Ceph中国社区 IT大咖说 知识共享平台

CEPH

CEPHALOCON APAC 2018

THE FUTURE OF STORAGE

22-23 March 2018 | BEIJING

Pinpoint Ceph bottleneck out of cluster behavior mists

Cheng, Yingxin

Why performance matters? - Collect

What is performance? - Represent

How to improve it? - Analyze

Why performance matters?

Answer: **better user experience**

- **Not** all activities matter. Requests impact users.
- **Not** all costs matter. Costs impact responses.
- The requests' **responding costs** matter.

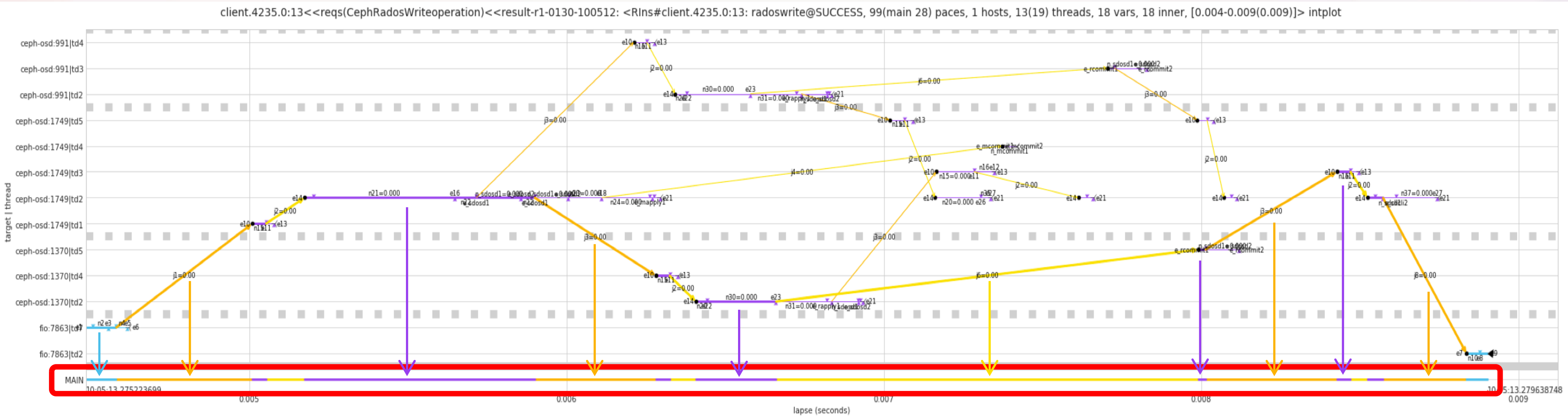
Bad: bottom-up strategies.

Approach?

1. Request-oriented distributed tracing.
2. Back-trace from response point.
3. Collect responding costs.

Why performance matters?

Collect responding costs.



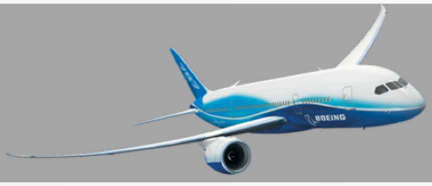
Responding costs

- Motivation: Cover the request responding time.
- Consecutive: Single path, and no overlap with each other.
- Category: Execution costs in thread, or waiting costs between threads.

Represent performance of concurrent requests … ***?***

What is performance?

