Ceph Storage on SSD for Container

JANGSEON RYU
NAVER

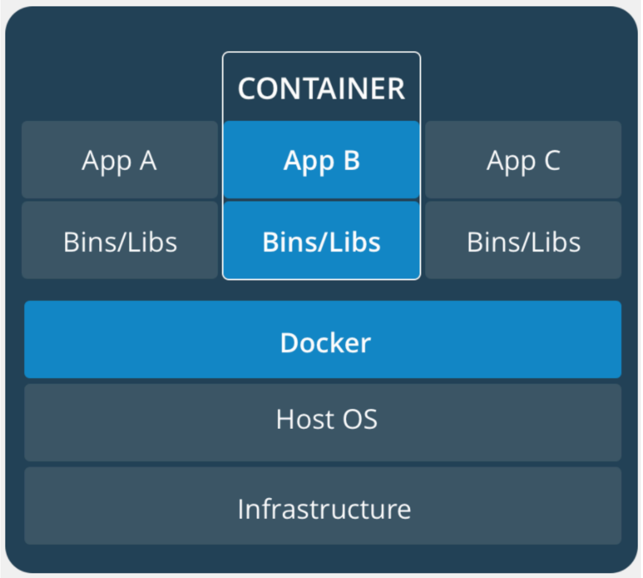
Agenda

- **What is Container?**
- **Persistent Storage**
- **What is Ceph Storage?**
- **Write Process in Ceph**
- **Performance Test**

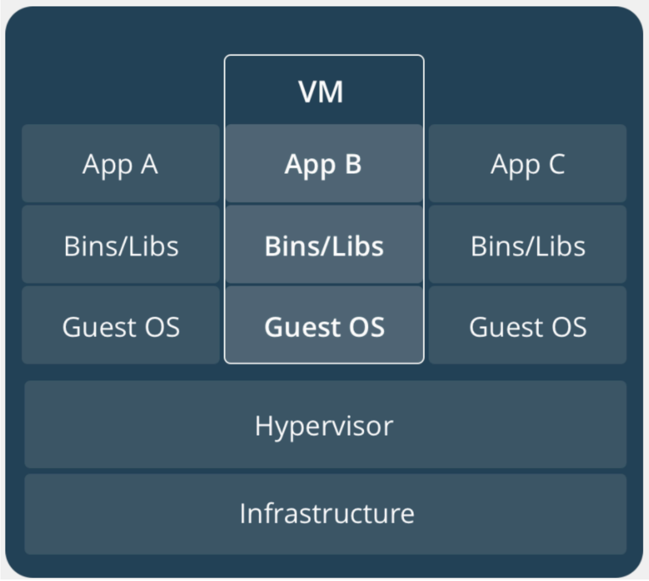
What is Container ?

What is Container?

- **VM vs Container**
- OS Level Virtualization
- Performance
- Run Everywhere – Portable
- Scalable – lightweight
- Environment Consistency
- Easy to manage Image
- Easy to deploy



Docker Container

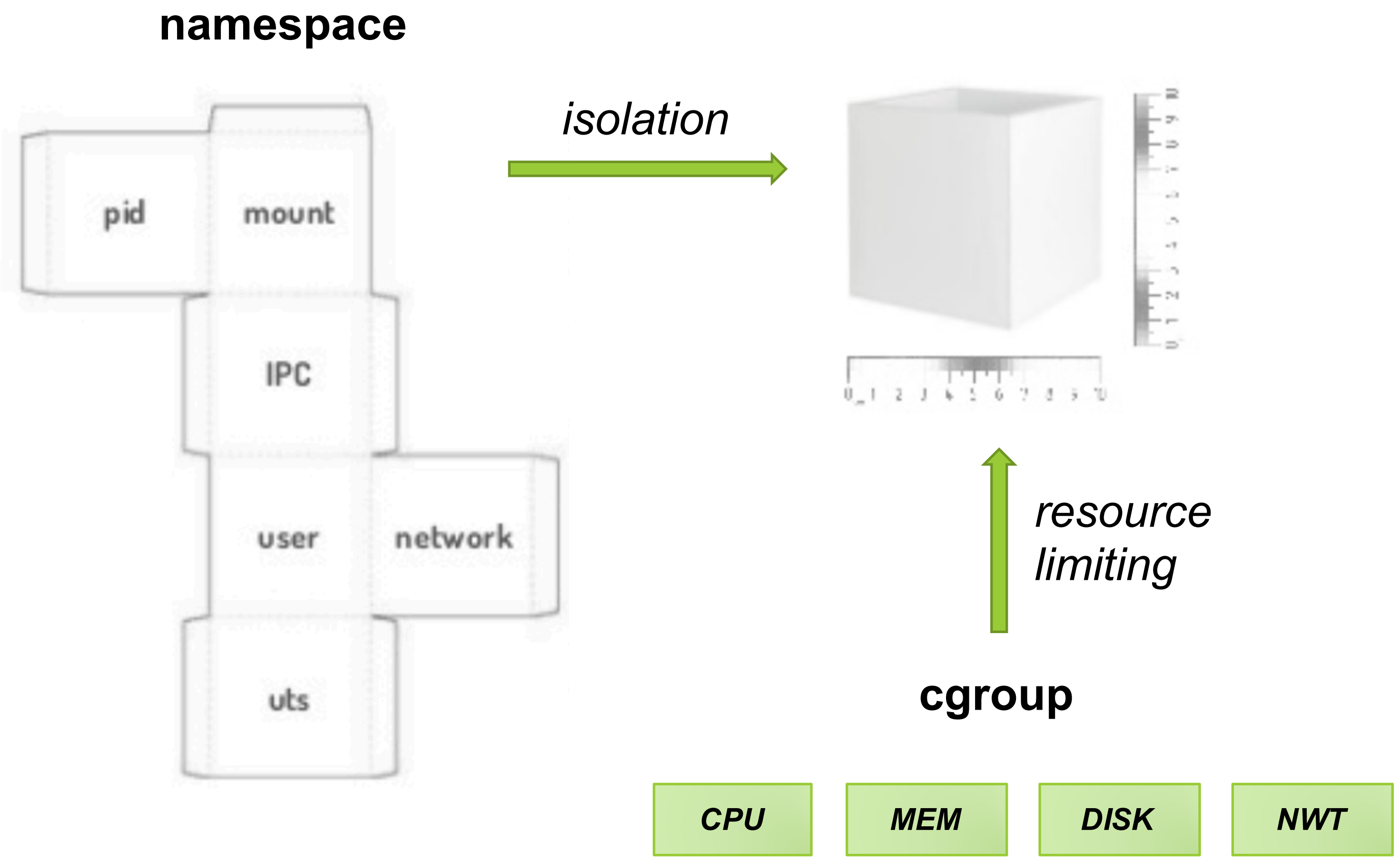


Virtual Machine

Reference : https://www.docker.com/what-container

What is Container?

- VM vs Container
- **OS Level Virtualization**
- Performance
- Run Everywhere – Portable
- Scalable – lightweight
- Environment Consistency
- Easy to manage Image
- Easy to deploy



namespace

isolation

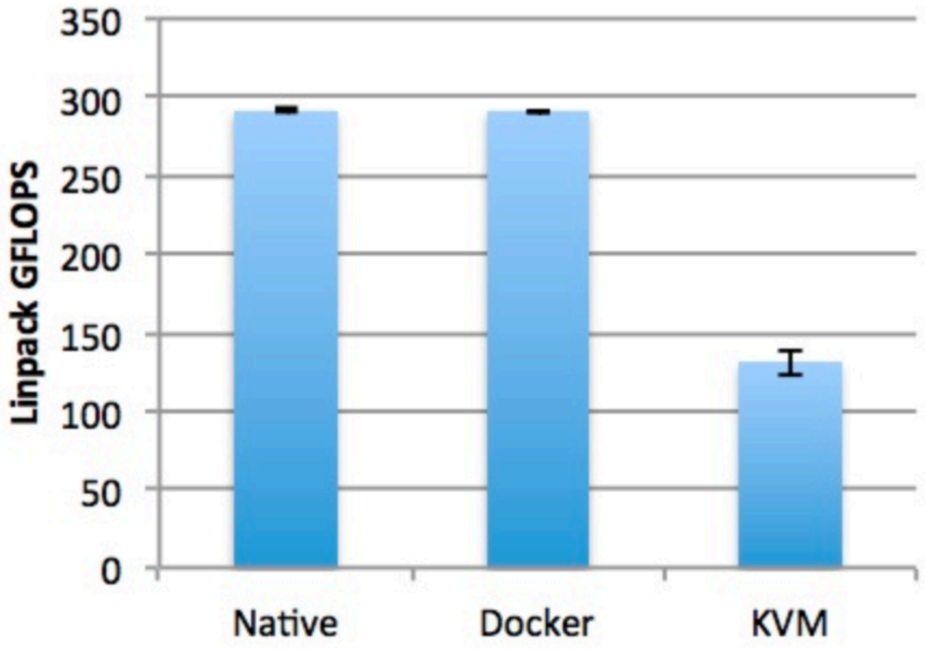
resource limiting

cgroup

Reference : https://www.slideshare.net/PhilEstes/docker-london-container-security

What is Container?

