Performance Analysis of Ceph Storage Clusters Based on Kernel and Userspace Traces

Progress Report Meeting

December 6, 2018

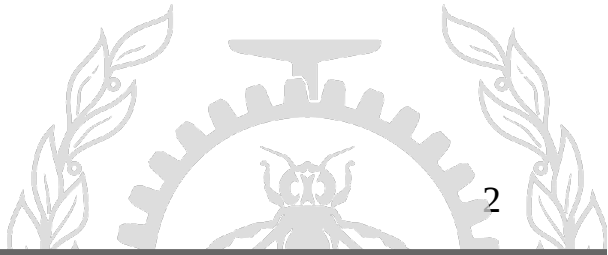
Houssem Daoud **Michel Dagenais**

École Polytechnique de Montréal

Laboratoire **DORSAL**

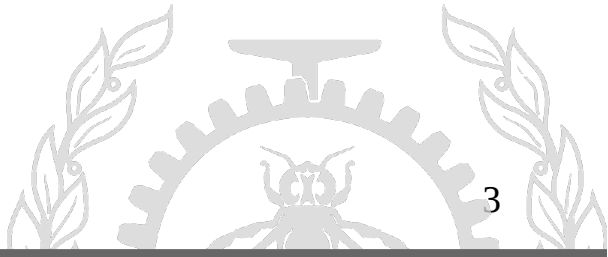
Agenda

- **Introduction**
- Background
- Proposed solution
- Evaluation
- Conclusion



Introduction

- The growing need to store large amount of data is generating demand for sophisticated storage systems.
- Many distributed storage systems have been developed to offer the flexibility of storing data using a cluster of storage nodes.
- Those systems must offer scalability, high performance and availability



Introduction

Ceph is an open-source distributed storage system

- Provides object, block and file storage
- Can run on commodity hardware
- Provides advanced replication and fault recovery mechanisms

Introduction

- The performance of a storage cluster can be affected by different factors: network bandwidth, disk speed, node failure, etc

Our goal is to provide a performance analysis tool that helps detecting performance bottlenecks in distributed storage systems

Background

Distributed storage systems

- **NFS:**
  - Client/Server architecture
  - Clients are able to mount a filesystem over the network.
  - Limited scalability
- **OceanStore & Farsite:**
  - Support large scale cluster of untrusted nodes
  - Use Byzantine agreement algorithm → big overhead
- **PVFS & Vesta**
  - Stripe data across multiple nodes → improve parallel access
  - Meta-data bottleneck

Background

Distributed storage systems

- **GPFS:**
  - Partially solve metadata access bottleneck by using *meta-nodes*
- **zFS & Lustre:**
  - Object-based storage: Data and meta-data are represented by an object.
  - Use a distributed lookup table
- **Sorento**
  - Replaced the lookup table by using a consistent hashing algorithm
  - The algorithm doesn't offer the flexibility to adapt the algorithm to the physical architecture of the cluster.

Background

Design challenges

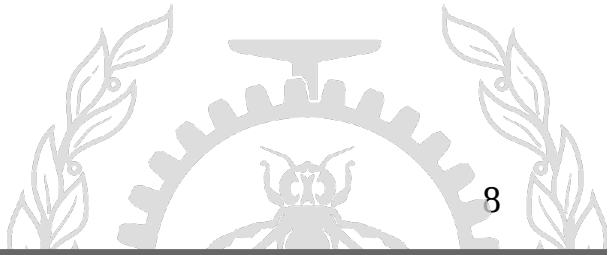
- Huge allocation tables

- Metadata access bottleneck

- Parallel access contention

- Block/File storage limitations

- Rebalancing, replication and fault recovery



Background

Design challenges

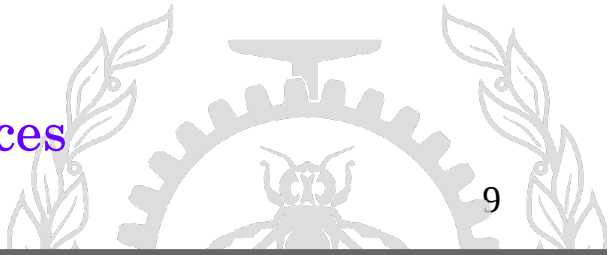
- Huge allocation tables
  → Using a pseudo-random placement algorithm (CRUSH)