Answer: **Latency and throughput**

Understand performance	Latency	Throughput
	✓	X
	X	✓

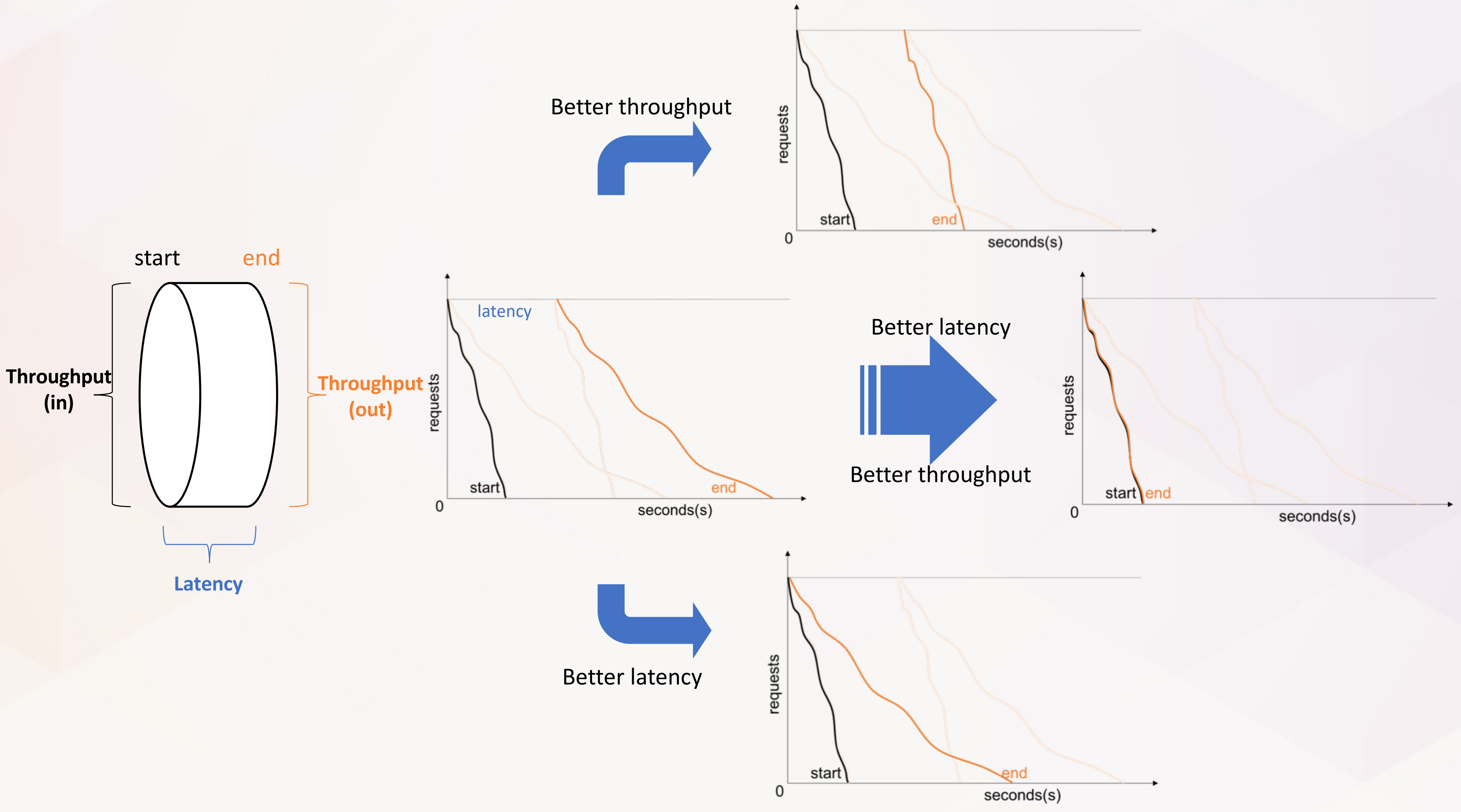
Bad: latency-only analysis, measure requests individually.

Approach?

1. Focus on performance of parallel requests.
2. New visualization for both *throughput* and *latency* of requests.

What is performance?

