aggregate throughput

Enterprise workloads: Jenkins

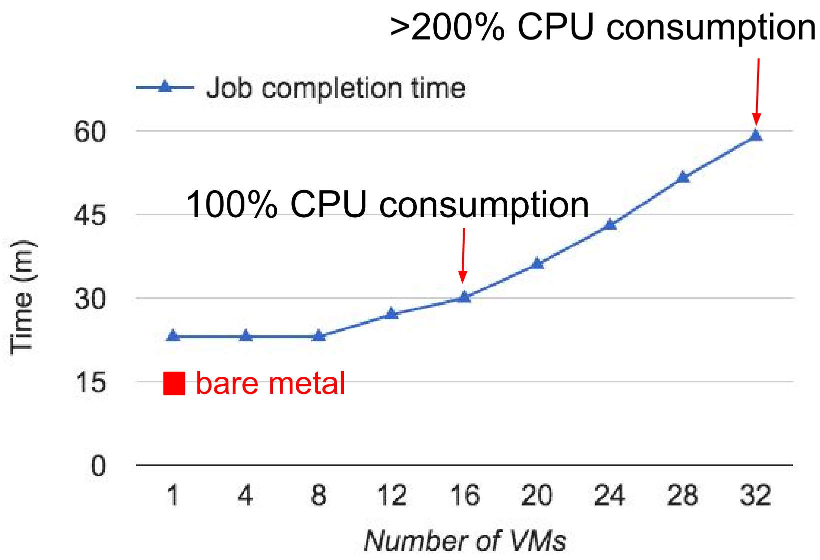
Workload contention: Linux kernel compile

- VM specs
  - Ubuntu 14.04, 64 bits
  - X-Large (8 vCPU, 16 GB RAM)
  - 50 GB Local SSD
- Same job on a bare-metal is faster (23 mins vs. 15 mins)



Workload contention: Linux kernel compile

- VM specs
  - Ubuntu 14.04, 64 bits
  - X-Large (8 vCPU, 16 GB RAM)
  - 50 GB Local SSD
- Same job on a bare-metal is faster (23 mins vs. 15 mins)
- Observations
  - Only 30% increase until full CPU saturation
  - Up to 260% increase w/ CPU overcommit of 2x

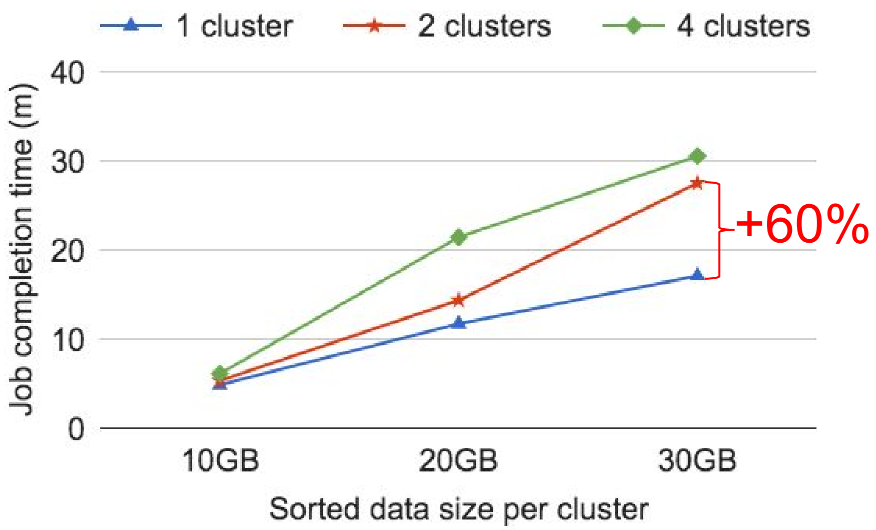


Do not overcommit CPU for compute-heavy workloads
Less critical for batch jobs that are not latency sensitive