- Metadata access bottleneck
  → Distributed metadata management

- Parallel access contention
  → Striping files across multiple disks

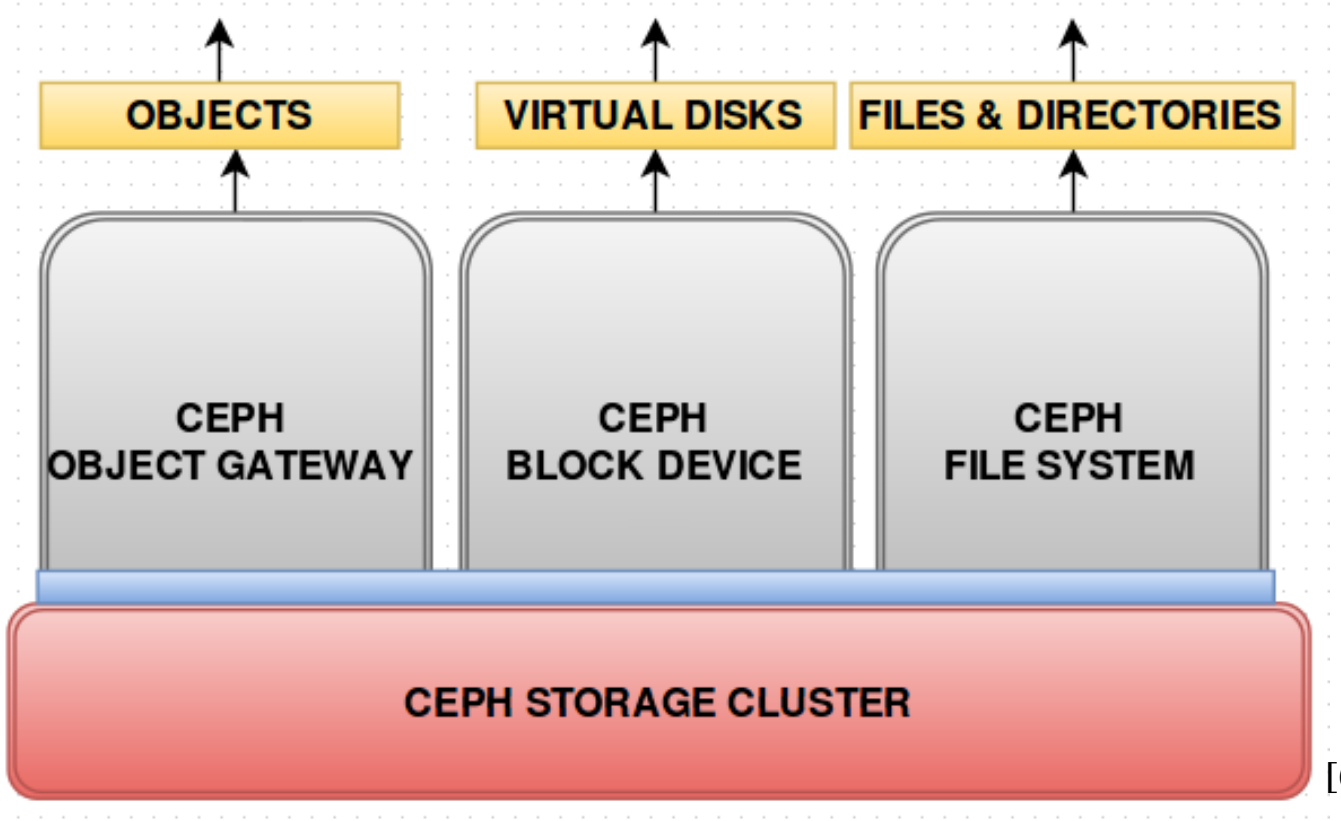
- Block/File storage limitations
  → Object storage

- Rebalancing, replication and fault recovery
  → Delegating management tasks to storage devices