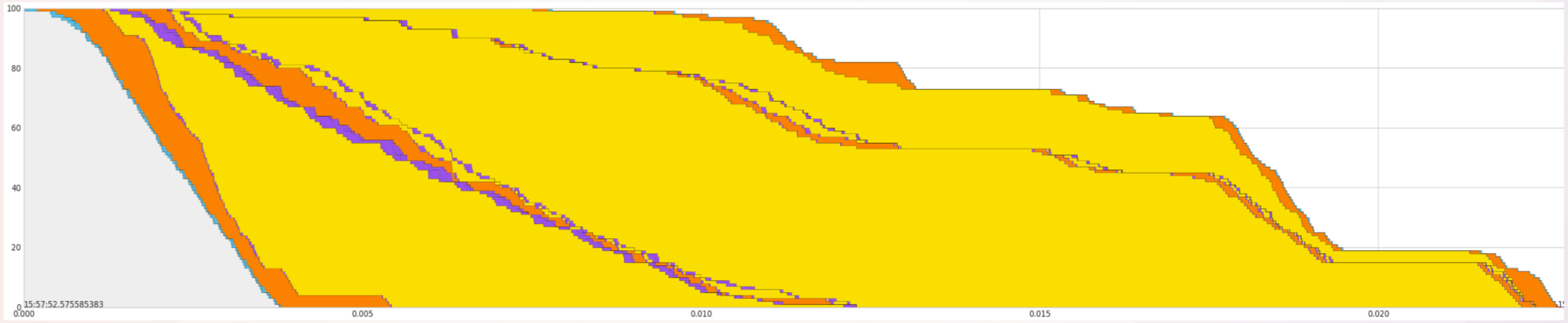
Represent both latency and throughput



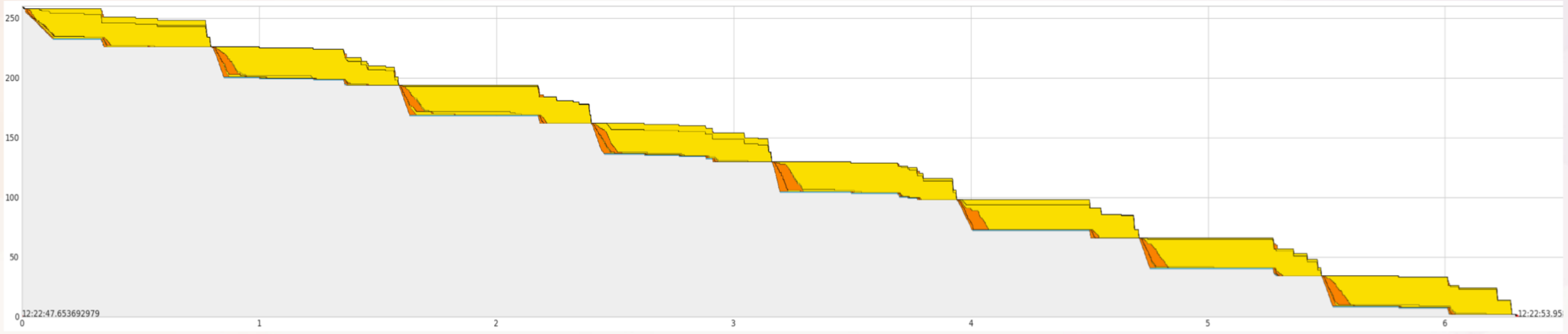
What is performance?