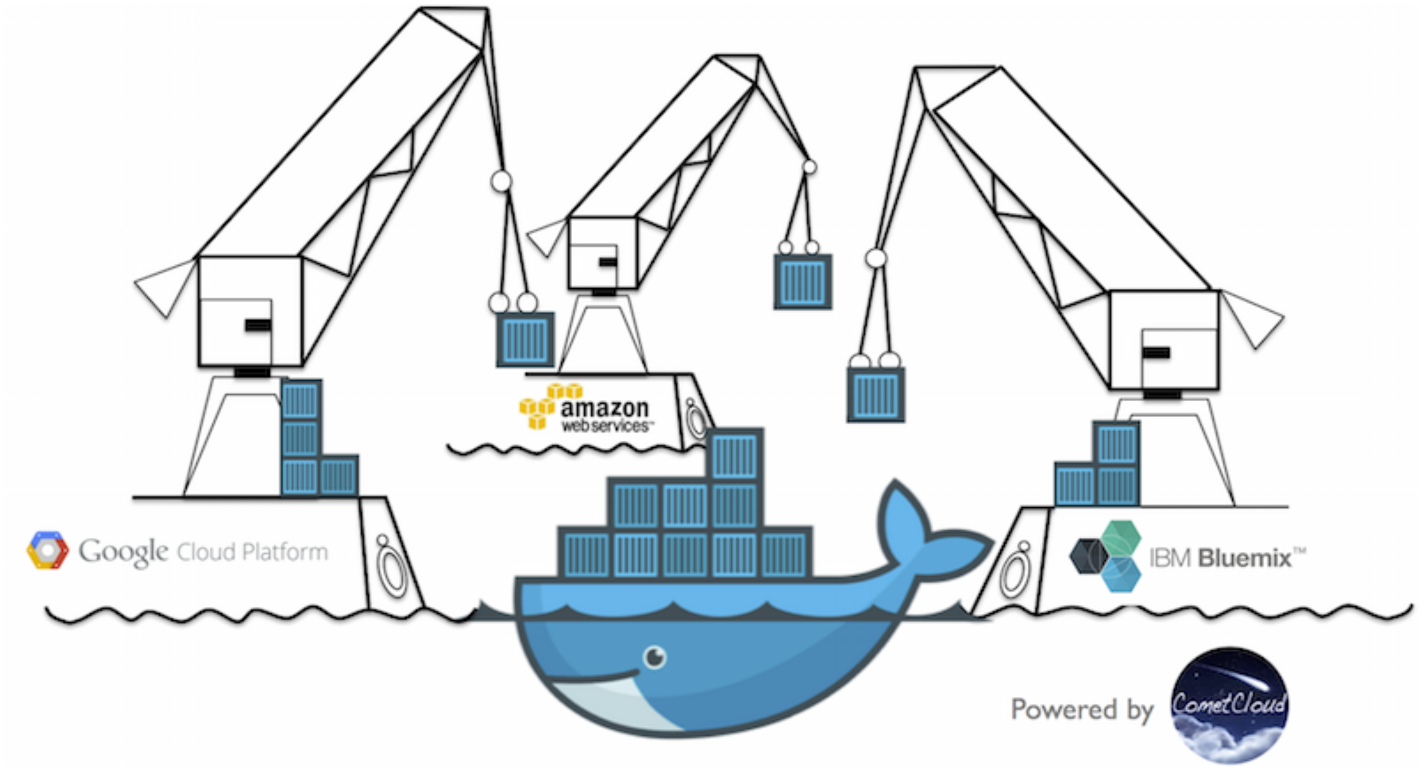
- VM vs Container
- OS Level Virtualization
- **Performance**
- Run Everywhere – Portable
- Scalable – lightweight
- Environment Consistency
- Easy to manage Image
- Easy to deploy



Reference : https://www.theregister.co.uk/2014/08/18/docker_kicks_kvms_butt_in_ibm_tests/

What is Container?

- VM vs Container
- OS Level Virtualization
- Performance
- **Run Everywhere – Portable**
- Scalable – lightweight
- Environment Consistency
- Easy to manage Image
- Easy to deploy

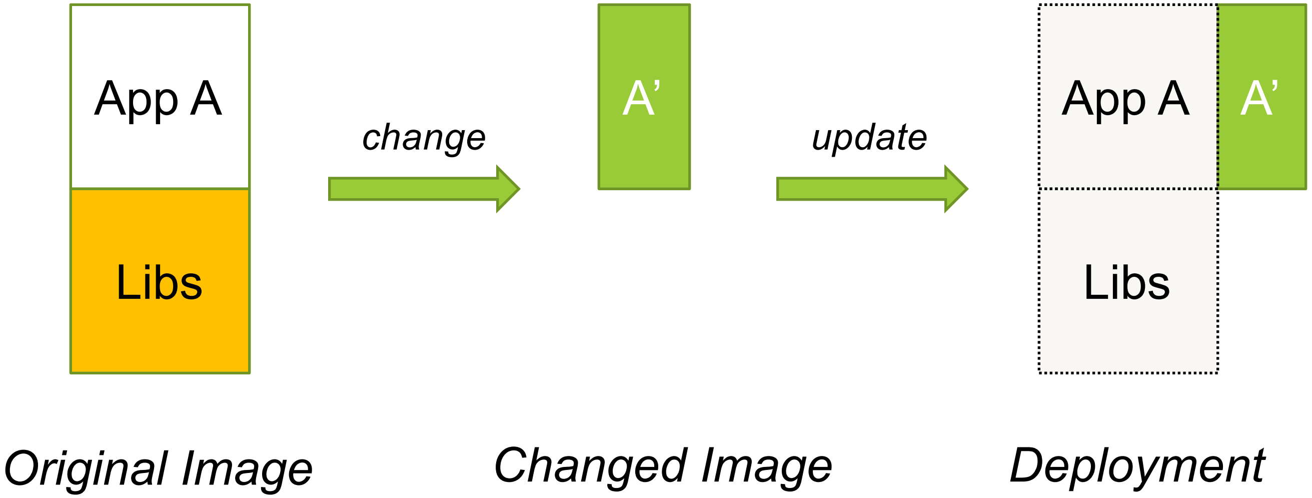


PM / VM / Cloud

Reference : https://www.ibm.com/blogs/bluemix/2015/08/c-ports-docker-containers-across-multiple-clouds-datacenters/

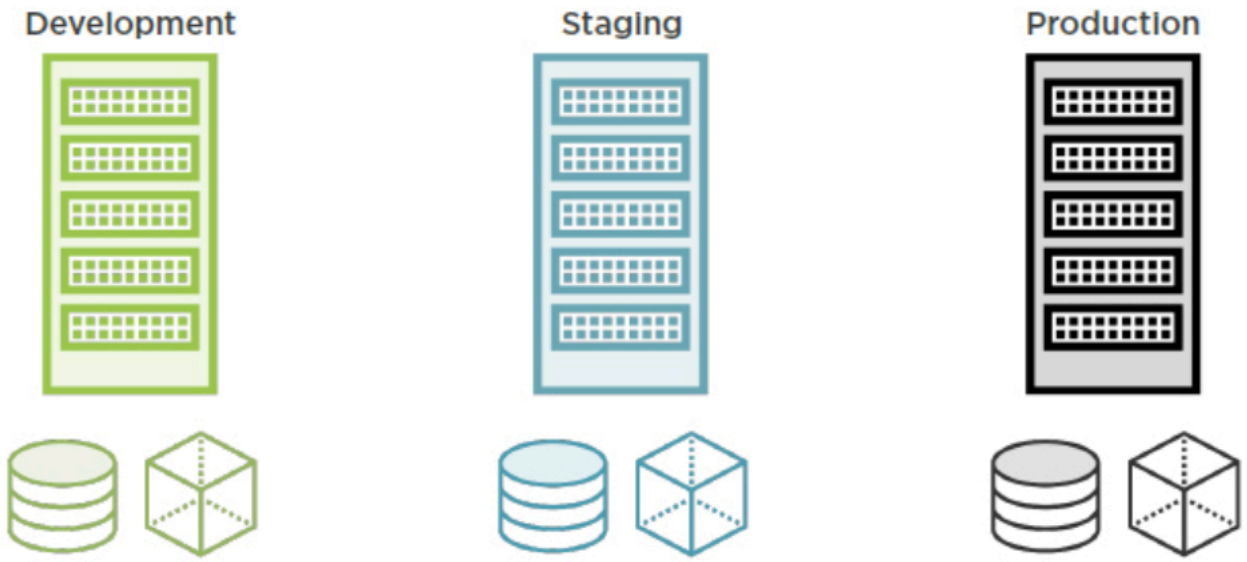
What is Container?

- VM vs Container
- OS Level Virtualization
- Performance
- Run Everywhere – Portable
- **Scalable – lightweight**
- Environment Consistency
- Easy to manage Image
- Easy to deploy



What is Container?

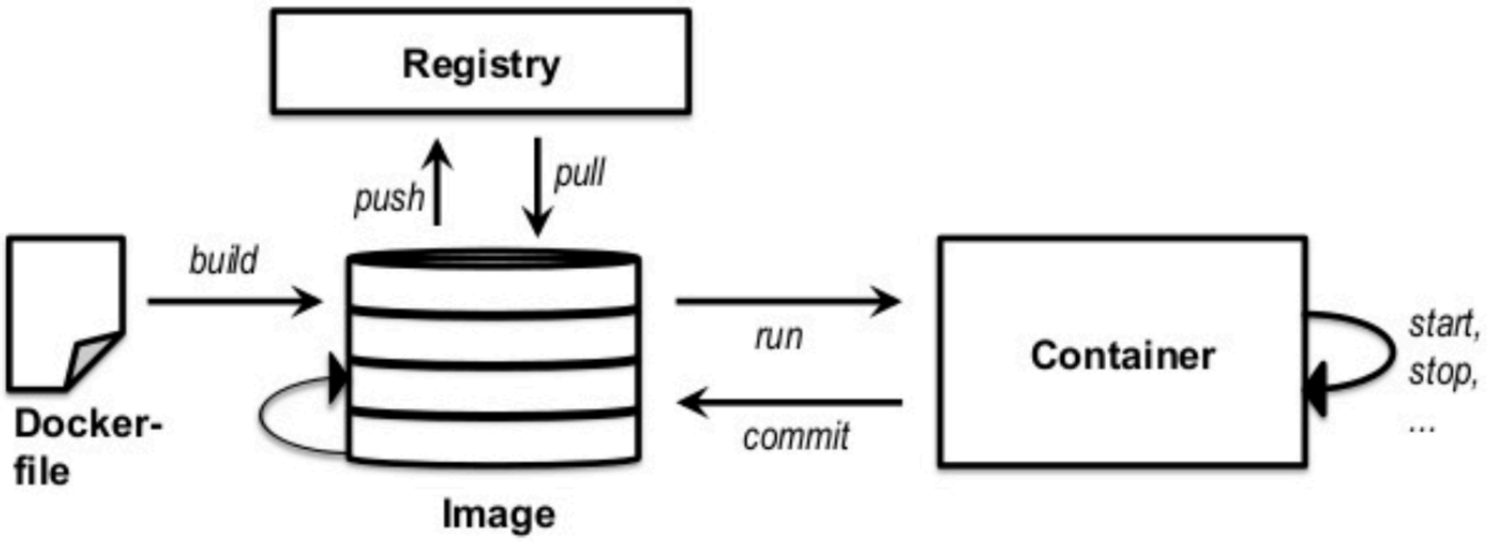
- VM vs Container
- OS Level Virtualization
- Performance
- Run Everywhere – Portable
- Scalable – lightweight
- **Environment Consistency**
- Easy to manage Image
- Easy to deploy



Reference : http://www.devopsschool.com/slides/docker/docker-web-development/index.html#/17

What is Container?