Background

Ceph architecture



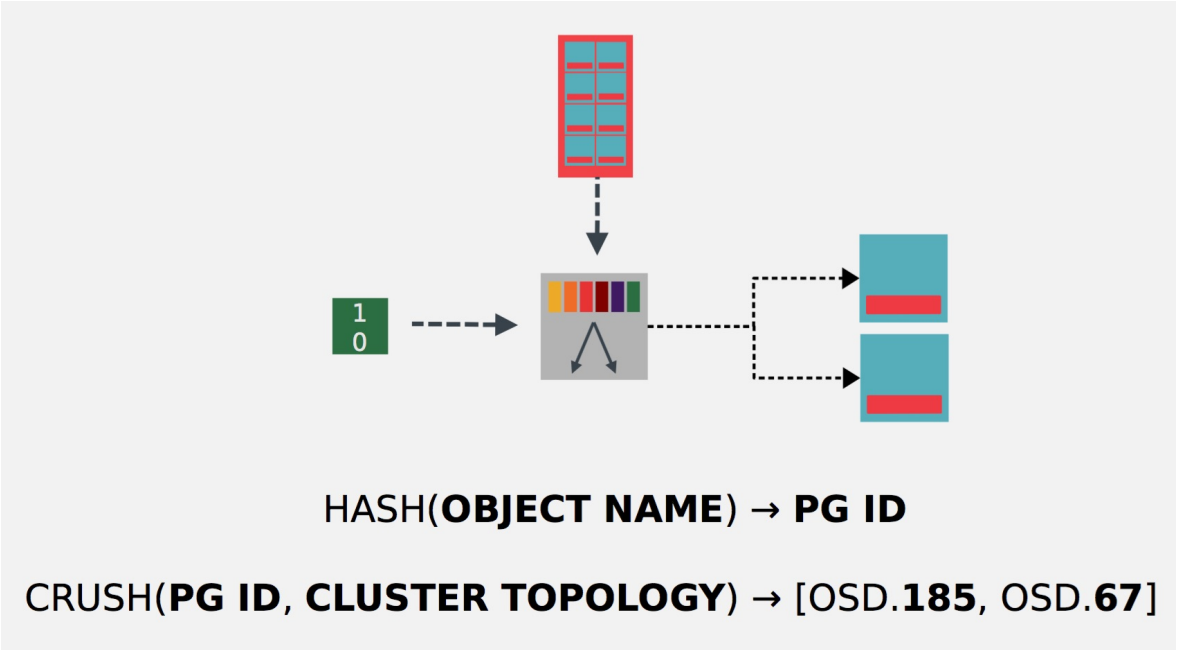
[6]

Background

Ceph architecture

CRUSH: Pseudo-random placement algorithm

- Statistically uniform distribution
- Stable → limited data migration on change



[6]

Background

Ceph architecture

- **OSD (Object Storage Daemon):**

  It is responsible for storing objects on a local file system and providing access to them over the network.

- **MON (Monitor):**

  Ceph monitors are light-weight processes that maintain a master copy of the cluster map.

- **MGR (Manager):**

  Provides additional monitoring and interfaces to external monitoring and management systems.

- **MDS (Metadata Server)**

  CephFS metadata information is kept in a separate RADOS pool and an additional cluster of metadata servers (MDS) is needed

Background

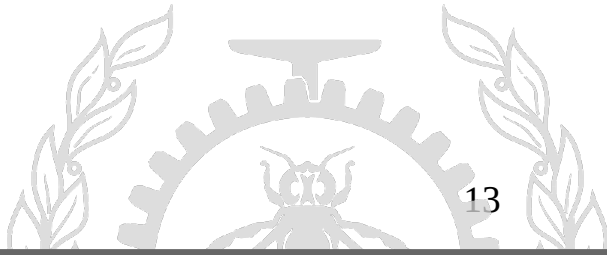
Literature review

- Zhang and al. [1] deployed a Ceph cluster on Openstack and investigated its performance by using multiple storage benchmarking tools including Bonnie++, dd and rados bench.

- Gudu and al. [2] studied the impact of journaling and proved that placing the journal in a fast device, such as main memory, improves dramatically the overall performance of the system.

- Poat and al. [3] investigated the performance of different data placement techniques like primary affinity and cache tiering on the overall performance of Ceph.

- Lee and al. [4] compared the performance of the different Ceph storage backends: FileStore,KStore and BlueStore.