Represent performance of parallel requests

Responding costs: One-way, Consecutive, Flatten



More: Cluster behavior of 4M-Seq, iodepth=32



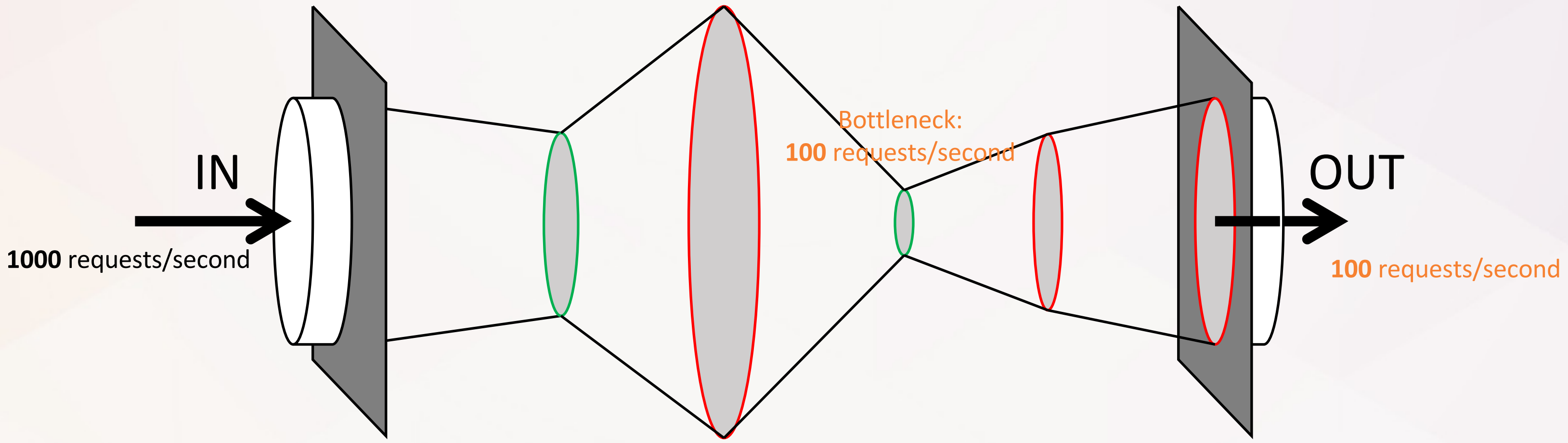
What is performance?

Represent bottleneck

Latencies are not necessarily dependent.

Throughputs are dependent.

- Lowest throughput -> system throughput.
- Worse: causes wait latencies; **most of times, bottleneck**



How to improve it?

Answer: **identify bottleneck root causes**

Root causes categories:

- Physical: configuration, deployment, hardware
- Logical: parameters, algorithm, architecture
- Other workload

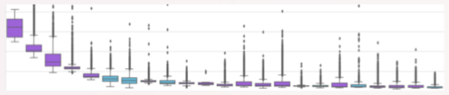
Bad: do optimization subjectively and in blindness.

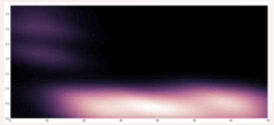
Approach?

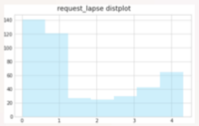
1. Relate each cost with:
   - Physical location: host, component, process(service), thread
   - Logical location: code, workflow
   - Runtime context: request, write length, offset …
2. Incremental analysis
   - Controlled-variables
   - Orthogonal methods
   - Verification

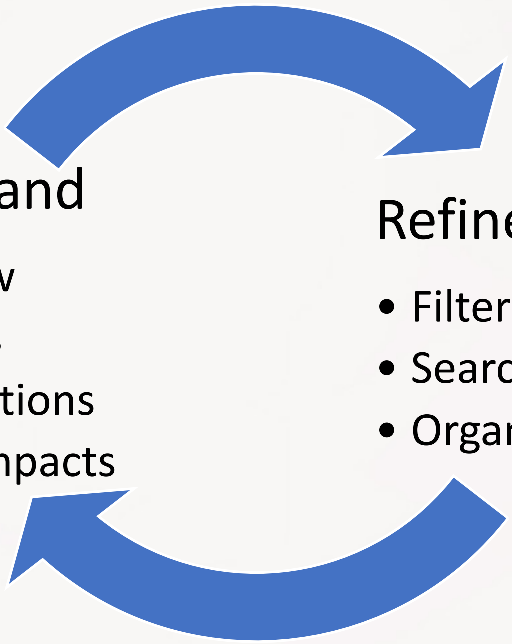
How to improve it?

