Ceph at Spreadshirt

June 2016

Ansgar Jazdzewski, System Engineer

Jens Hadlich, Chief Architect

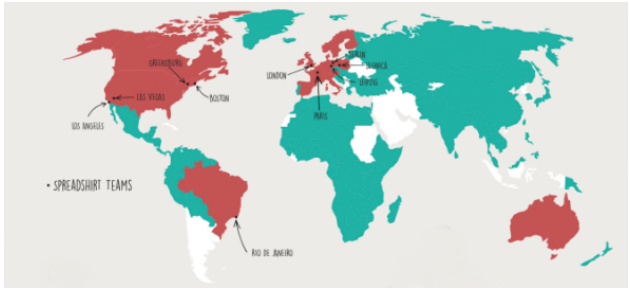
Agenda

- Introduction
- Setup
- Performance
- Geo-Replication & Backup
- Lessons learned

Introduction

About Spreadshirt

Spread it with Spreadshirt



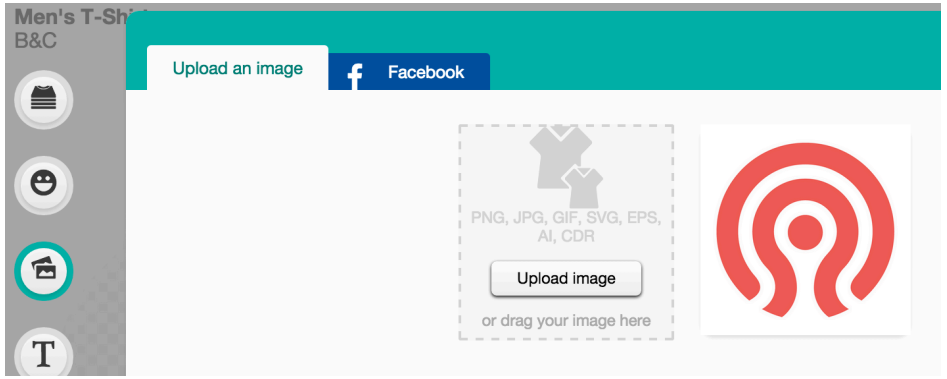
A global e-commerce platform for everyone to create, sell and buy ideas on clothing and accessories across many points of sale.

- Founded in 2002, based in Leipzig
- 550+ employees around the world
- Global shipping (180+ countries)
- Community of 70.000+ active sellers
- € 85M revenue (2015)
- 3.6M shipped items (2015)

Why do we need object storage?

Main use case: user generated content (vector & pixel images)

- Some 10s of terabytes (TB) of data
- 2 typical sizes:
  - A few MB (originals)
  - A few KB (post-processed)
- 50.000 uploads per day at peak
- Currently 35M+ objects
- Read > write



Previous approach

NFS

- Well-branded vendor
- Millions of files and directories
- Sharding per user / user group
- Same shares mounted on almost every server

Previous approach

NFS

- Problems
  - Regular UNIX tools became unusable
  - Not designed for “the cloud” (e.g. replication designed for LAN)
  - Partitions can only grow, not shrink
  - Performance bottlenecks
- Challenges
  - Growing number of users → more content
  - Build a truly global platform (multiple regions and data centers)

Why Ceph?

- Vendor independent
- Open source
- Runs on commodity hardware (big or small)
- Local installation for minimal network latency
- Existing knowledge and experience
- S3 API
- Easy to add more storage