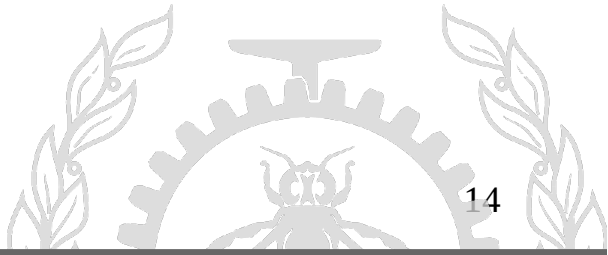
Background

Literature review

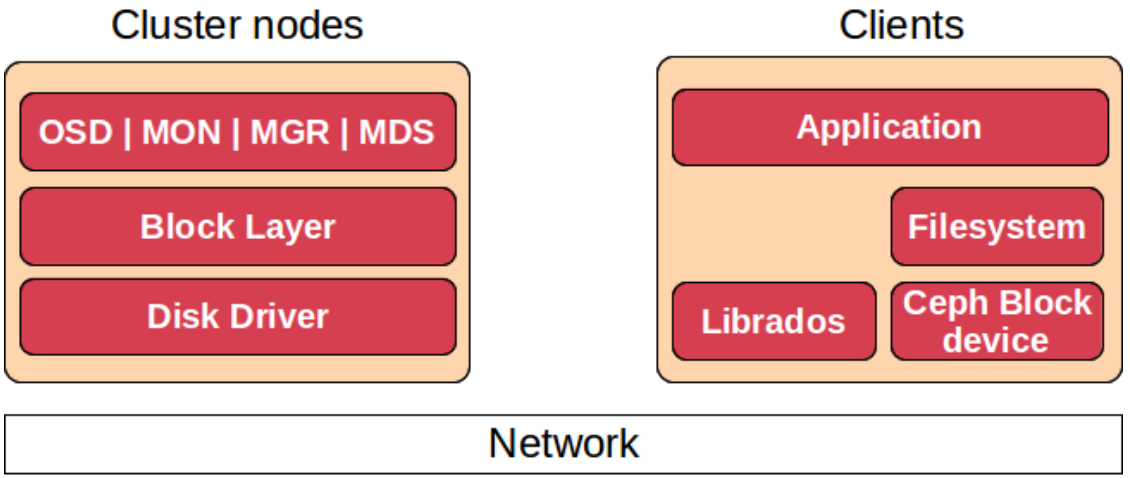
The studies reviewed so far, however, suffer from the fact that they use benchmarking as the only performance evaluation method

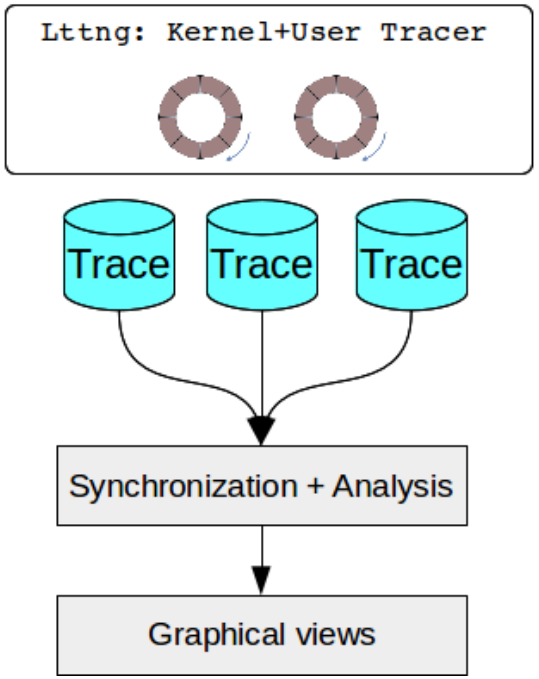
- The workloads used are often not representative of real world I/O patterns
- The approach doesn't provide a way to detect the root cause of unwanted latencies

The advantage of our approach is that:
- It collects data from real Ceph deployments at runtime
- It provides low-level data that helps pinpointing the cause of performance issues