- VM vs Container
- OS Level Virtualization
- Performance
- Run Everywhere – Portable
- Scalable – lightweight
- Environment Consistency
- **Easy to manage Image**
- **Easy to deploy**



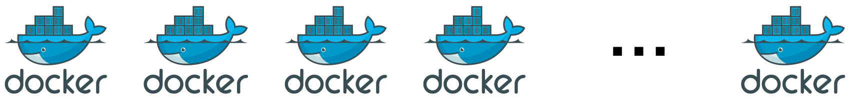
Reference : https://www.slideshare.net/insideHPC/docker-for-hpc-in-a-nutshell

Persistent Storage

Stateless vs Stateful Container

Stateless

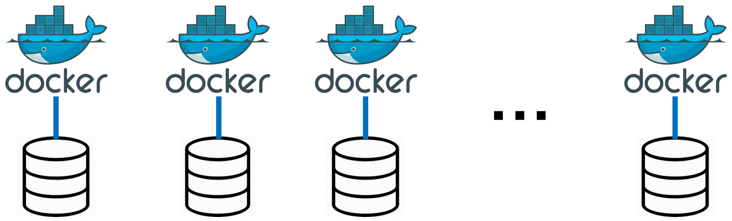
- Nothing to write on disk
- Web (Front-end)
- Easy to scale in/out
- Container is ephemeral
- If delete, will be lost data



Easy to scale out

Stateful

- Needs storage to write
- Database
- Logs
- CI config / repo data
- Secret Keys



Hard to scale out

Ephemeral vs Persistent

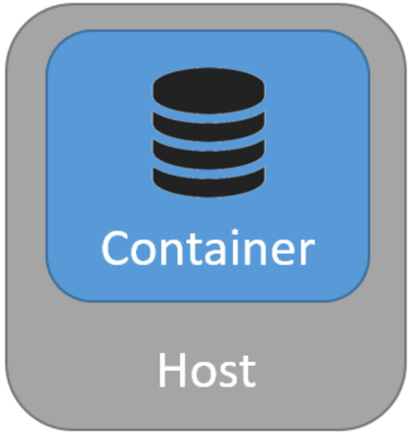
Ephemeral Storage

- Data Lost
- Local Storage
- Stateless Container

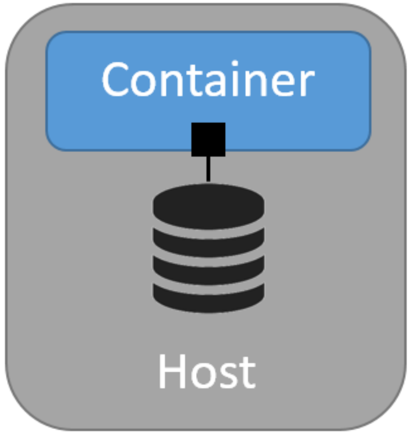
VS

Persistent Storage

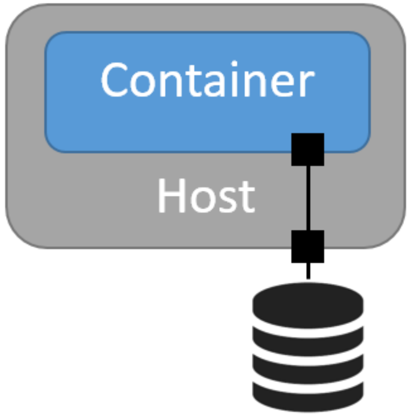
- Data Save
- Network Storage
- Stateful Container



Data in the container
Lost when the container terminates



Data in a Host Volume
Lost when the host terminates

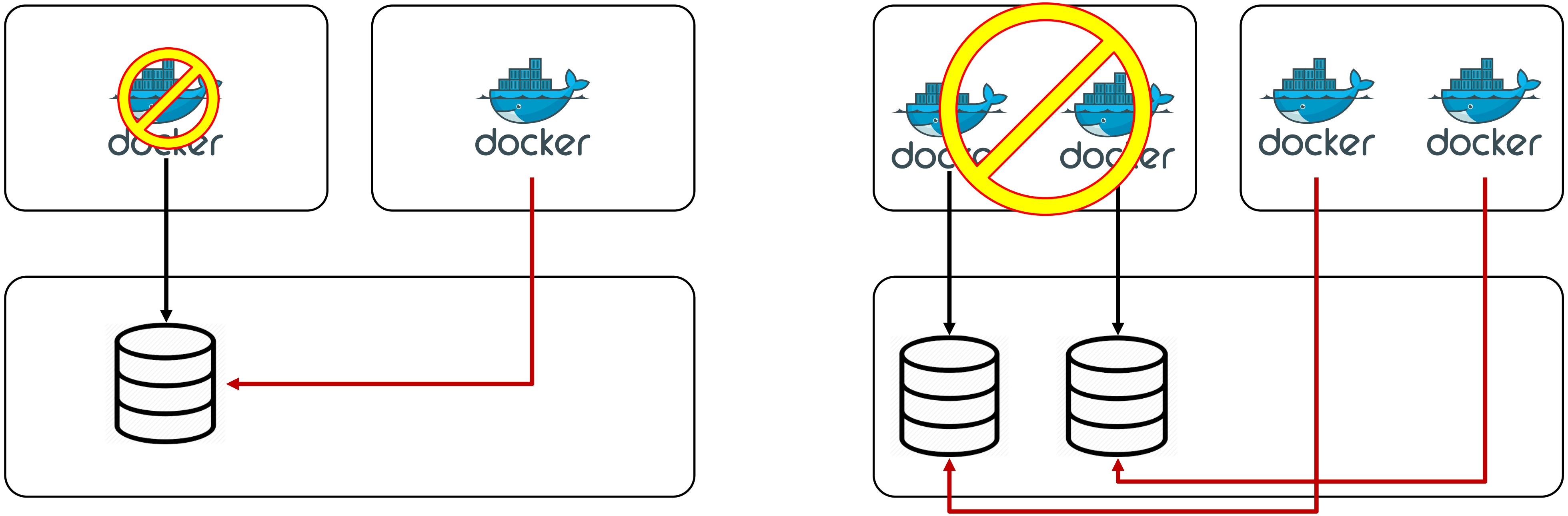


Networked Volume / File System
Independent of host and container

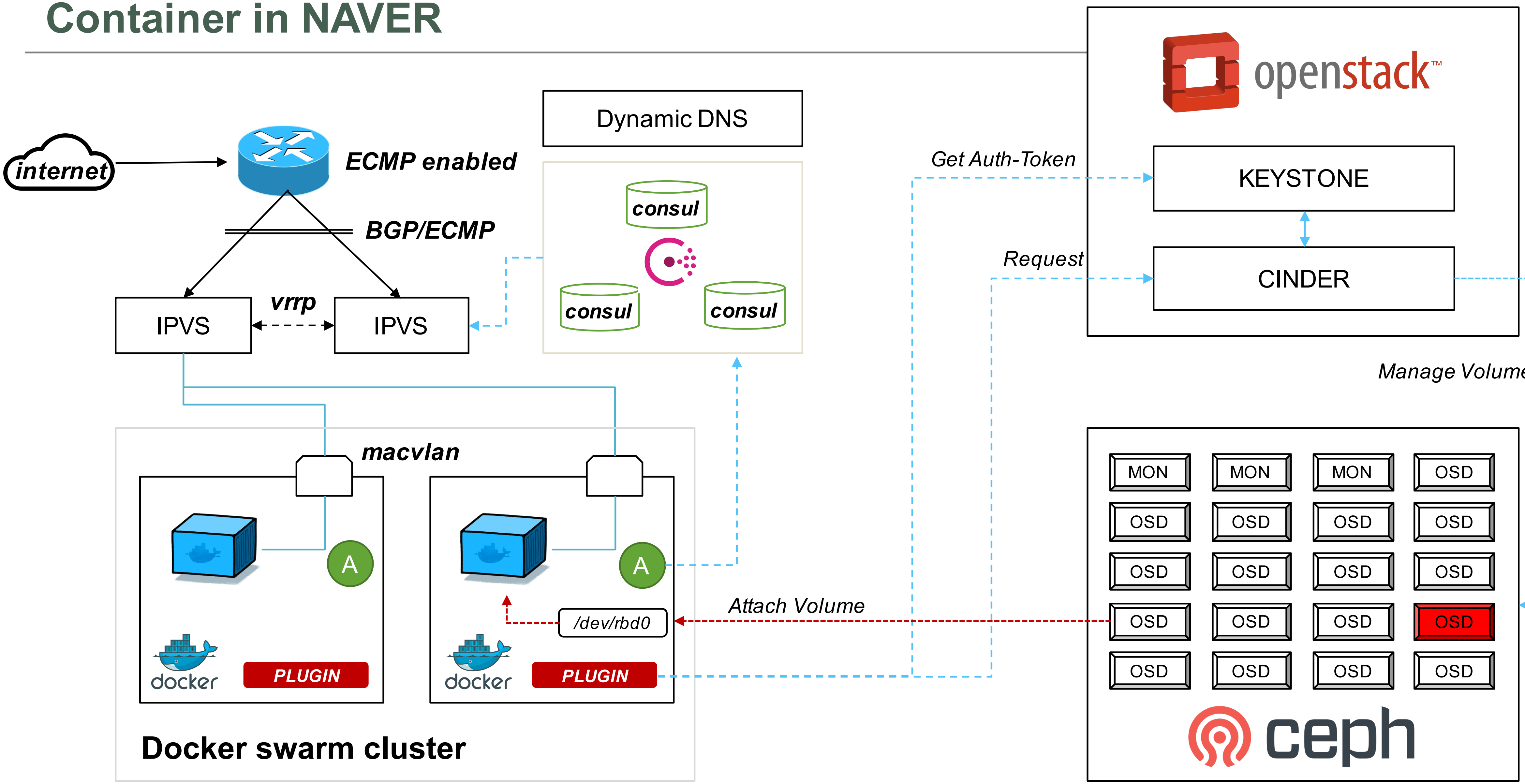
Reference : https://www.infoworld.com/article/3106416/cloud-computing/containerizing-stateful-applications.html

Needs for Persistent Storage

Our mission is to provide **“Persistence Storage Service”** while maintaining “**Agility” & “Automation”** of Docker Container



Container in NAVER



DEVIEW 2016 : https://www.slideshare.net/deview/221-docker-orchestration

OpenStack 2017 : https://www.slideshare.net/JangseonRyu/on-demandblockstoragefordocker-77895159

What is Ceph Storage?

What is Ceph Storage?

- **Open Source Distributed Storage Solution**
- Massive Scalable / Efficient Scale-Out
- Unified Storage
- Runs on commodity hardware
- Integrations into Linux Kernel
  / QEMU/KVM Driver / OpenStack
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placement
- Replicated / Erasure Coding
- Architecture

Vender
NO Lock-in

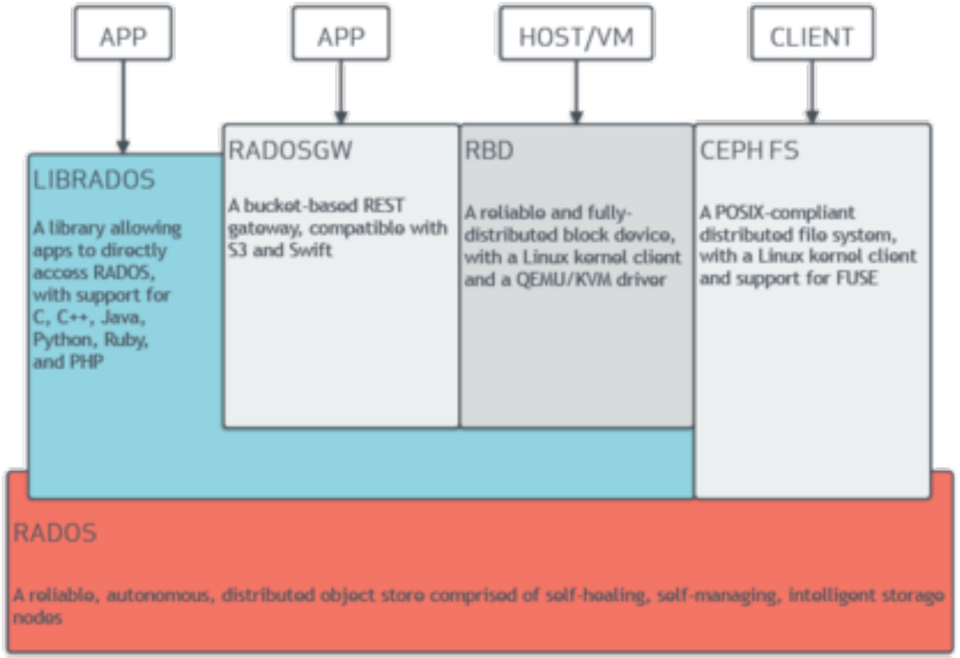
What is Ceph Storage?