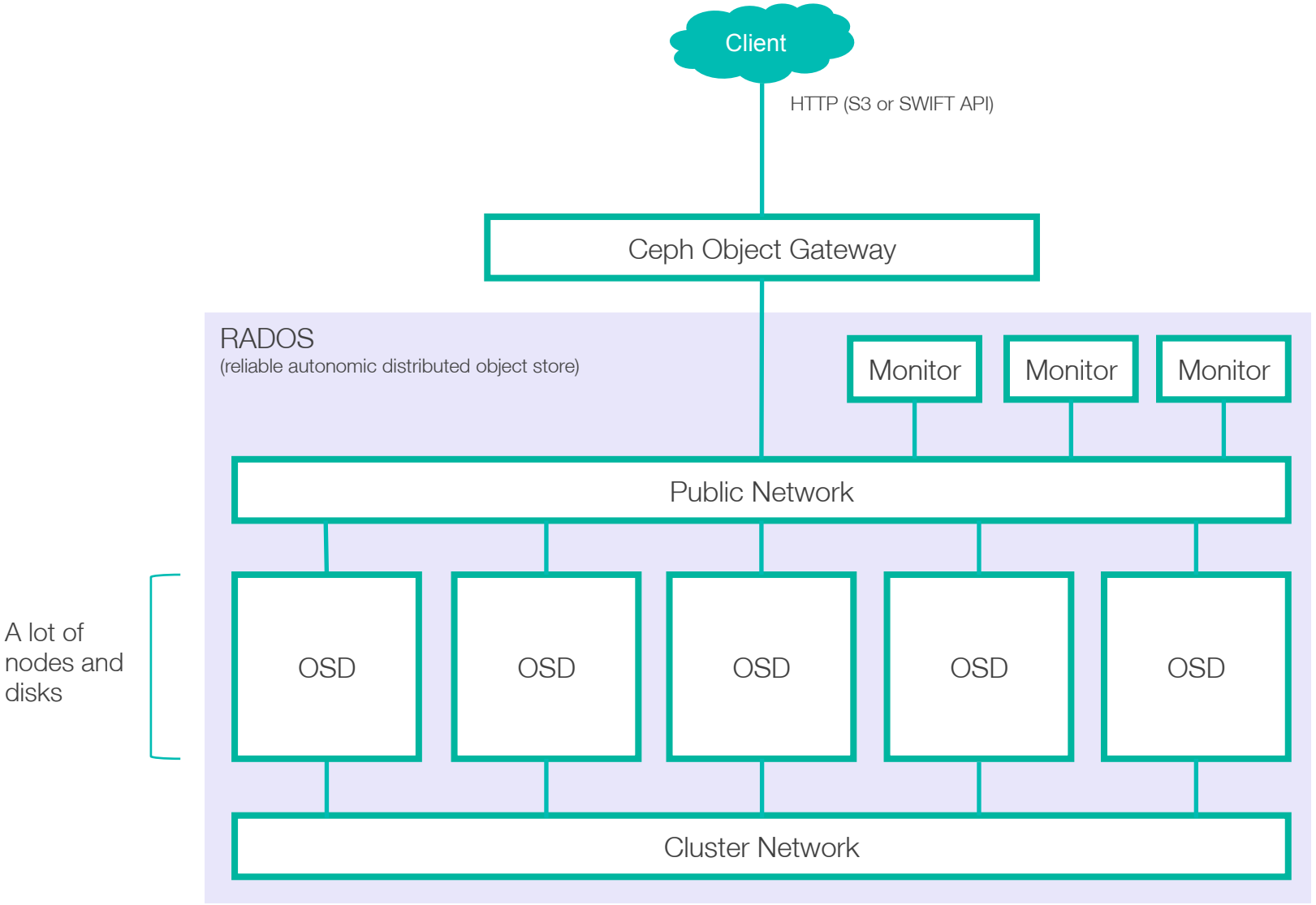
The Ceph offering

- Object storage
- Block storage
- File system

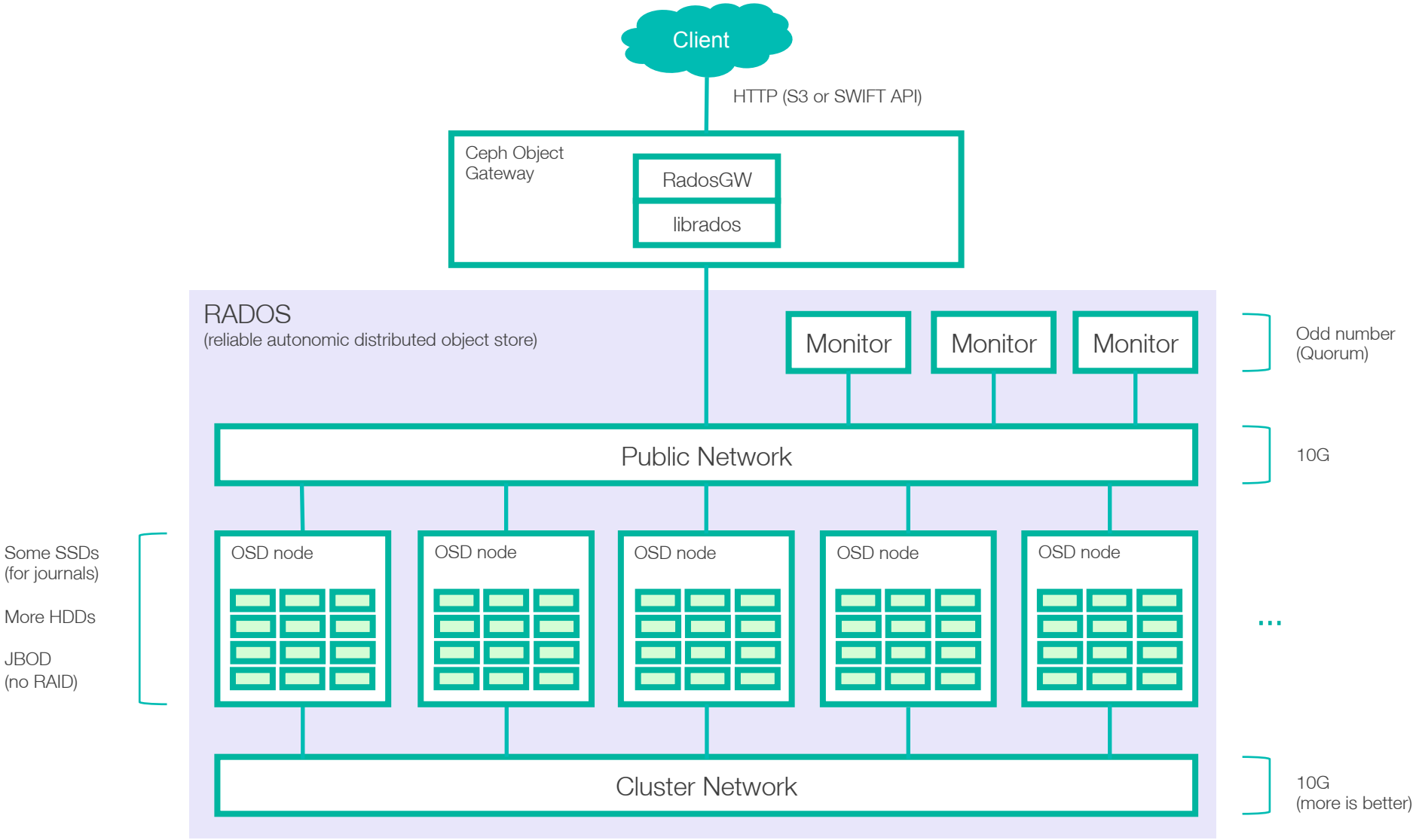
→ we (currently) focus on **object storage** (Ceph Object Gateway)