Proposed solution





Proposed solution

Userspace Tracing

We use LTTng-UST to collect runtime information from Ceph

opwq_process_start opwq_process_finish	OSD worker creation/termination
do_osd_op_pre do_osd_op_pre_create do_osd_op_pre_delete do_osd_op_pre_truncate do_osd_op_pre_read do_osd_op_pre_write **do_osd_op_post**	I/O operations
async_enqueue_msg async_write_msg osd_op_reply	Networking queue

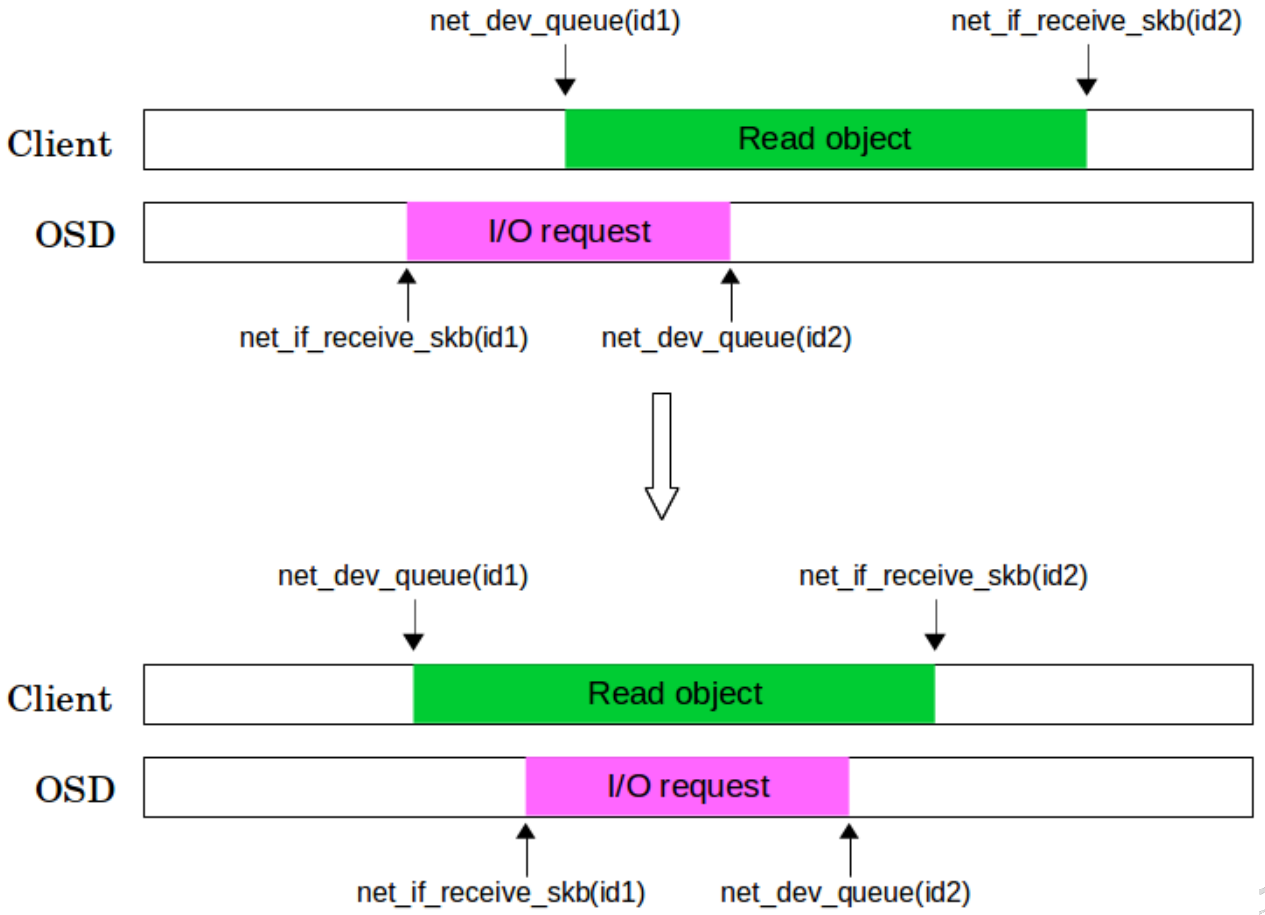
Proposed solution

Kernel Tracing

open, openat, read, readv, write, writev, lseek, dup, close	I/O related system calls
block_getrq block_rq_insert block_rq_issue block_rq_complete	Block layer tracepoints
scsi_dispatch_cmd_start scsi_dispatch_cmd_error scsi_dispatch_cmd_done scsi_dispatch_cmd_timeout	SCSI devices tracepoints