- Open Source Distributed Storage Solution
- **Massive Scalable / Efficient Scale-Out**
- Unified Storage
- Runs on commodity hardware
- Integrations into Linux Kernel
  / QEMU/KVM Driver / OpenStack
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placement
- Replicated / Erasure Coding
- Architecture

Up-to 16 Exa

What is Ceph Storage?

- Open Source Distributed Storage Solution
- Massive Scalable / Efficient Scale-Out
- **Unified Storage**
- Runs on commodity hardware
- Integrations into Linux Kernel
  / QEMU/KVM Driver / OpenStack
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placement
- Replicated / Erasure Coding
- Architecture



LIBRADOS

A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP

RADOSGW

A bucket-based REST gateway, compatible with S3 and Swift

RBD

A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver

CEPH FS

A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE

RADOS

A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

Reference : https://en.wikipedia.org/wiki/Ceph_(software)

What is Ceph Storage?

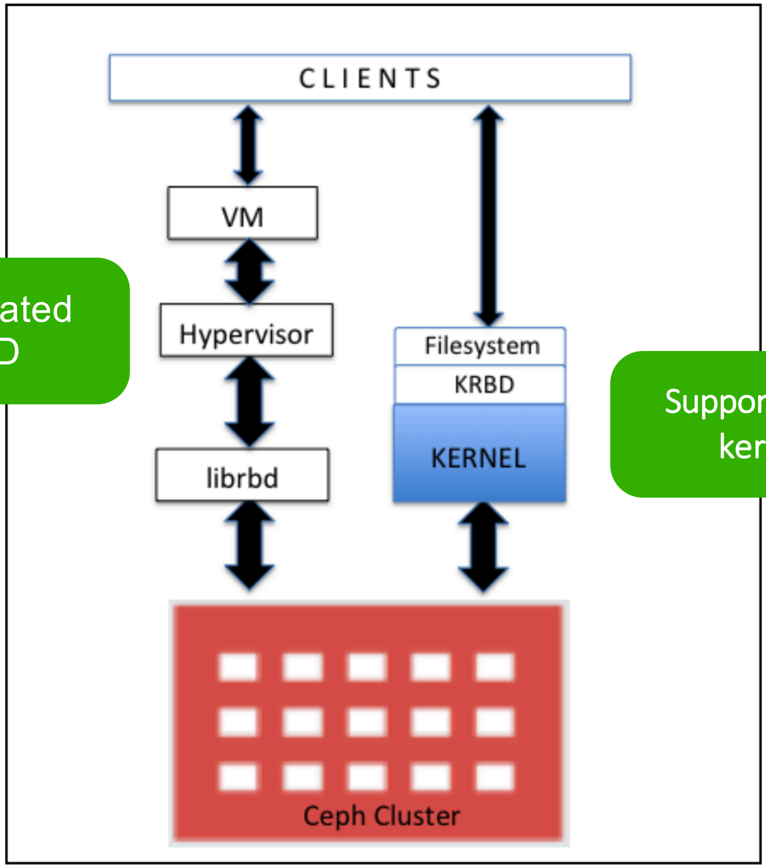
- Open Source Distributed Storage Solution
- Massive Scalable / Efficient Scale-Out
- Unified Storage
- **Runs on commodity hardware**
- Integrations into Linux Kernel
  / QEMU/KVM Driver / OpenStack
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placement
- Replicated / Erasure Coding
- Architecture

Low TCO

What is Ceph Storage?

- Open Source Distributed Storage Solution
- Massive Scalable / Efficient Scale-Out
- Unified Storage
- Runs on commodity hardware
- **Integrations into Linux Kernel / QEMU/KVM Driver / OpenStack**
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placement
- Replicated / Erasure Coding
- Architecture

openstack



KVM integrated with RBD

Supported above kernel 2.6

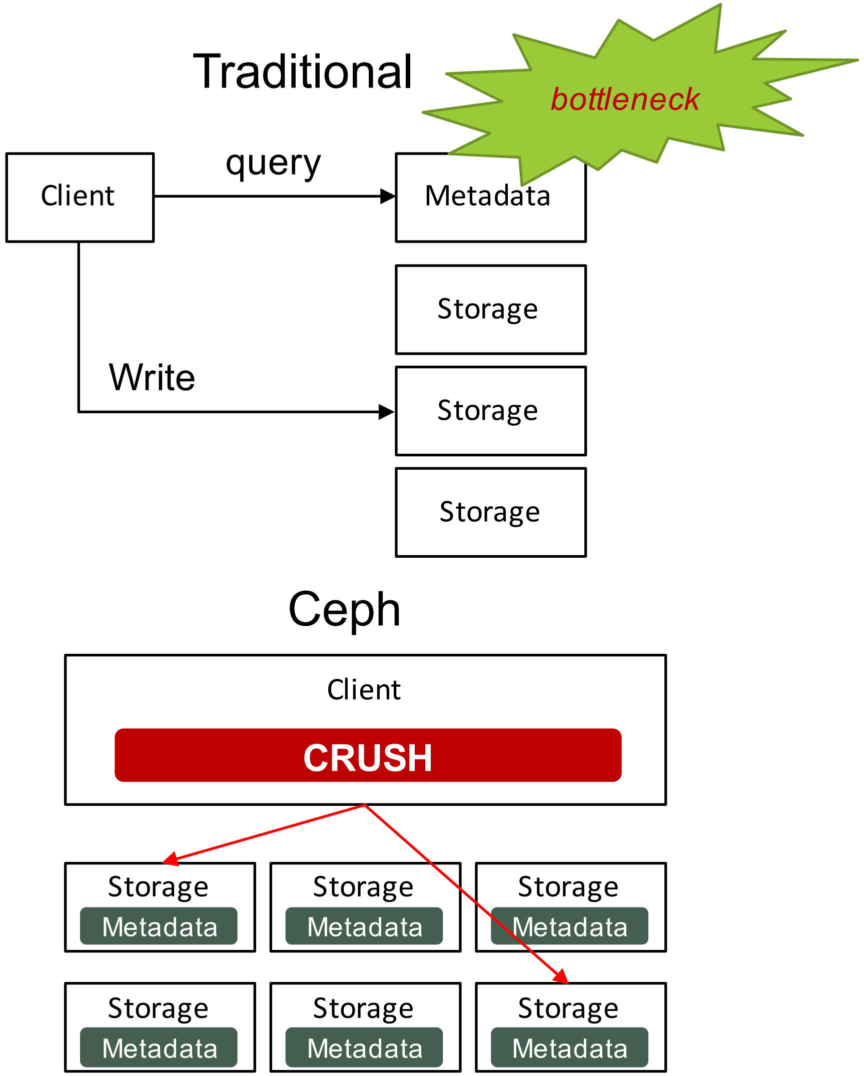
What is Ceph Storage?

- Open Source Distributed Storage Solution
- Massive Scalable / Efficient Scale-Out
- Unified Storage
- Runs on commodity hardware
- Integrations into Linux Kernel
  / QEMU/KVM Driver / OpenStack
- **Self-managing / Self-healing**
- **Peer-to-Peer Storage Nodes**
- **RESTful API**
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placement
- Replicated / Erasure Coding
- Architecture

What is Ceph Storage?

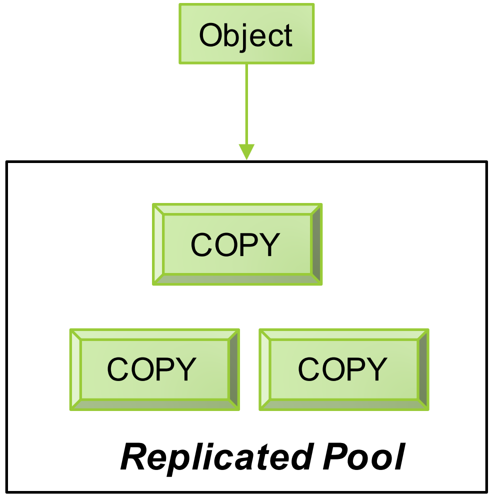
- Open Source Distributed Storage Solution
- Massive Scalable / Efficient Scale-Out
- Unified Storage
- Runs on commodity hardware
- Integrations into Linux Kernel

  / QEMU/KVM Driver / OpenStack
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- **No metadata bottleneck (no lookup)**
- **CRUSH algorithm determines data placement**
- Replicated / Erasure Coding
- Architecture

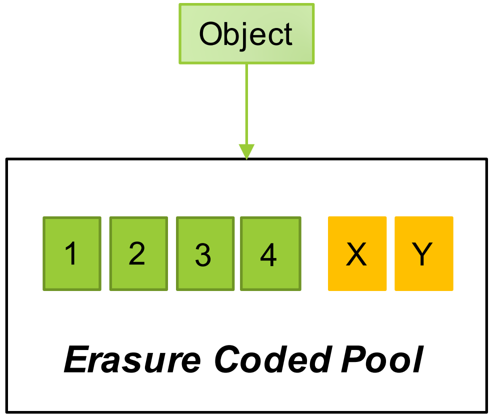


What is Ceph Storage?