Setup

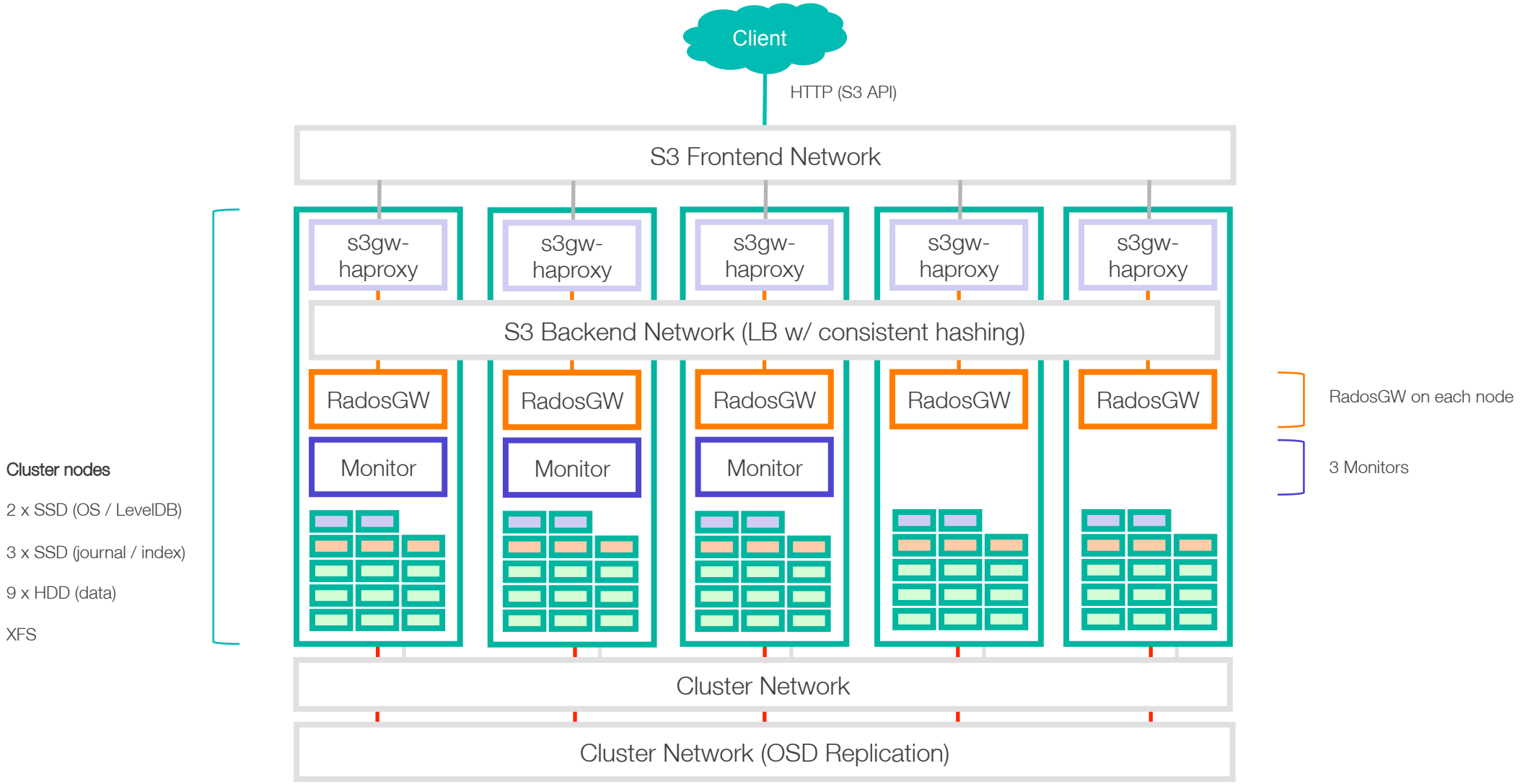
General Ceph Object Storage Architecture



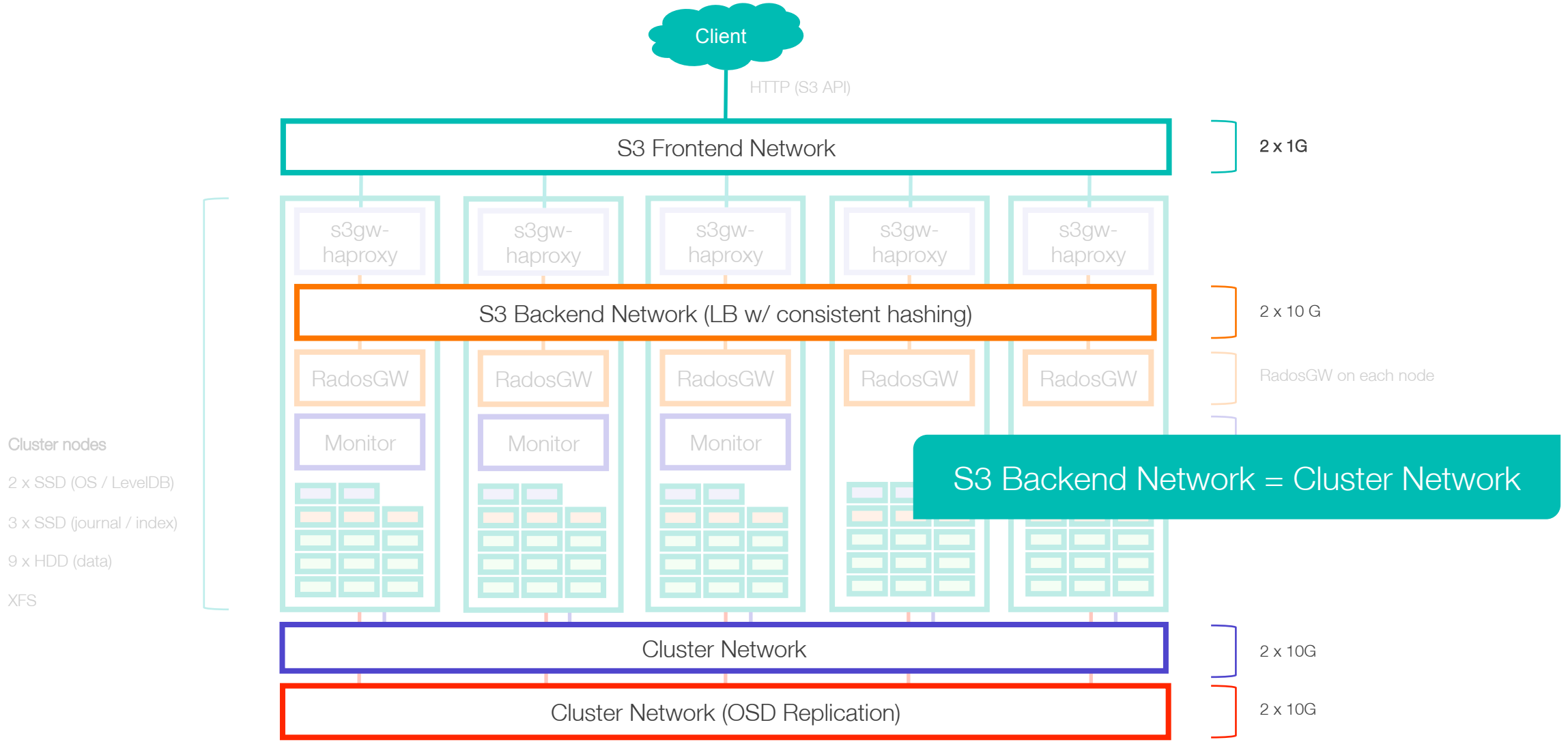
General Ceph Object Storage Architecture