Proposed solution

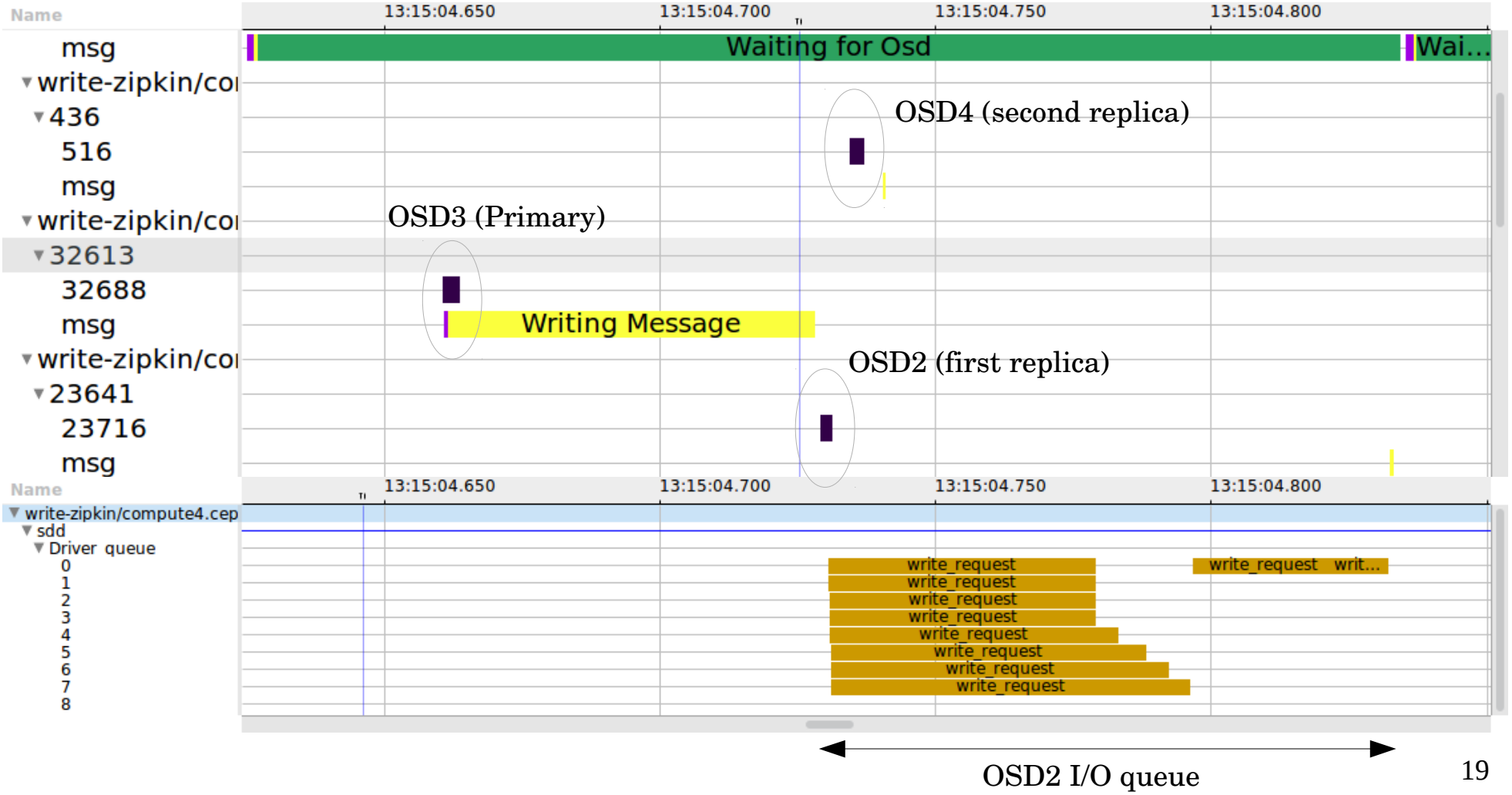
Trace synchronization



The events **net_dev_queue** and **net_if_receive_skb** are used for trace synchronization