Incremental & interactive **analysis**







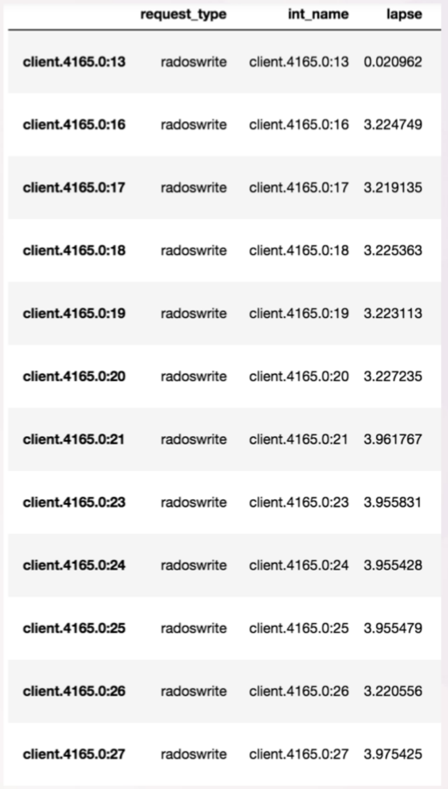


Understand

- Overview
- Statistics
- Visualizations
- Factor impacts

Refinements

- Filter
- Search
- Organize



	request_type	int_name	lapse
client.4165.0:13	radoswrite	client.4165.0:13	0.020962
client.4165.0:16	radoswrite	client.4165.0:16	3.224749
client.4165.0:17	radoswrite	client.4165.0:17	3.219135
client.4165.0:18	radoswrite	client.4165.0:18	3.225363
client.4165.0:19	radoswrite	client.4165.0:19	3.223113
client.4165.0:20	radoswrite	client.4165.0:20	3.227235
client.4165.0:21	radoswrite	client.4165.0:21	3.961767
client.4165.0:23	radoswrite	client.4165.0:23	3.955831
client.4165.0:24	radoswrite	client.4165.0:24	3.955428
client.4165.0:25	radoswrite	client.4165.0:25	3.955479
client.4165.0:26	radoswrite	client.4165.0:26	3.220556
client.4165.0:27	radoswrite	client.4165.0:27	3.975425

Distributed-tracing:	Motivation-aligned.
Visualization:	Straightforward performance representation.
Interactive frontend:	Be analysis-friendly.

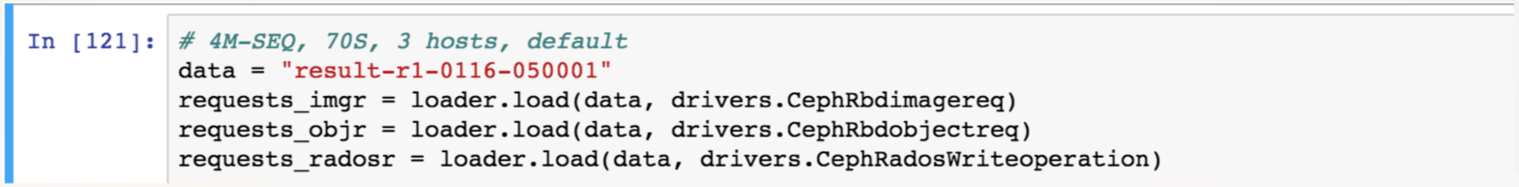
An example

1. Distributed-tracing: RBD image write

Background: RBD image write data-path

Component	Request	Entity
RBD	ImageWriteRequest	Images
	ObjectRequest	Objects
RADOS	ObjectWriteOperation	
OSD		
ObjectStore		

Experiment: 3VMs, 4M-SEQ-Write, iodepth=16



```
In [121]: # 4M-SEQ, 70S, 3 hosts, default
          data = "result-r1-0116-050001"
          requests_imgr = loader.load(data, drivers.CephRbdimagereq)
          requests_objr = loader.load(data, drivers.CephRbdobjectreq)
          requests_radosr = loader.load(data, drivers.CephRadosWriteoperation)
```