Ceph Object Storage at Spreadshirt



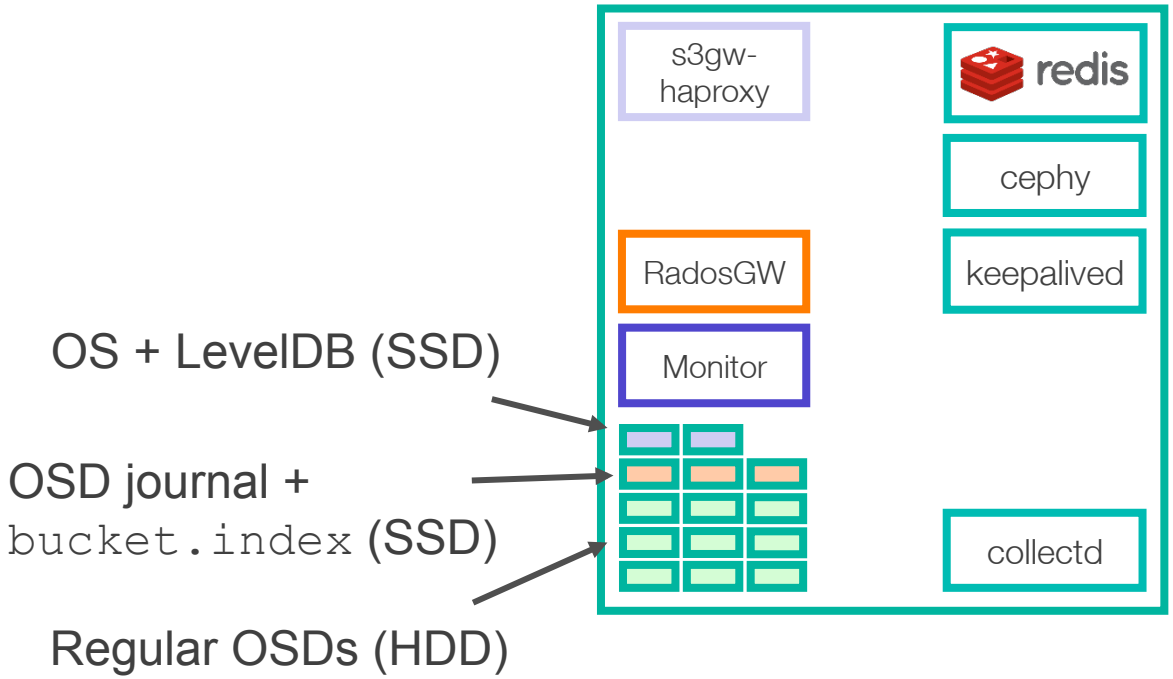
Ceph Object Storage at Spreadshirt



Ceph Object Storage at Spreadshirt

- Hardware configuration
  - 5 x Dell PowerEdge R730xd
    - Intel Xeon E5-2630v3, 2.4 GHz, 8C/16T
    - 64 GB RAM
    - 9 x 4 TB NLSAS HDD, 7.2K
    - 3 x 200 GB SSD Mixed Use
    - 2 x 120 GB SDD for Boot & Ceph Monitors (LevelDB)
    - 2 x 1 Gbit + 4 x 10 Gbit NW
  - Same for each data center (EU, NA)

One node in detail



Component	What?
`keepalived`	High-availability
`collectd`	Metrics (Grafana)
`s3gw-haproxy`	Geo-Replication (explained later)
Redis	
`cephy`	

Ceph OSD tree

```
ID  WEIGHT    TYPE NAME           UP/DOWN REWEIGHT PRIMARY-AFFINITY
 -5   1.27487 root ssd
 -4   0.25497     host stor05_ssd
  9   0.08499         osd.9            up  1.00000          1.00000
...
 -7   0.25497     host stor04_ssd
 21   0.08499         osd.21           up  1.00000          1.00000
...
 -9   0.25497     host stor03_ssd
 33   0.08499         osd.33           up  1.00000          1.00000
...
-11   0.25497     host stor02_ssd
 45   0.08499         osd.45           up  1.00000          1.00000
...
-13   0.25497     host stor01_ssd
 57   0.08499         osd.57           up  1.00000          1.00000
...
 -3 163.79997 root hdd
 -2  32.75999     host stor05_hdd
  0   3.64000         osd.0            up  1.00000          1.00000
  1   3.64000         osd.1            up  1.00000          1.00000
  2   3.64000         osd.2            up  1.00000          1.00000
  3   3.64000         osd.3            up  1.00000          1.00000
...
```

Performance

Why does latency matter?