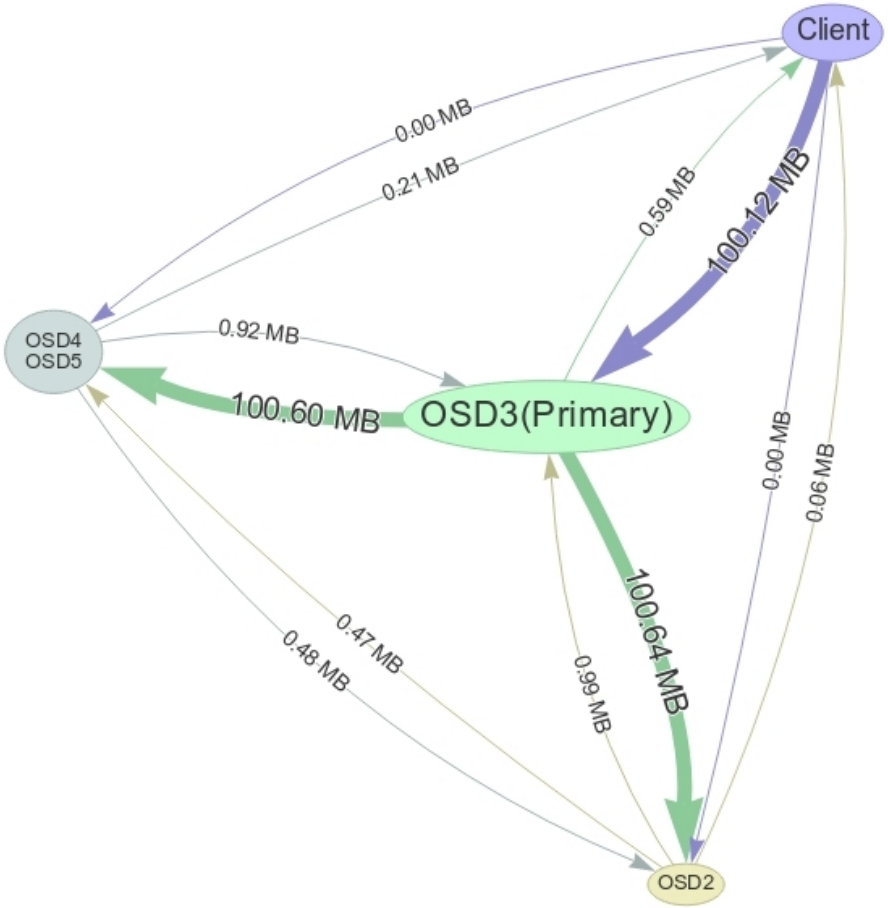
Proposed solution

This view shows the state of the different OSD processes and the disk I/O queue associated with each of them