- Open Source Distributed Storage Solution
- Massive Scalable / Efficient Scale-Out
- Unified Storage
- Runs on commodity hardware
- Integrations into Linux Kernel
  / QEMU/KVM Driver / OpenStack
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placement
- **Replicated / Erasure Coding**
- Architecture



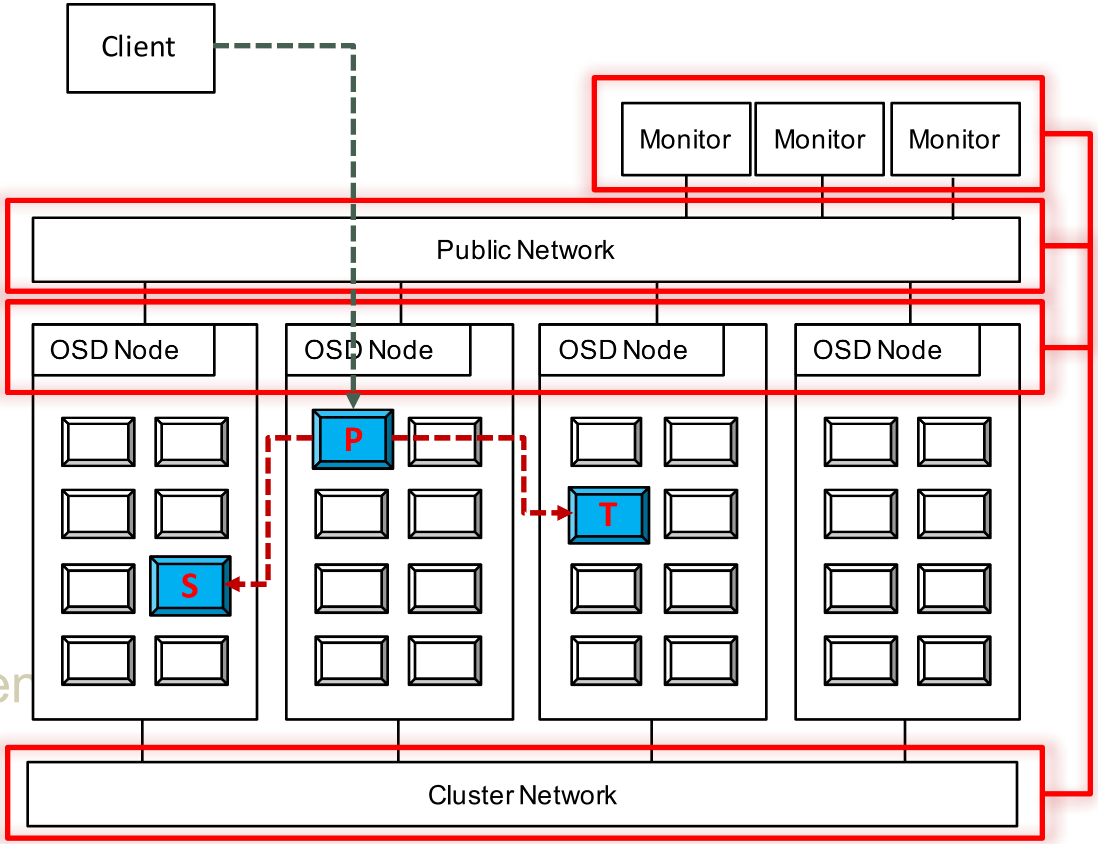
- Very high durability
- 200 % overhead
- Quick recovery



- Cost-Effective
- 50 % overhead
- Expensive Recovery

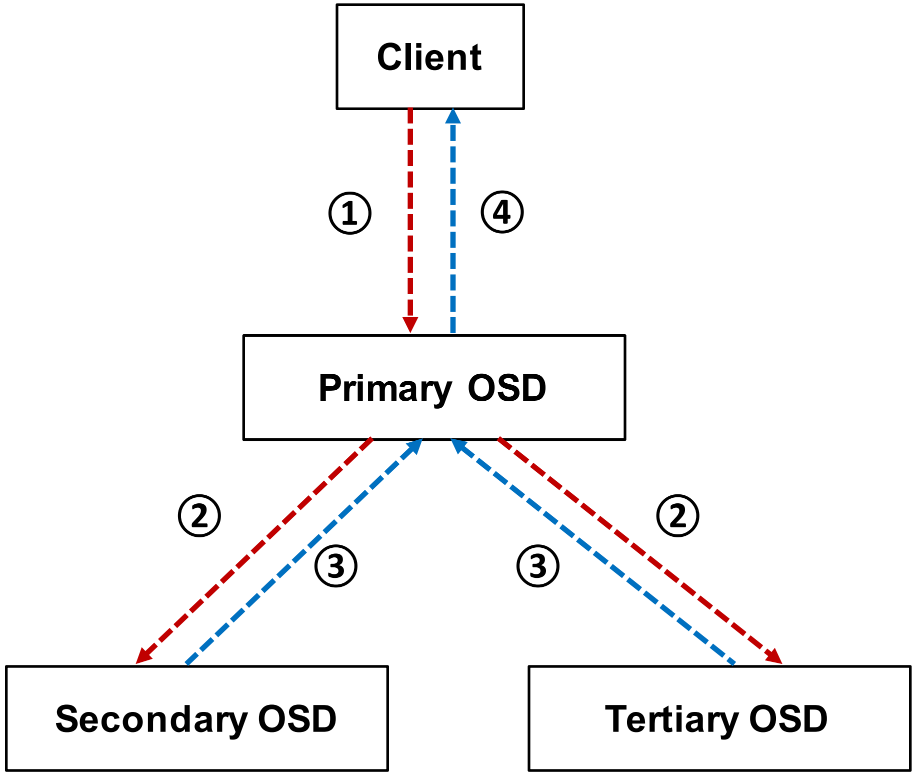
What is Ceph Storage?

- Open Source Distributed Storage Solution
- Massive Scalable / Efficient Scale-Out
- Unified Storage
- Runs on commodity hardware
- Integrations into Linux Kernel
  / QEMU/KVM Driver / OpenStack
- Self-managing / Self-healing
- Peer-to-Peer Storage Nodes
- RESTful API
- No metadata bottleneck (no lookup)
- CRUSH algorithm determines data placemen
- Replicated / Erasure Coding
- **Architecture**



Write Process in Ceph

Write Flow



- ■ Strong Consistency
- ■ CAP Theorem : CP system
  (Consistency / network Partition)

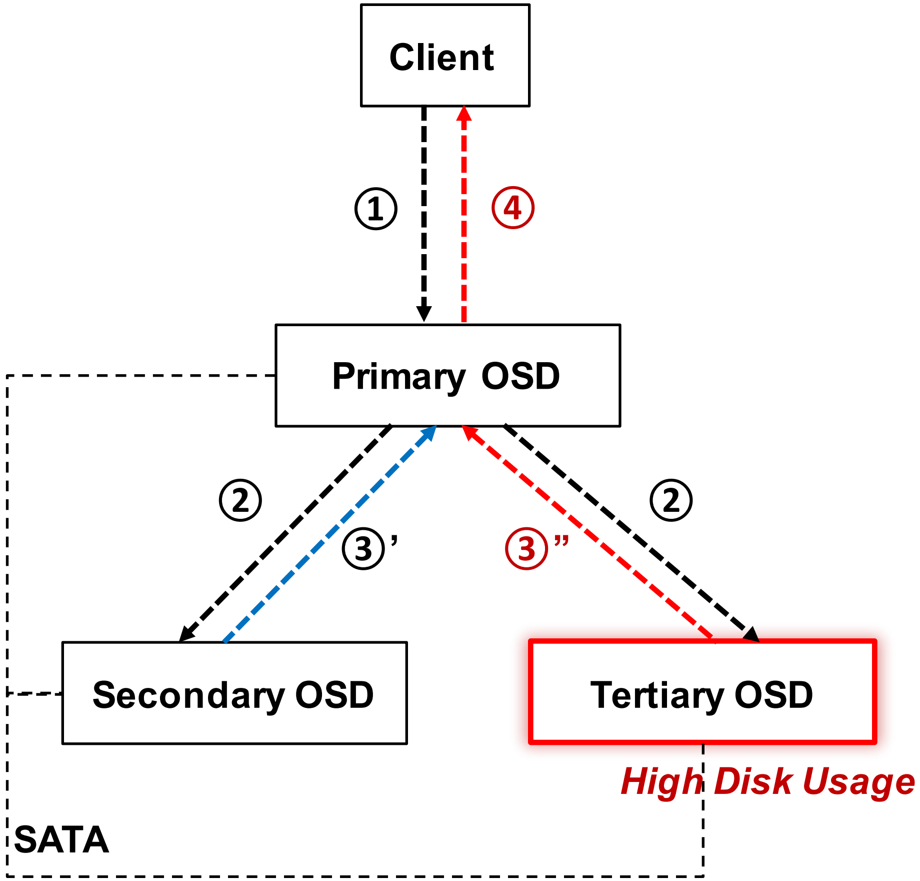
① Client writes data to primary osd.

② Primary OSD sends data to replica OSDs, write data to local disk.

③ Replica OSDs write data to local disk, signal completion to primary.

④ Primary OSD signals completion to client.

Slow Requests



■ All of Data Disks are SATA.
■ 7.2k rpm SATA : ~ 75 ~ 100 IOPS

① Client writes data to primary osd.

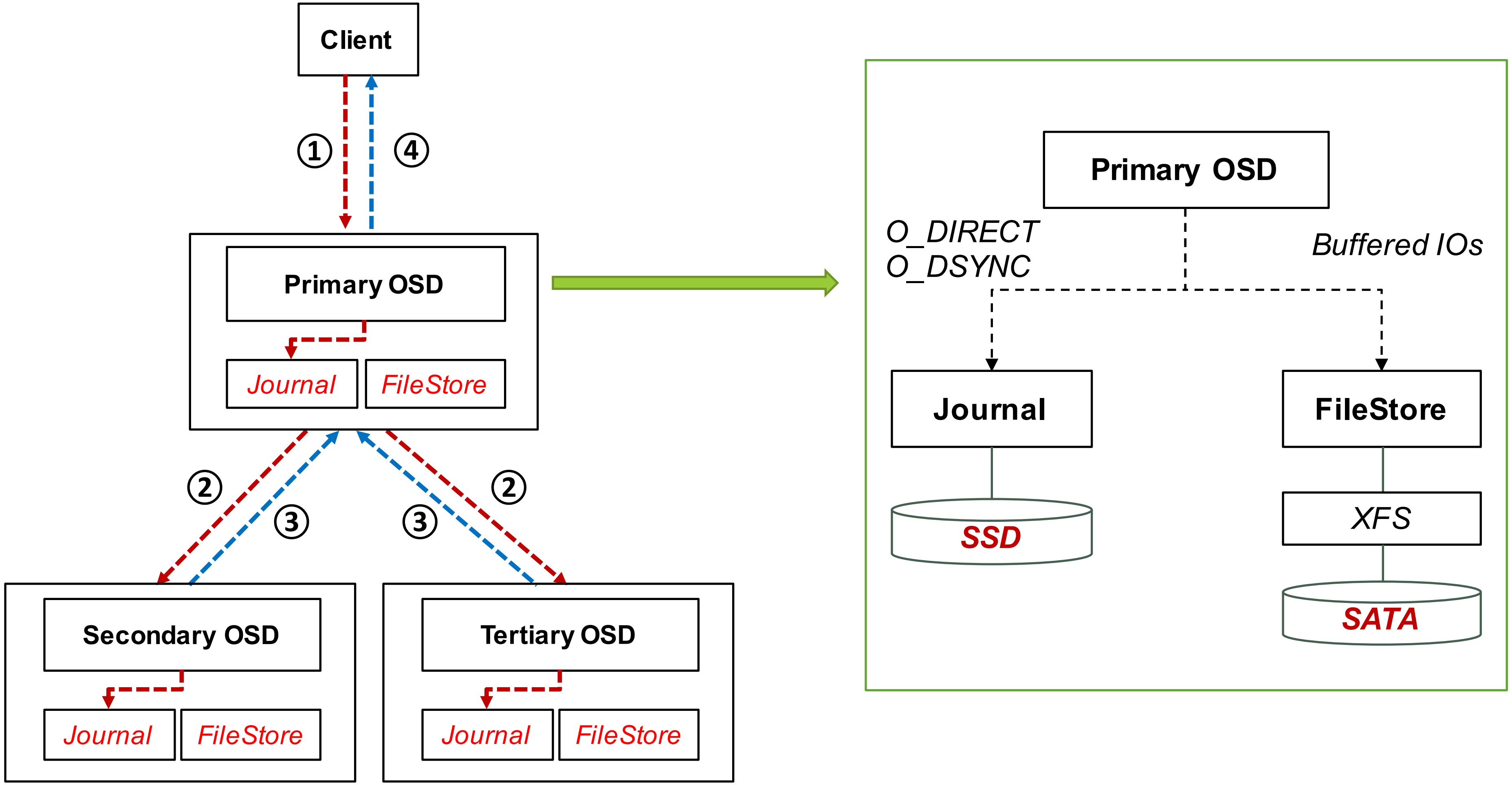
② Primary OSD sends data to replica OSDs,
write data to local disk.