Results of some early performance smoke tests*:

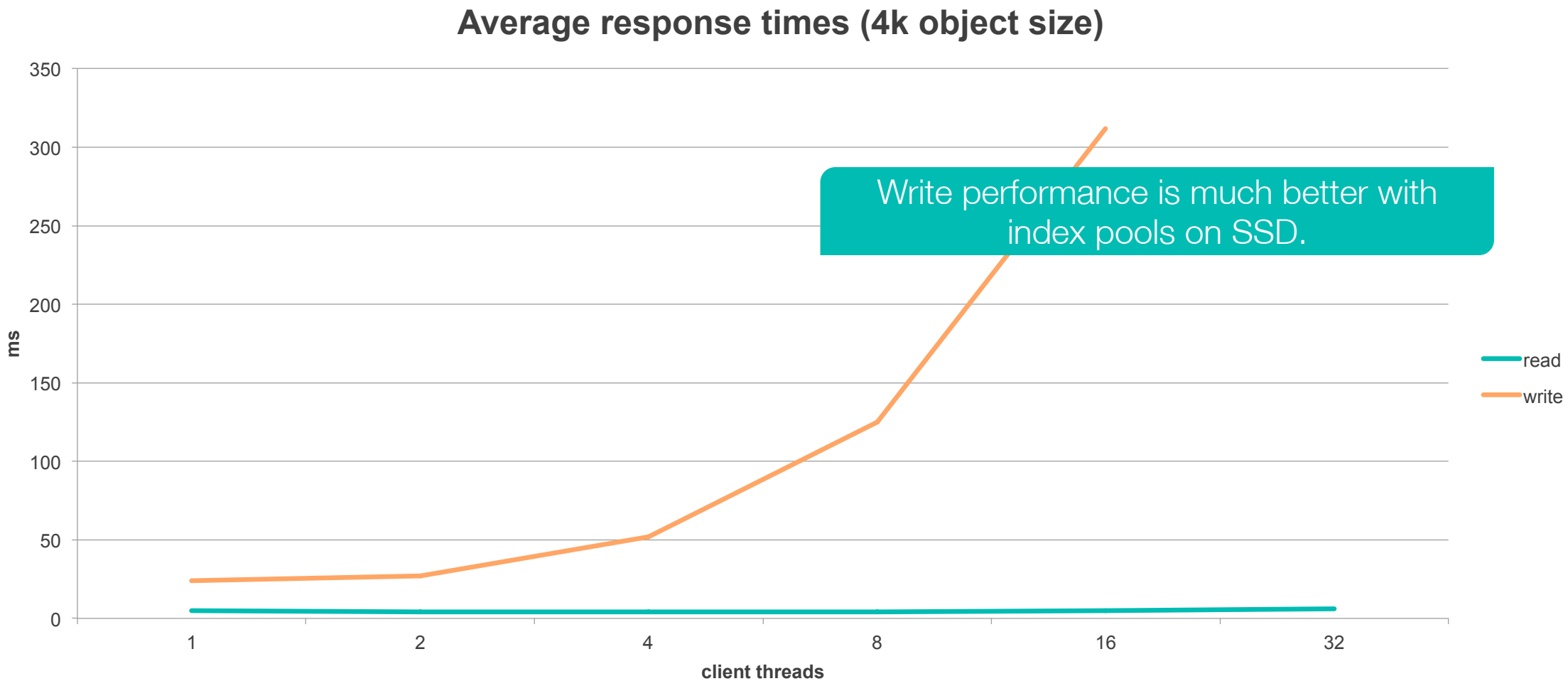
	Ceph S3 (test setup)	AWS S3 eu-central-1	AWS S3 eu-west-1
Location	Leipzig	Frankfurt	Ireland
Response time (average)	6 ms	25 ms	56 ms
Response time (p99)	47 ms	128 ms	374 ms
Requests per second	405	143	62

* Random read, object size 4KB, 4 parallel threads, location: Leipzig

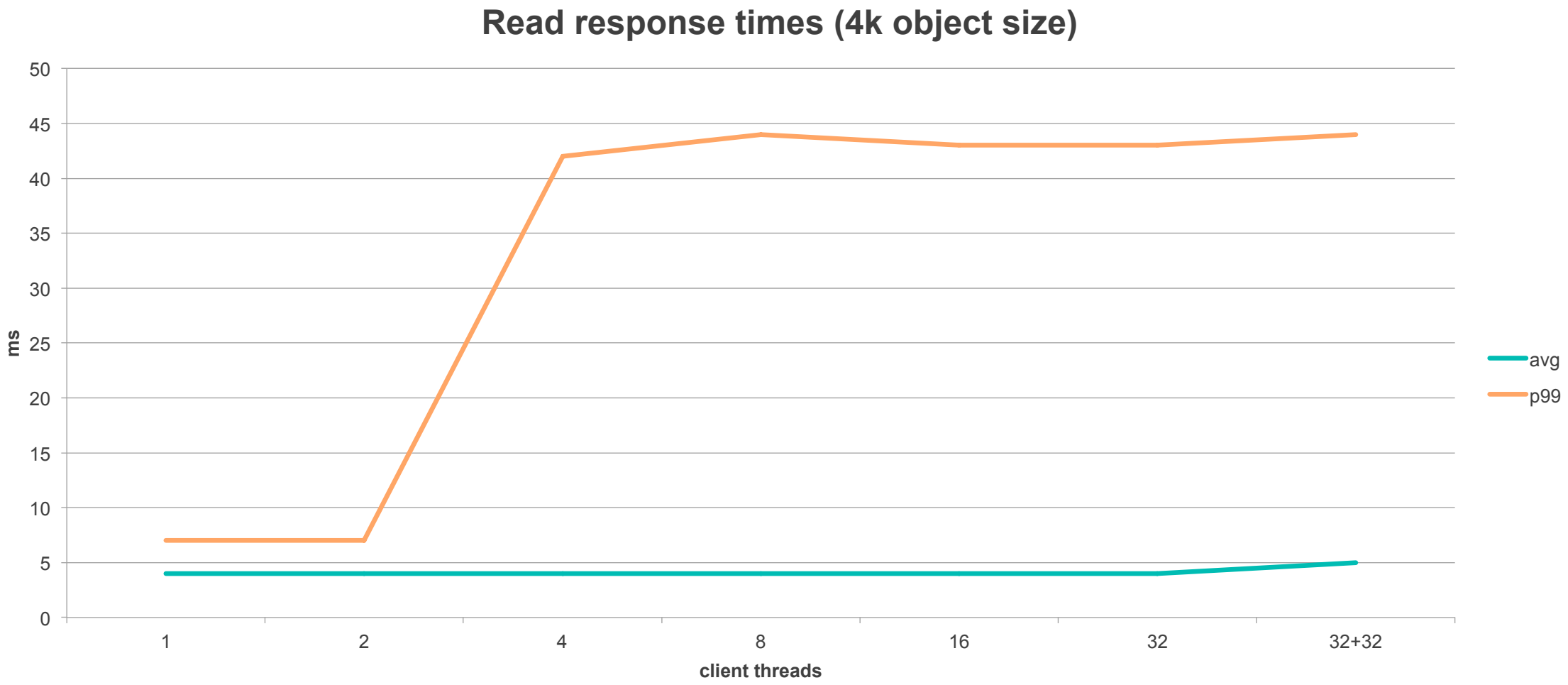
Performance

- Test setup
  - “Giant” release
  - Index pools still on HDDs
- Test result
  - E.g. random read, 16 parallel threads, 4KB object size
  - Response time:
    - 4 ms (average)
    - 43 ms (p99)
  - ~2.800 requests/s