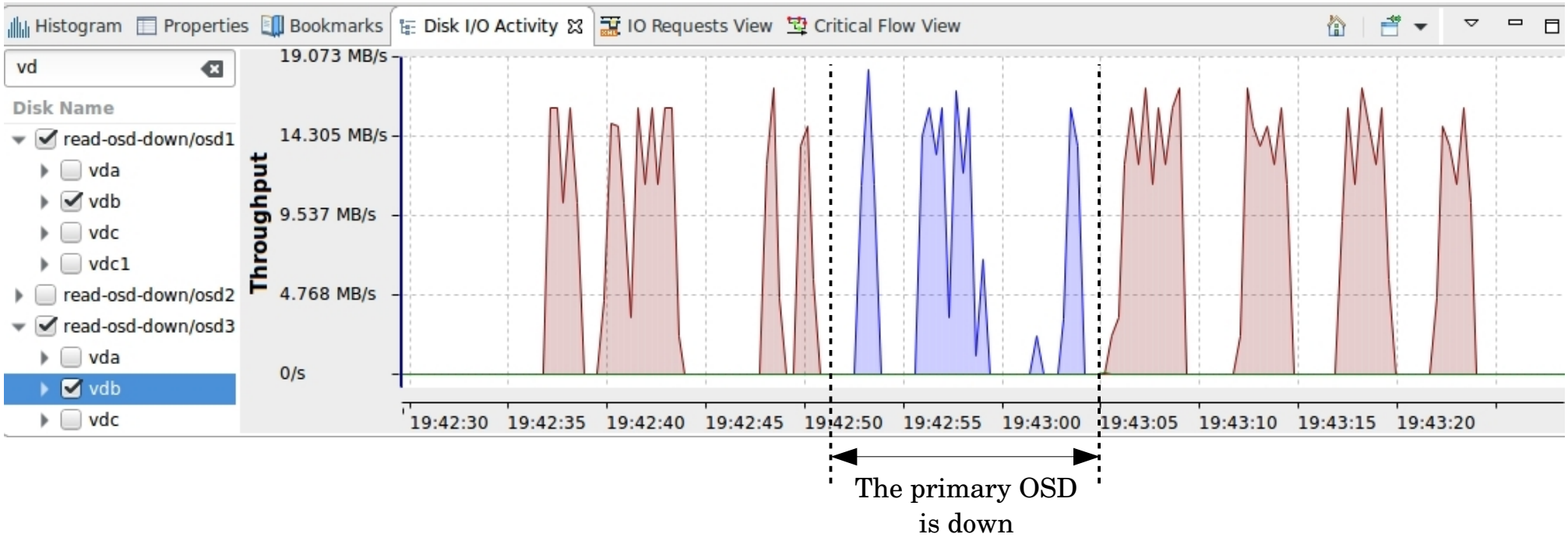


Proposed solution

- This view shows the network traffic between the different cluster nodes.
- The size of the nodes and the width of the arrows connecting them are scaled according to the quantity of network data exchanged



Proposed solution



This view provides a high level monitoring for all the OSDs involved in the cluster by showing the I/O throughput of all the disks on the same timeline.

Evaluation

We used ***rados bench*** to evaluate tracing overhead

Hardware environment	Software environment
Processor: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	**Operating system:** CentOS 7.5.1804
Cores: 24	**Linux Kernel:** 3.10
Memory: 188 GB	**LTTng:** 2.10.4
Storage: 1 SSD (256 GB) + 2 HDD (1 TB)	**Ceph:** Nautilus

Benchmark	Replication	Base (MB/s)	Tracing (MB/s)	Overhead (%)
Write	2	82.45	81.09	1.6
	3	64.34	63.69	1.0
	4	64.09	62.01	3.2
Sequential Read	2	111.99	111.97	0.0
	3	111.71	111.44	0.2
	4	111.92	111.94	0.0
Random Read	2	112.02	112.13	0.0
	3	111.93	111.04	0.7
	4	112.12	112.14	0.0

Demo

Conclusion

- We traced the different components of Ceph and we provided a comprehensive performance analysis framework for distributed storage clusters
- By using Kernel and userspace traces, our tool is able to detect low-level performance issues which are difficult to discover using existing tools
- As a future work, we propose to create an architecture that makes trace collection possible in larger clusters

houssem.daoud@polymtl.ca

References

[1] Zhang X, Gaddam S, Chronopoulos A. Ceph distributed file system benchmarks on an openstack cloud Cloud Computing in Emerging Markets (CCEM), 2015 IEEE International Conference on, IEEE, 2015; 113–120.

[2] Gudu D, Hardt M, Streit A. Evaluating the performance and scalability of the ceph distributed storage system BigData (Big Data), 2014 IEEE International Conference on, IEEE, 2014; 177–182.

[3] Poat M, Lauret J. Performance and advanced data placement techniques with ceph's distributed storage system. Journal of Physics: Conference Series, vol. 762, IOP Publishing, 2016; 012 025.

[4] Lee DY, Jeong K, Han SH, Kim JS, Hwang JY, Cho S. Understanding write behaviors of storage backends in cephobject store. Proceedings of the 2017 IEEE International Conference on Massive Storage Systems and Technology,vol. 10, 2017

[5] https://www.supportsages.com/ceph-part-3-technical-architecture-and-components/

[6] http://yauuu.me/ride-around-ceph-crush-map.html

houssem.daoud@polymtl.ca