Enterprise workloads: Hadoop

Workload contention: Hadoop Terasort

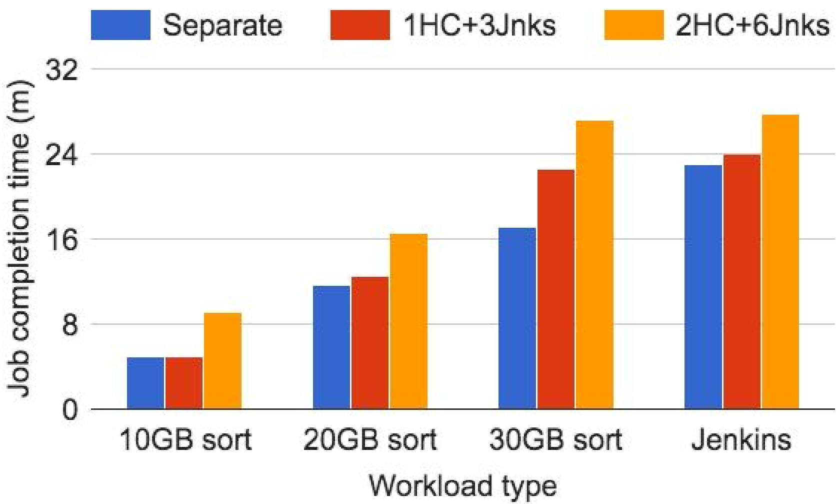
- Run the job on a cluster of 4 nodes
  - one master and three slave VMs
  - all X-Large instances with a 100 GB local SSD volume
  - one salt-master VM to orchestrate cluster creation
- Total data sorted
  - (number of clusters) x (data size)
  - e.g., (4 clusters)x(30 GB)=120 GB
- More data = more contention

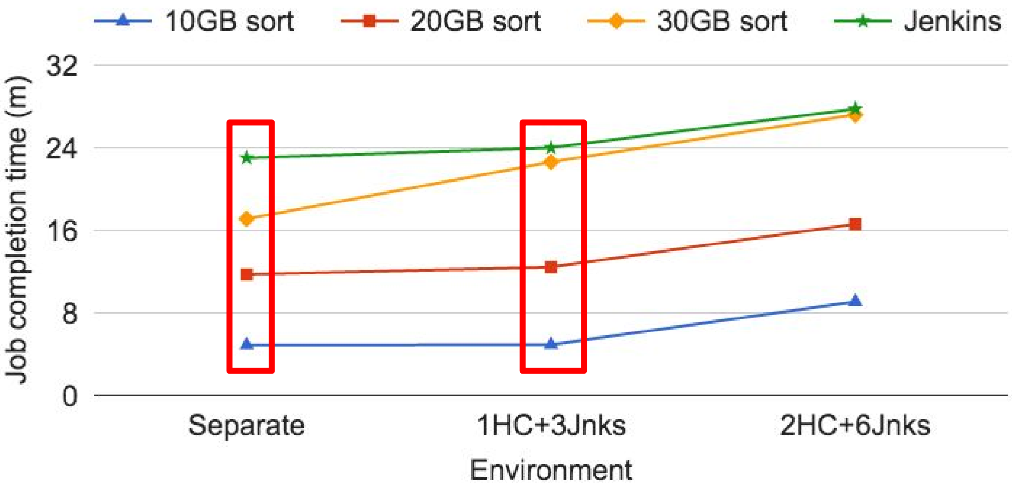


Performance degrades due to storage and network contention (2 clusters)
CPU contention also kicks in (4 clusters)