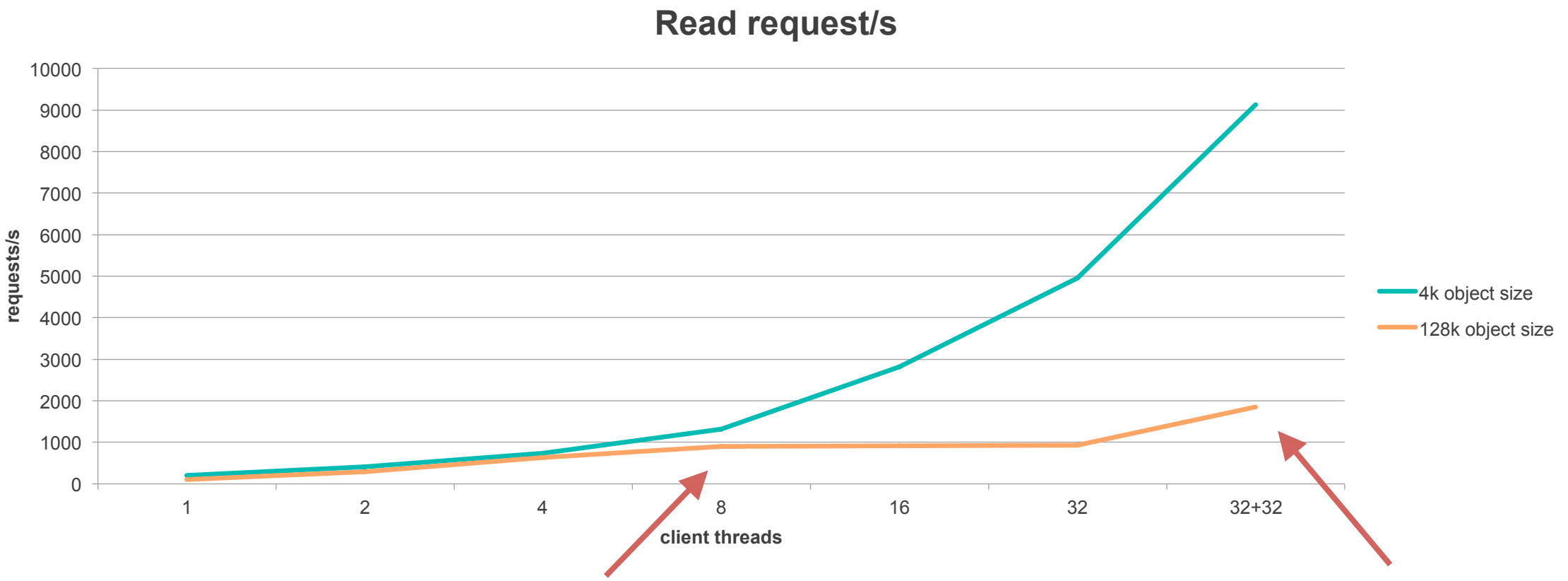
Performance: read vs. write



Performance: read



Performance: horizontal scaling



1 client / 8 threads: 1G network
almost saturated at ~117 MB/s

2 clients: 1G network
saturated again;
but scale out works ☺

Geo-Replication & Backup

Geo-Replication & Backup

Why build a custom solution?

- Ceph provided `radosgw-agent` only provides (very) basic functionality:

  “For all buckets of a given user, synchronize all objects sequentially.”

Geo-Replication & Backup

Our requirements

- All data globally available
- 50M+ objects
- Near real time (NRT)
- S3 users / keys / buckets are pretty static

Geo-Replication & Backup

Our requirements

- All data globally available
- 50M+ objects
- Near real time (NRT)
- S3 users / keys / buckets are pretty static

Backup-Strategy

- No dedicated backup → replication
- No restore → fail-over to next alive cluster

Geo-Replication & Backup

Our requirements

- All data globally available
- 50M+ objects
- Near real time (NRT)
- S3 users / keys / buckets are pretty static

We need bucket notifications!

Backup-Strategy

- No dedicated backup → replication
- No restore → fail-over to next alive cluster

Geo-Replication

“Spreadshirt Cloud Object Storage”

- Ceph Object Storage (RadosGW) + bucket notifications + replication agent:
  - `s3gw-haproxy`* (custom version of HAProxy, written in C)
  - Redis
  - `cephy` tool suite (written in Go)
- Support from Heinlein for implementation