③' Write data to local disk,
Send ack to primary.

③" Slow write data to local disk,
Send ack to primary.

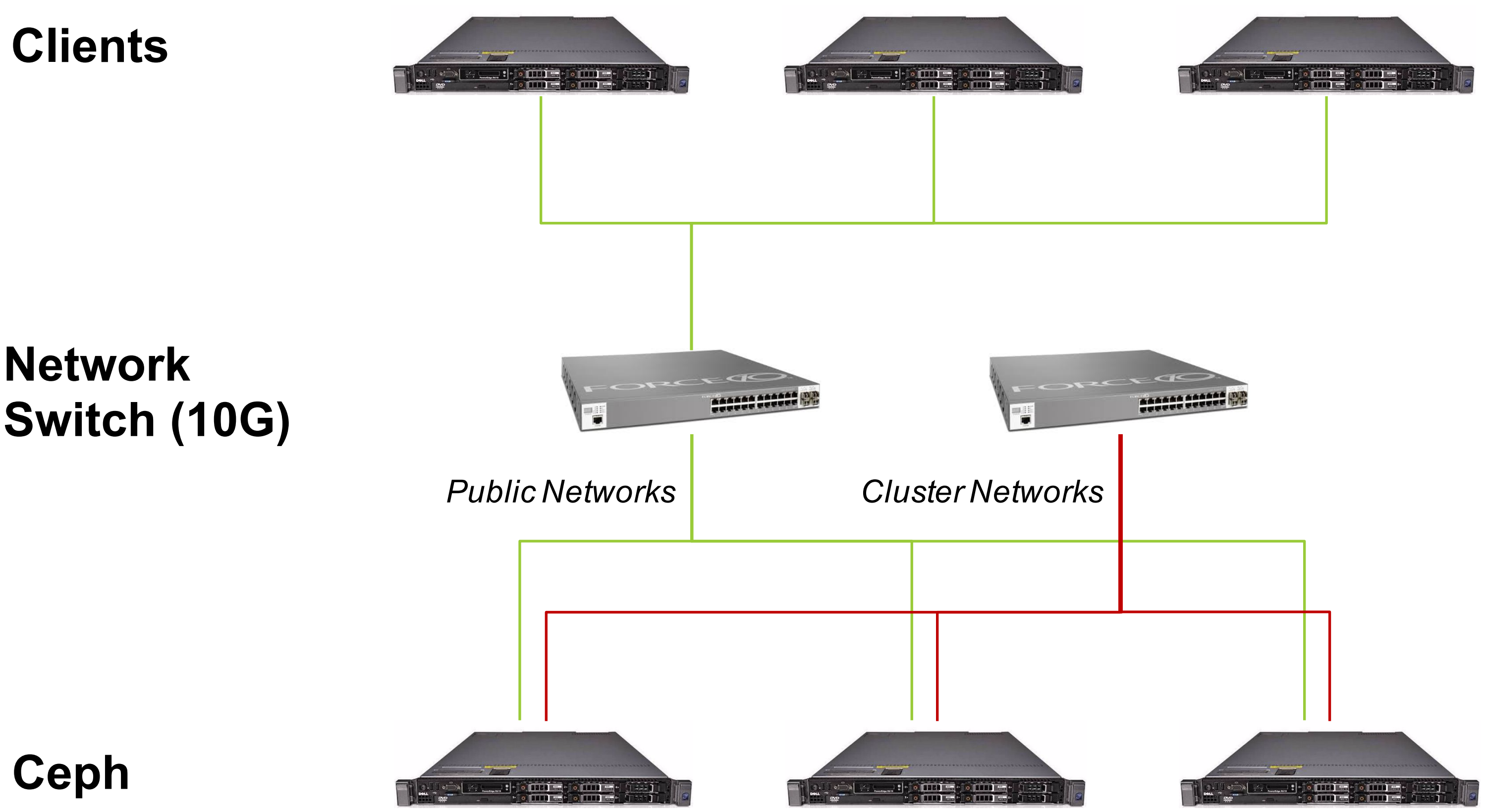
④ Slow send ack to client.

Journal on SSD



Performance Test

Test Environment



KRBD

- /dev/rbd0
- mkfs / mount
- fio

Ceph

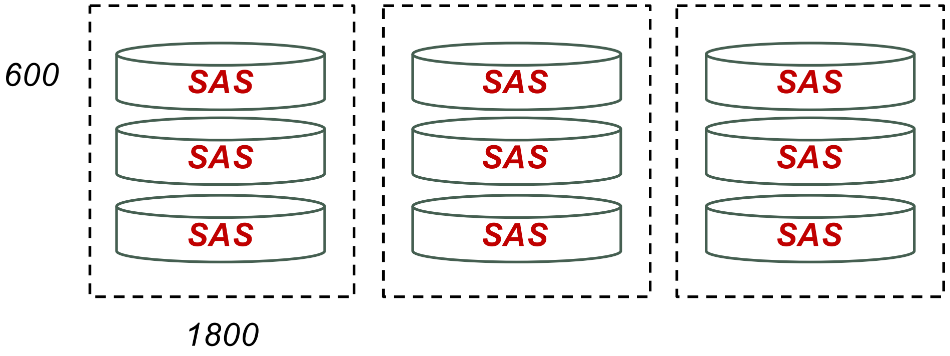
- FileStore
- Luminous (12.2.0)

Server

- Intel® Xeon CPU L5640 2.27 GHz
- 16 GB Memory
- 480GB SAS 10K x 5
- 480GB SAS SSD x 1
- Centos 7

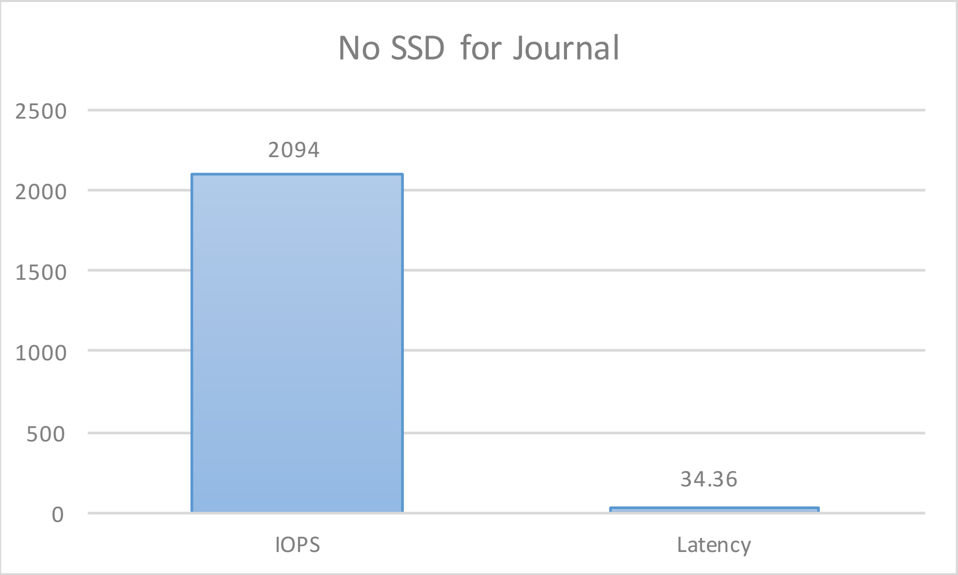
Case #1. Only Use SAS DISK

Expect



SAS : 10,000 rpm / 600 IOPS
*Per Node : SAS * 3 = 600 * 3 = 1,800 IOPS*
*Total : Node * 3 = 1,800 * 3 = 5,400 IOPS*
Replicas 3 = 5,400 / 3 = ***1,800 IOPS***

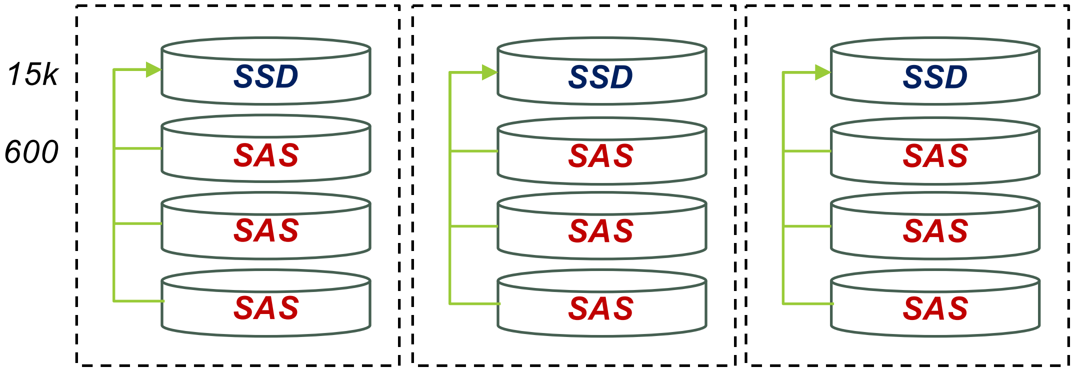
Result : 4K Rand Write



2094 IOPS/s

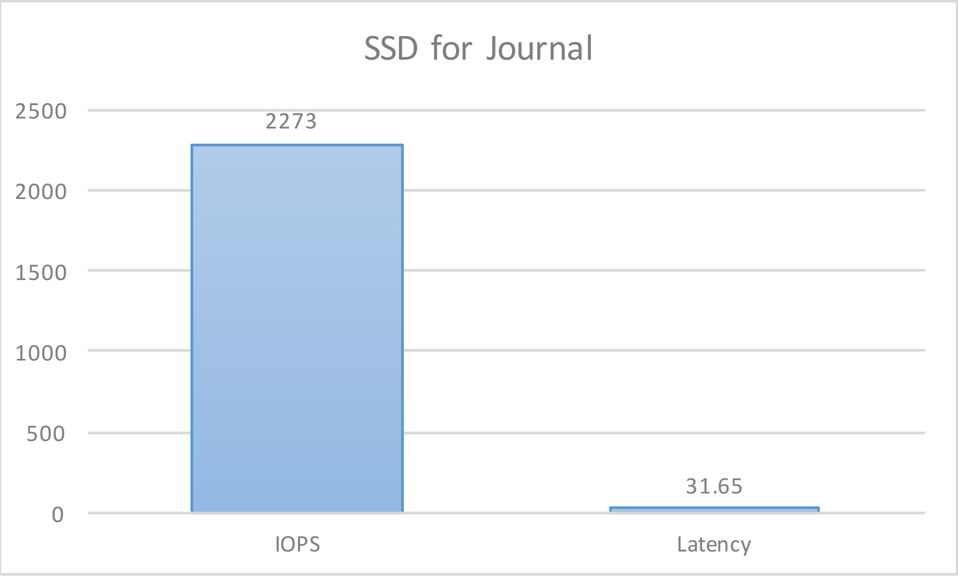
Case #2. Use SSD for Journal

Expect



SSD : 15,000 IOPS
*Total : Node * 3 = 15,000* 3 = 45,000 IOPS*
Replicas 3 = 45,000 / 3 = ***15,000 IOPS***