Enterprise workloads: Hadoop and Jenkins

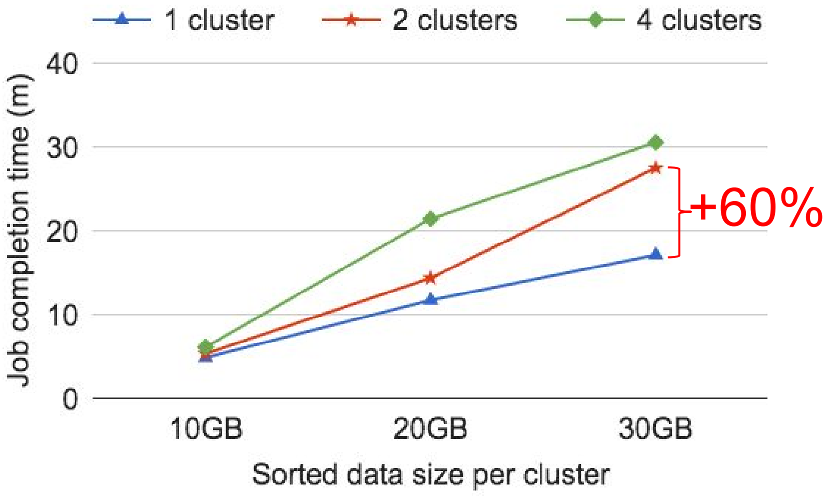
Workload contention: Hadoop and Jenkins

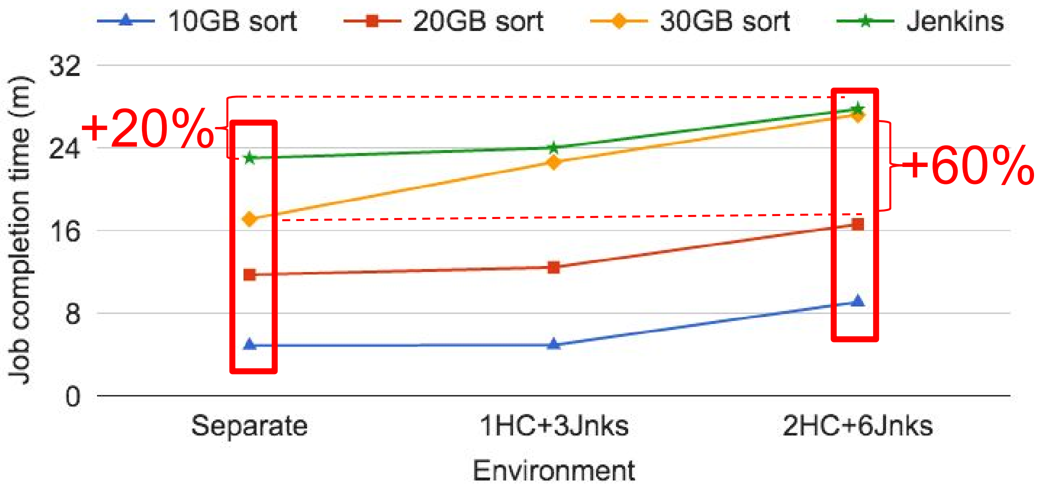




Interference is minimal when workloads stress
different resources at different times

Hadoop only vs. Hadoop+Jenkins



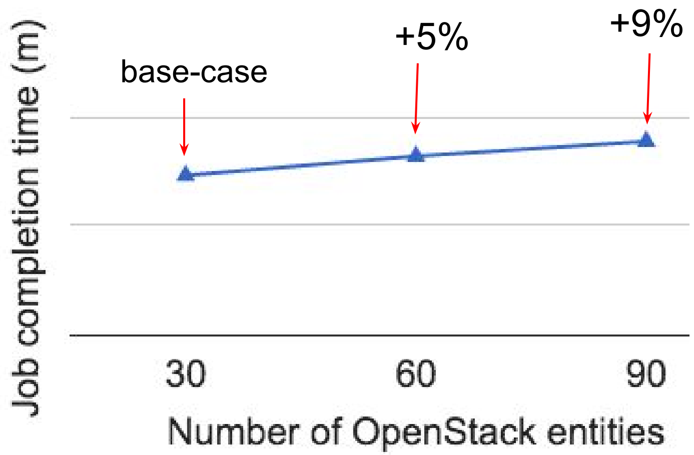


Impact on Jenkins is more than Hadoop.
Hard to predict impact on specific workload.
Need better QoS for isolation!

Control Plane Performance

Workload contention: Hadoop and Jenkins

- Evaluate impact of existing entities to new entity creation time
- Create 30 OpenStack entities
  - Networks
  - Subnets
  - Volumes
  - VMs
- Observation
  - API completion time increases as more objects are created



Provision additional service instances to reduce the impact
Need more visibility across services for each API call

Conclusion

- QoS is needed to reduce contention
  - Network, Storage contention is more critical
  - CPU and memory show less performance hit unless they are over-committed
  - We need control plane scaling
  - We also need control plane QoS to prevent API DoS attacks
- Placement policies can improve the performance drastically
- Private cloud needs to be application-aware

Thank You!

We are hiring: zerostack.com/careers

30 day free trial: www.zerostack.com/tryMyCloud

Learn more: visit booth D3 www.zerostack.com/resources