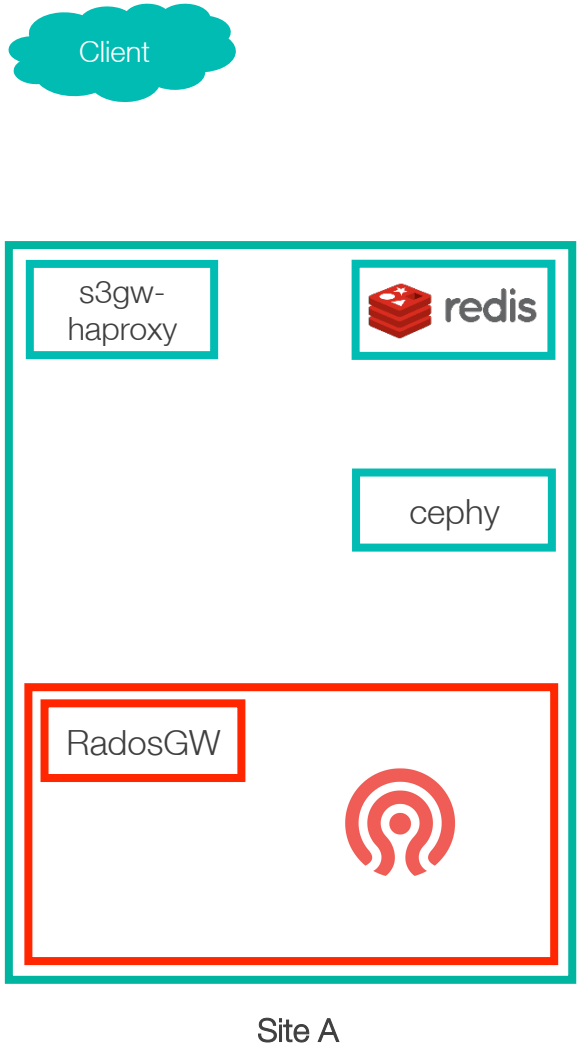
* https://github.com/spreadshirt/s3gw-haproxy

cephy

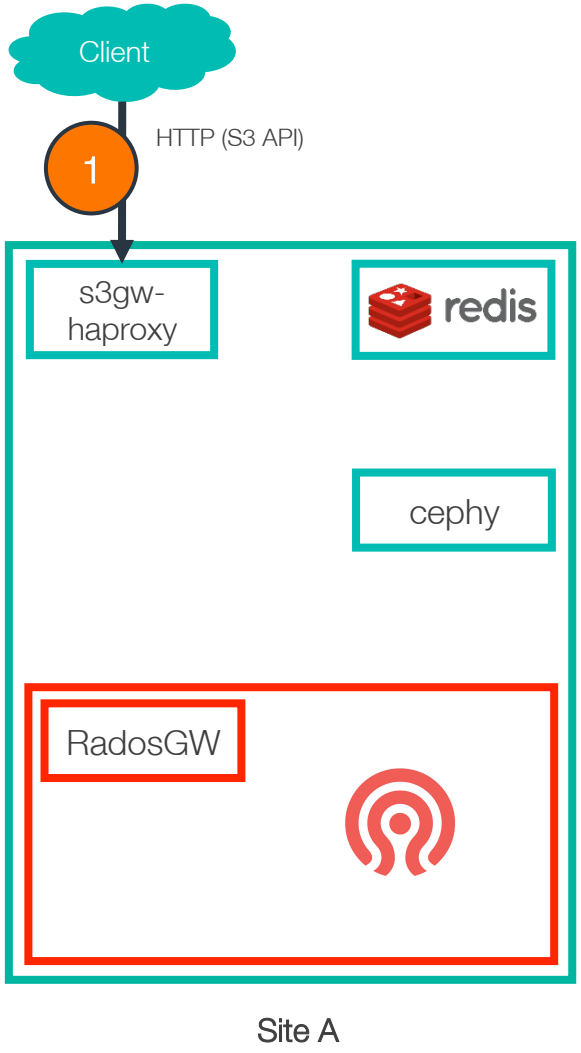
A tool suite for working with Ceph Object Storage through its S3 API:

- `cephy`
  basic file system operations between files/directories and objects/buckets
- `cephy-sync`
  synchronize two or more buckets (single source, multiple targets) using notifications from `cephy-listen` or `cephy-queue`
- `cephy-listen`
  creates sync notifications from S3 operations on a RadosGW socket
- `cephy-queue`
  generates sync notifications for all objects inside bucket

Replication in Action

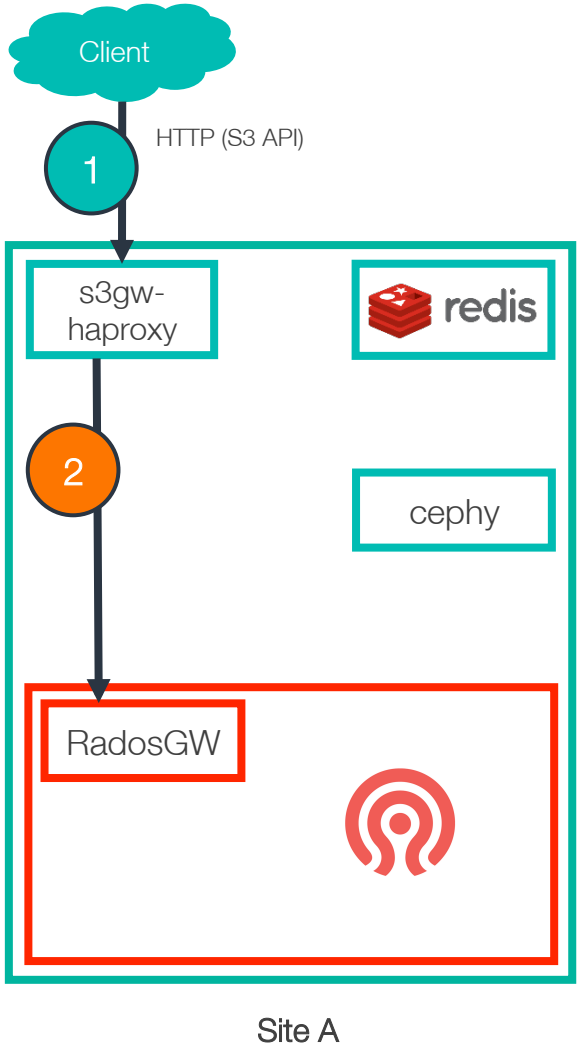


Replication in Action



1 `PUT` **`bucket/key`** (HAProxy)