Result : 4K Rand Write

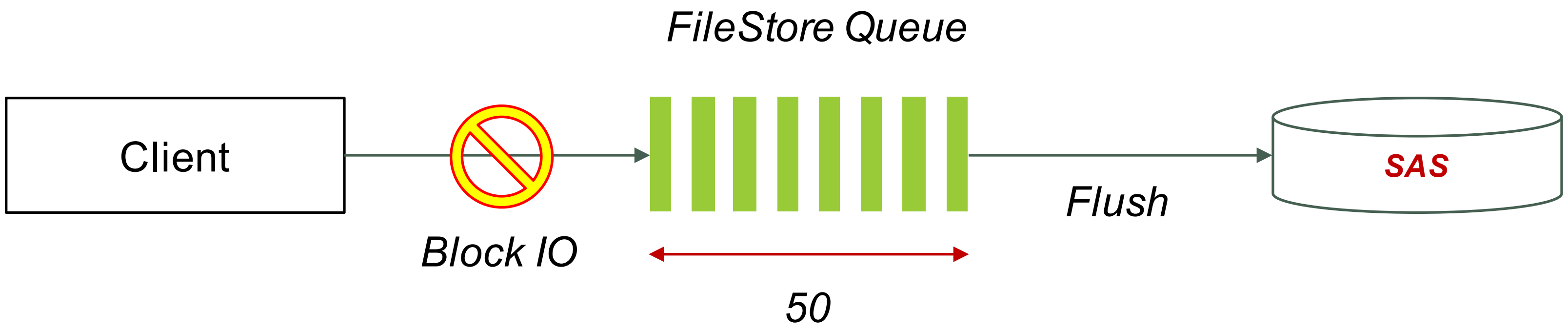


2094 → 2273

Analysis

```
# ceph --admin-daemon /var/run/ceph/ceph-osd.2.asok  perf dump | grep -A 15 throttle-filestore_ops
"throttle-filestore_ops": {
"val": 50,        ← limitation
"max": 50,
…

# ceph --admin-daemon /var/run/ceph/ceph-osd.2.asok  config show | grep queue_max
…
"filestore_queue_max_bytes": "104857600",
"filestore_queue_max_ops": ”50",      ← default value
…
```

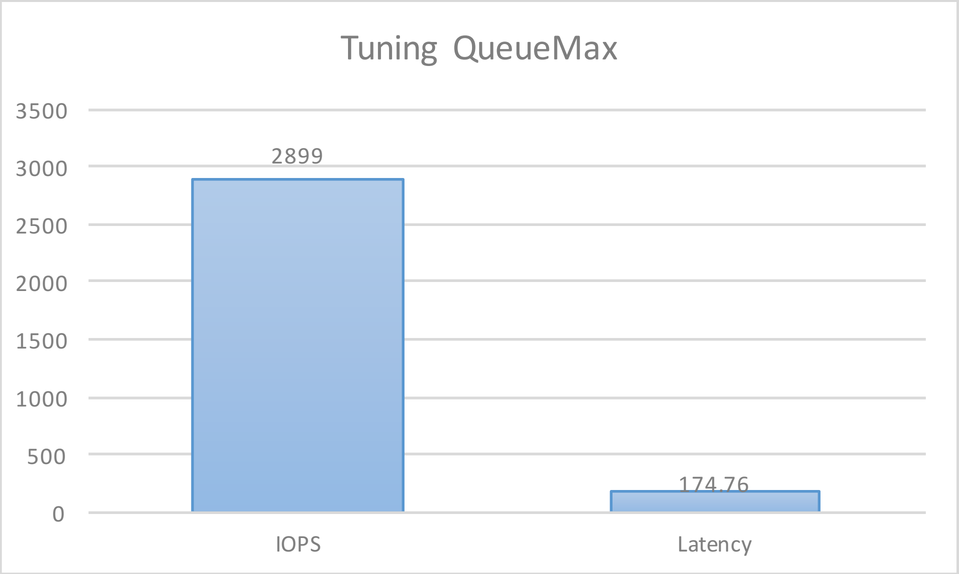


Case #3. Tuning FileStore

Tuning

filestore_queue_max_ops = 500000
filestore_queue_max_bytes = 42949672960
journal_max_write_bytes = 42949672960
journal_max_write_entries = 5000000

Result

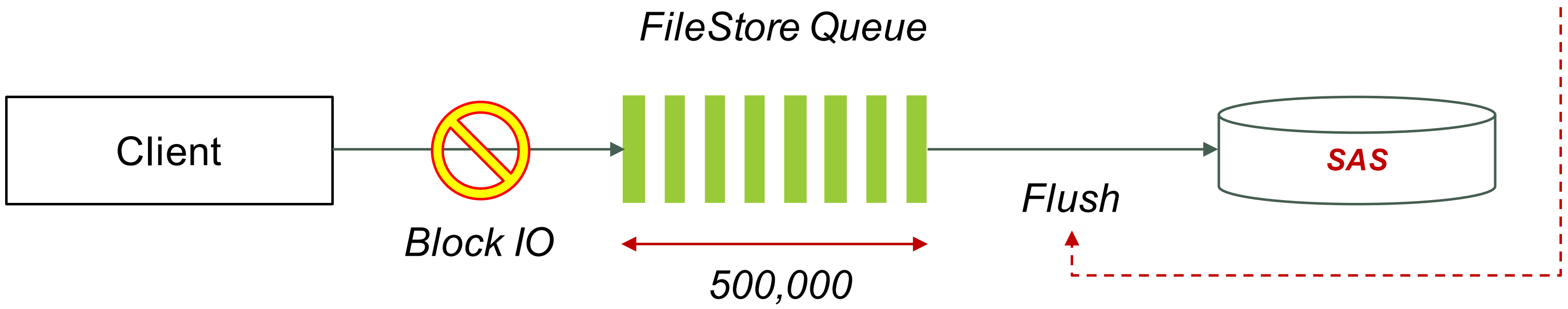


2094 → 2899 (38%)

Analysis

```
# ceph --admin-daemon /var/run/ceph/ceph-osd.2.asok perf dump | grep -A6 WBThrottle
"WBThrottle": {
"bytes_dirtied": 21049344,
"bytes_wb": 197390336,
"ios_dirtied": 5017,    ← limitation
"ios_wb": 40920,
"inodes_dirtied": 1195,
"inodes_wb": 20602
```

```
# ceph --admin-daemon /var/run/ceph/ceph-osd.2.asok config show | grep wbthrottle_xfs
"filestore_wbthrottle_xfs_bytes_hard_limit": "419430400",
"filestore_wbthrottle_xfs_bytes_start_flusher": "41943040",
"filestore_wbthrottle_xfs_inodes_hard_limit": "5000",
"filestore_wbthrottle_xfs_inodes_start_flusher": "500",
”filestore_wbthrottle_xfs_ios_hard_limit": "5000",
"filestore_wbthrottle_xfs_ios_start_flusher": "500",
```

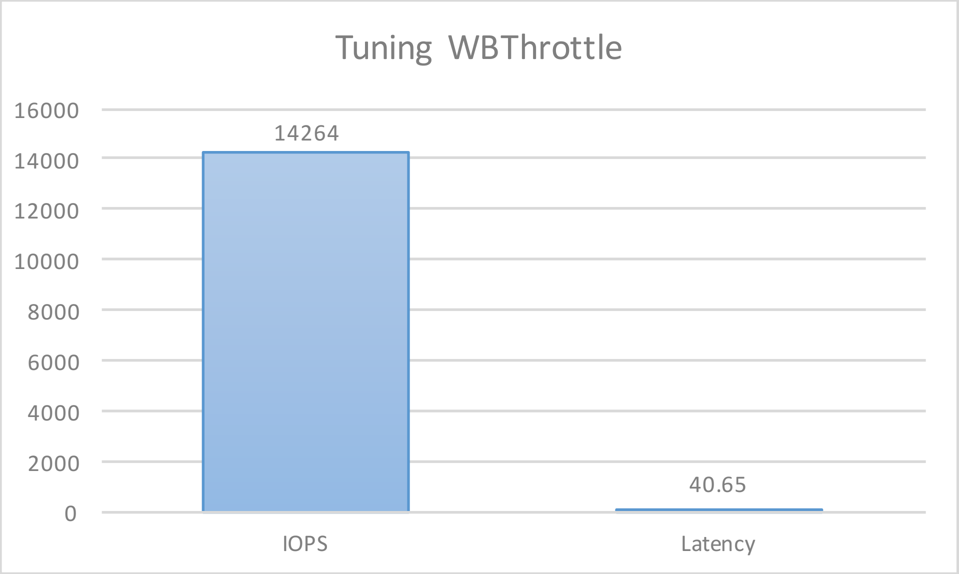


Case #4. Tuning WBThrottle

Tuning

```
 "filestore_wbthrottle_enable": "false",
or
 "filestore_wbthrottle_xfs_bytes_hard_limit": "4194304000",
"filestore_wbthrottle_xfs_bytes_start_flusher": "419430400",
"filestore_wbthrottle_xfs_inodes_hard_limit": "500000",
"filestore_wbthrottle_xfs_inodes_start_flusher": "5000",
"filestore_wbthrottle_xfs_ios_hard_limit": "500000",
"filestore_wbthrottle_xfs_ios_start_flusher": "5000",
```