An example

2.1. Visualize performance (ImageWriteRequests)

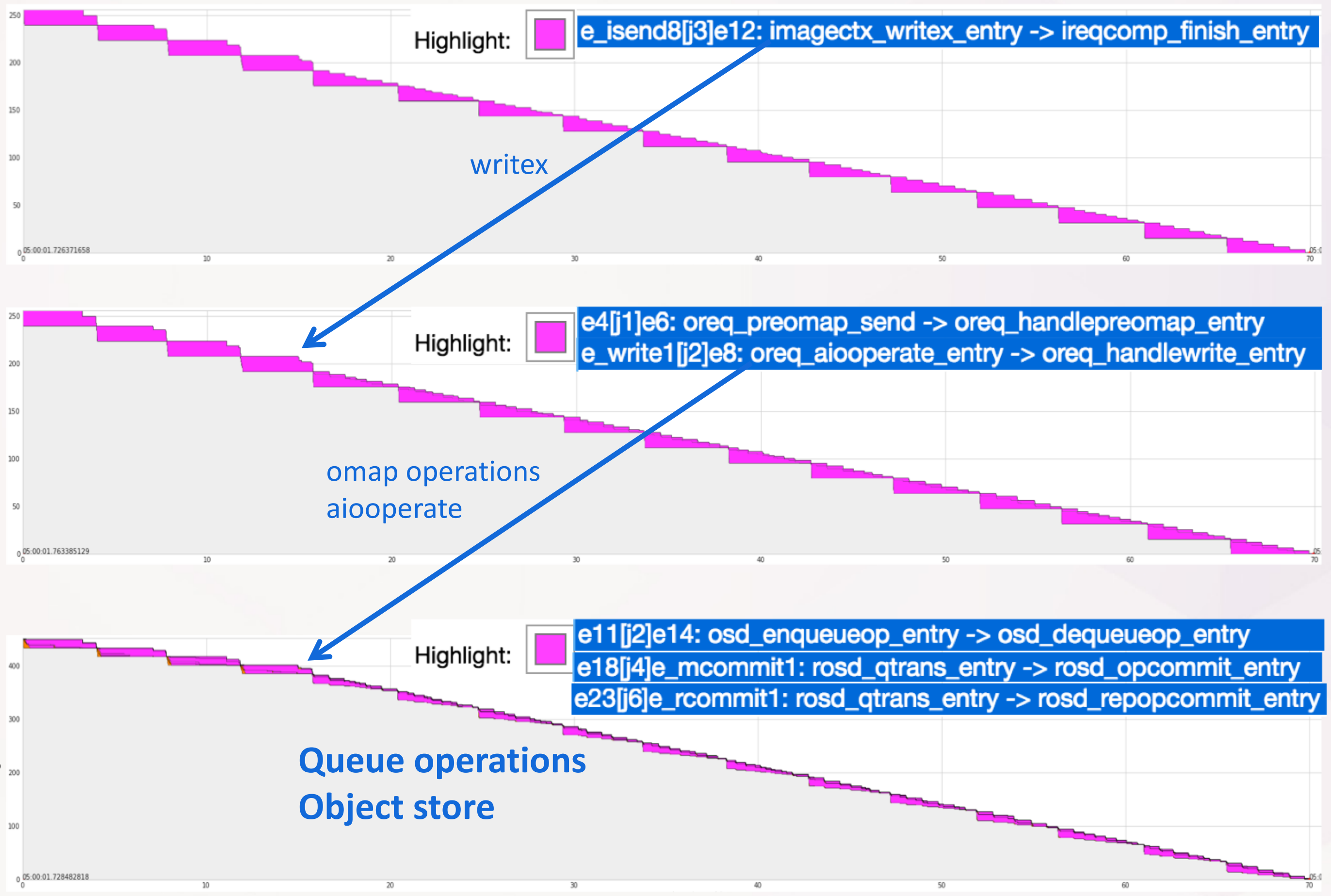
RBD::ImageWriteRequests

↓

RBD::ObjectRequests

↓

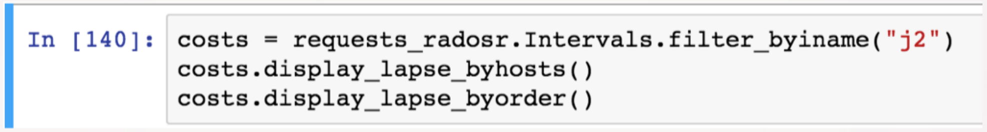
RADOS::ObjectWriteOperations



An example

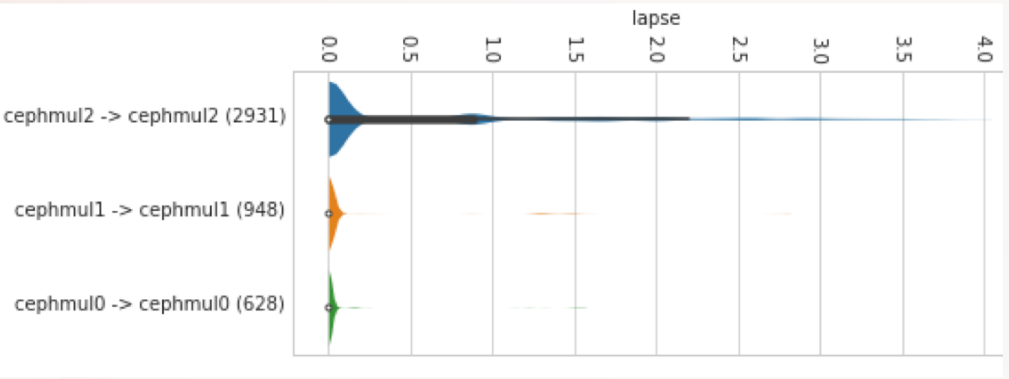
3.1. Interactive Analysis

Filter by workflow step "j2" (osd enqueueop -> dequeueop)

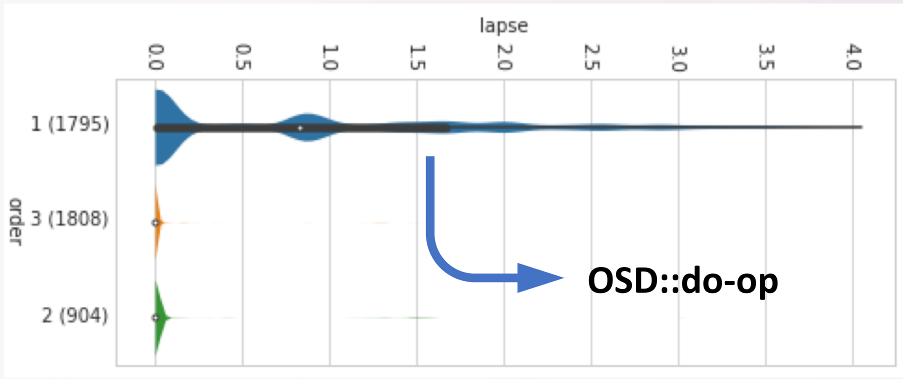


```
In [140]: costs = requests_radosr.Intervals.filter_byiname("j2")
          costs.display_lapse_byhosts()
          costs.display_lapse_byorder()
```