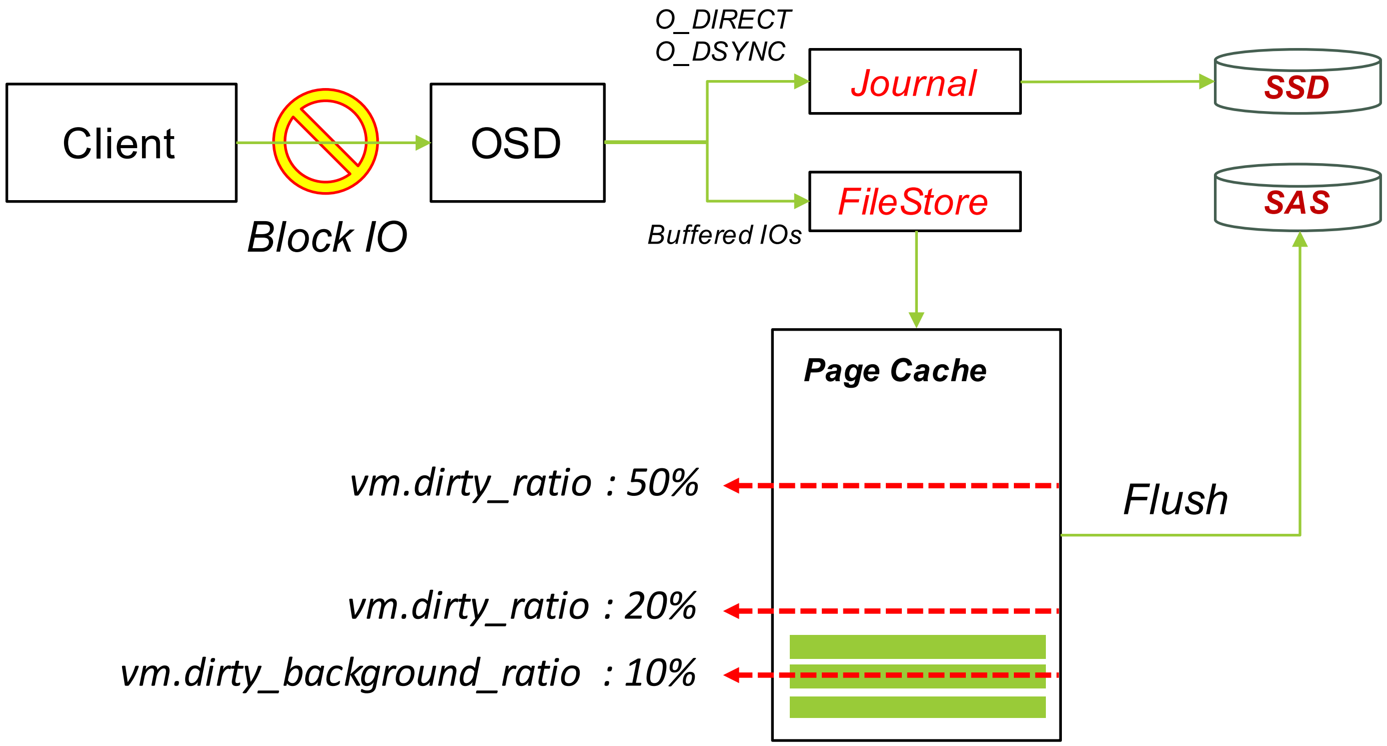
Result



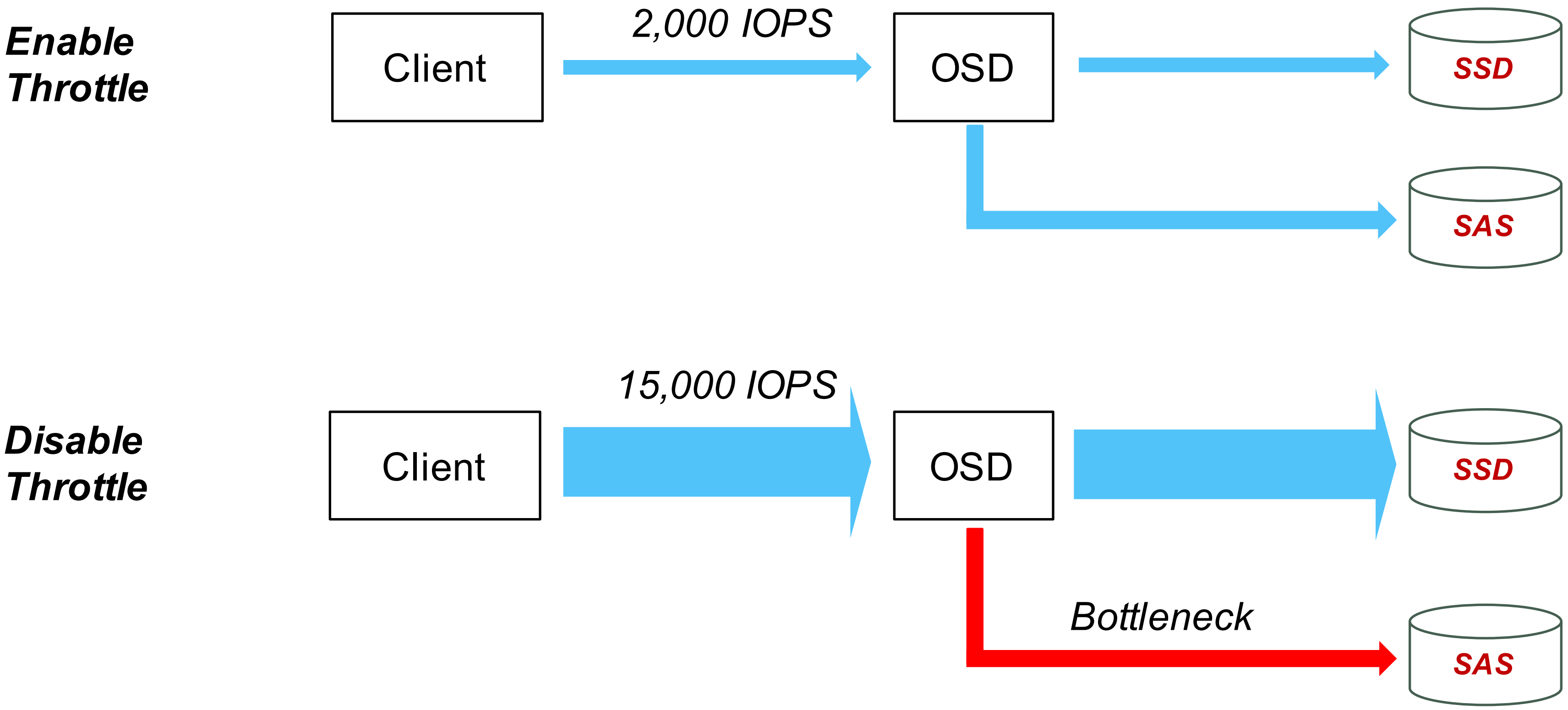
2,094 → 14,264 (x7)

Analysis

After 35 secs, Performance (IPOS) drops below 100 ~ 200 IOPS…



Problems of Throttle

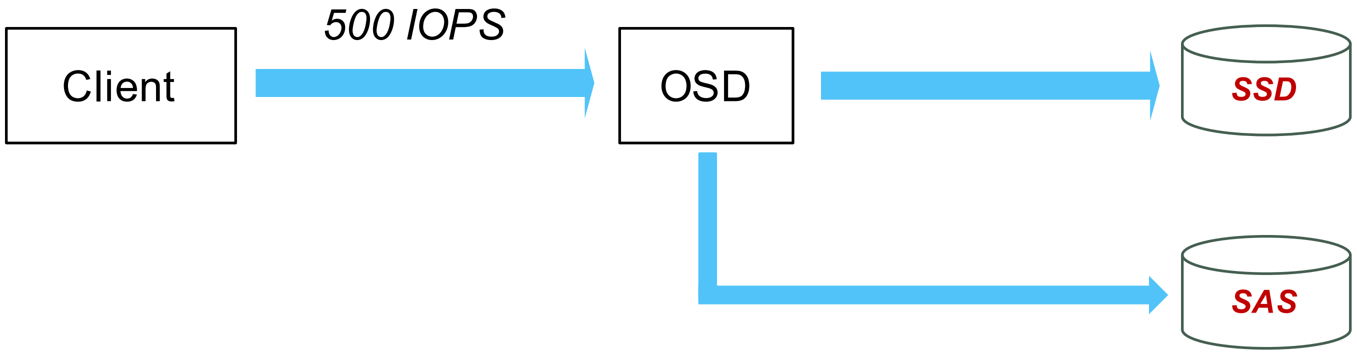


- *Slow performance*
- *No effect with SSD*

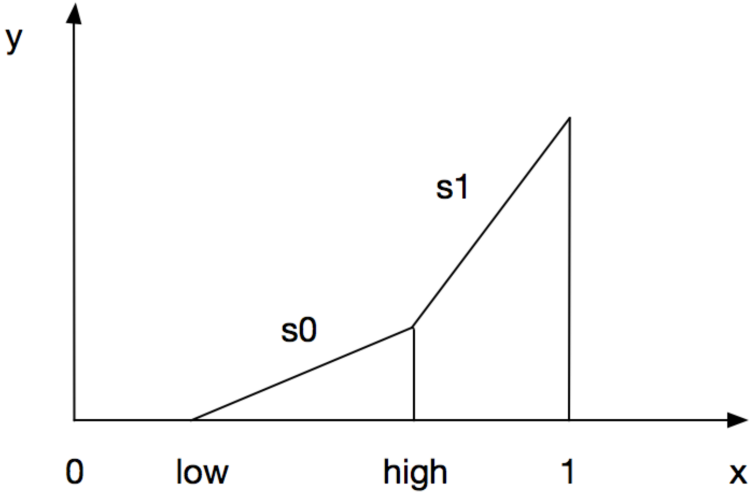
- *Fast performance*
- *Danger using High Page Cache*
- *Crash Ceph Storage*

Dynamic Throttle

Dynamic Throttle



- *Burst (~ 60 secs)*
- *Throttle From 80%*



filestore_queue_max_ops
filestore_queue_max_bytes

filestore_expected_throughput_ops
filestore_expected_throughput_bytes

filestore_queue_low_threshhold
filestore_queue_high_threshhold
filestore_queue_high_delay_multiple
filestore_queue_max_delay_multiple

```
r = current_op / max_ops
high_delay_per_count = high_multiple / expected_throughput_ops
max_delay_per_count = max_multiple / expected_throughput_ops
s0 = high_delay_per_count / (high_threshhold - low_threshhold)
s1 = (max_delay_per_count - high_delay_per_count) / (1 -
high_threshhold)
 if r < low_threshhold:
   delay = 0
elif r < high_threshhold:
   delay = (r - low_threshhold) * s0
else:
   delay = high_delay_per_count + ((r - high_threshhold) * s1)
```

Reference : http://blog.wjin.org/posts/ceph-dynamic-throttle.html

Conculsion

■ High performance improvements with SSD : **2,094 → 14,264 (x7)**

■ Must need Throttling for stable storage operation : Dynamic Throttle

■ Need to tune OS(page cache, io scheduler), Ceph config

QnA