Physical location: costs by host



Logical location: costs by workflow order



An example

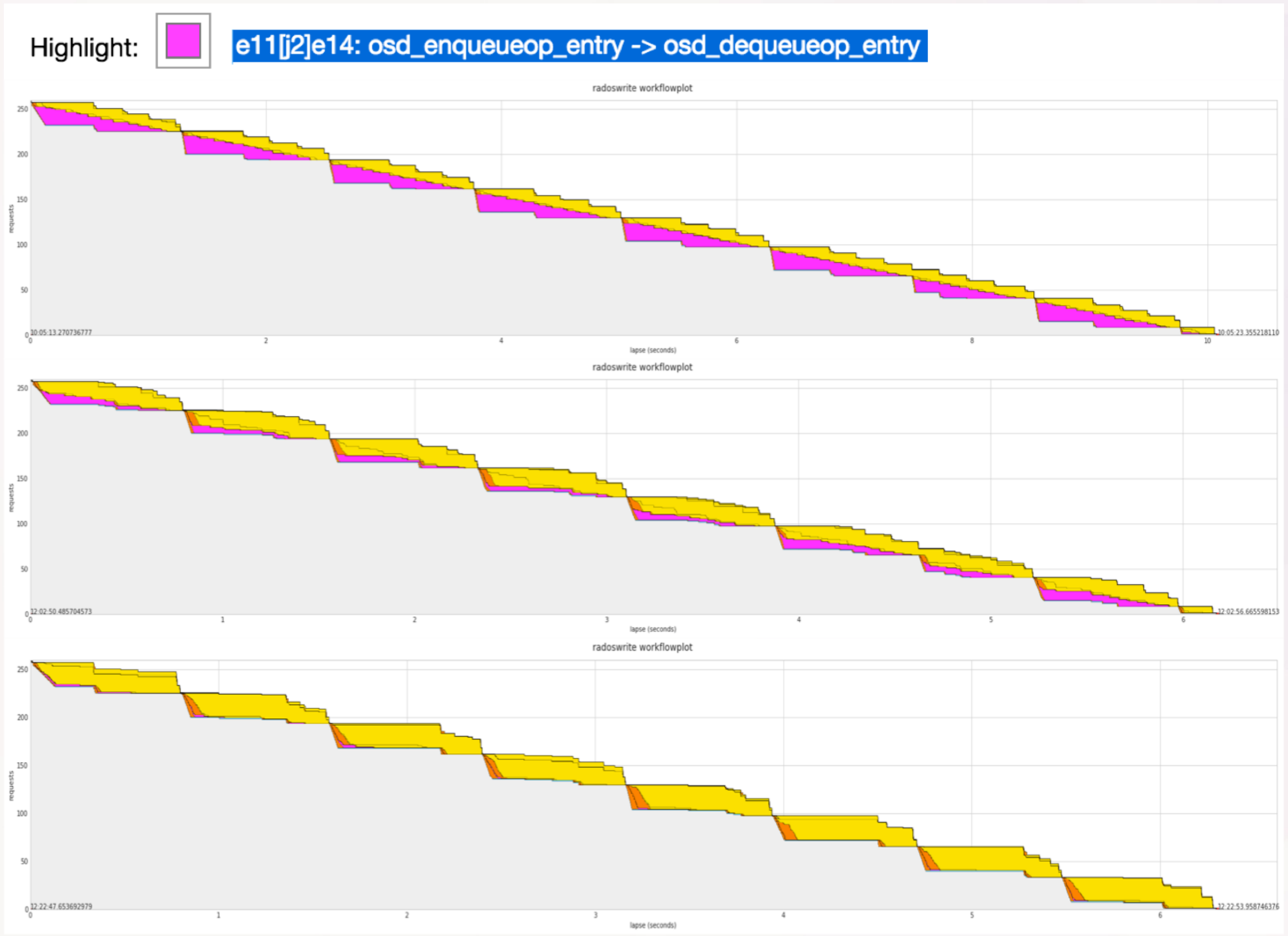
3.2. Root cause Analysis (do-op, 4M-SEQ)

Parameters:
- **osd_op_num_shards**
- osd_op_num_threads_per_shard
- pg_num

1 Shard

8 Shards

32 Shards



An example

Interactive analysis ...

- Lapse, host/thread count distribution
- Cost distribution by hosts, steps ...
- Show longest, most-complex request
- Message heatmap between hosts
- Write balance
- RBD cache validity
- Combination with resource monitoring tools

Ceph中国社区 IT大咖说 知识共享平台

Thank you!

Distributed-tracing

Visualization

Interactive frontend