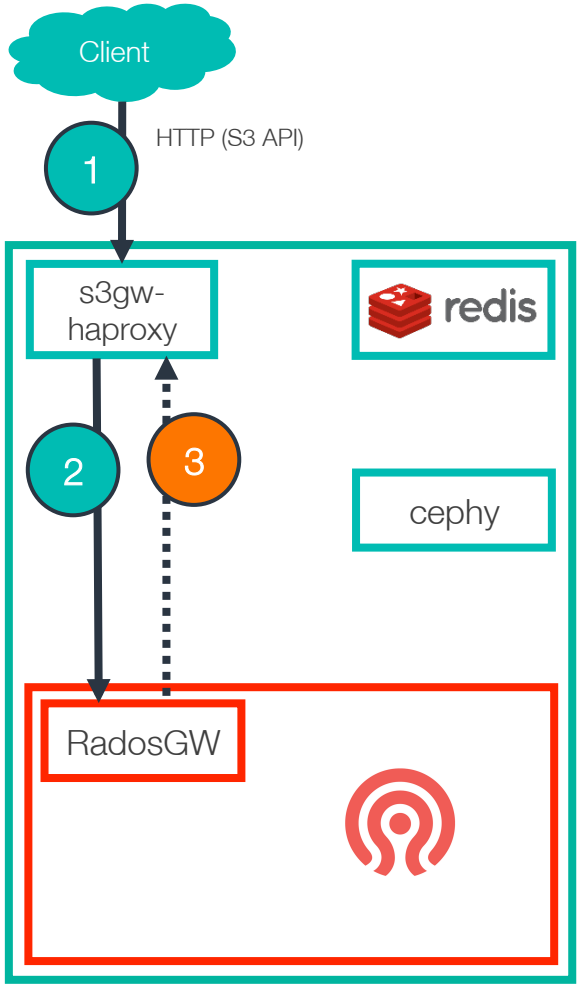
Replication in Action



1 `PUT` **`bucket/key`** (HAProxy)

2 `PUT` **`bucket/key`** (RadosGW)

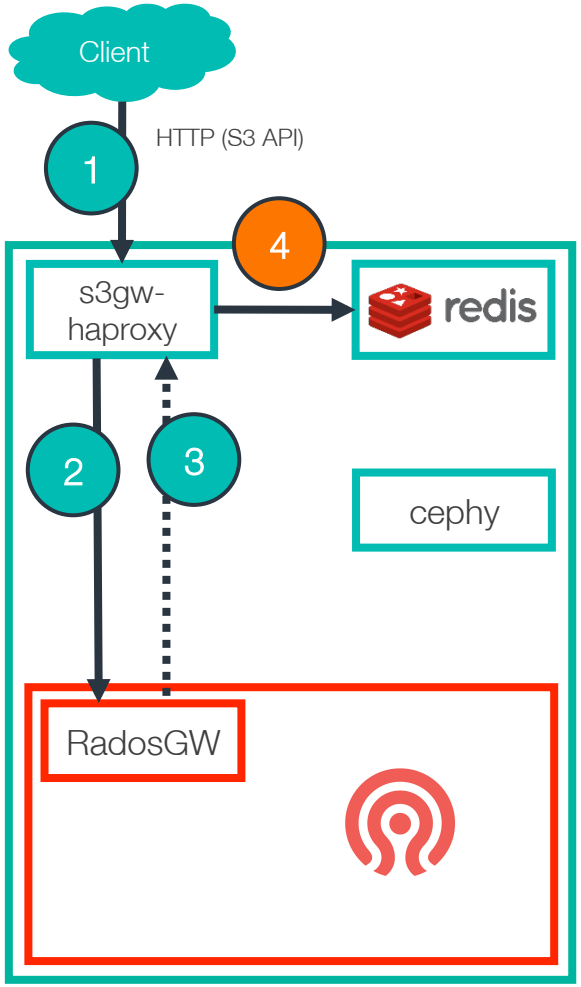
Replication in Action



Site A

1. `PUT` **`bucket/key`** (HAProxy)
2. `PUT` **`bucket/key`** (RadosGW)
3. Response Header `Status-Code 20x`

Replication in Action



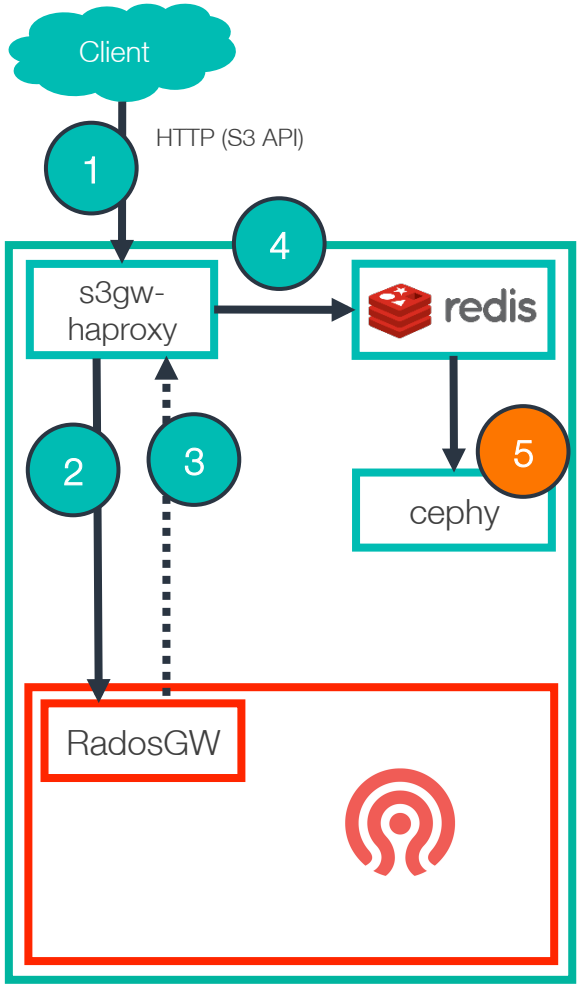
Site A

1. `PUT` **`bucket/key`** (HAProxy)
2. `PUT` **`bucket/key`** (RadosGW)
3. Response Header `Status-Code 20x`
4. Publish notification

```
LPUSH

s3notifications:bucket
{
    "event": "s3:ObjectCreated:Put",
    "objectKey": "key"
}
```

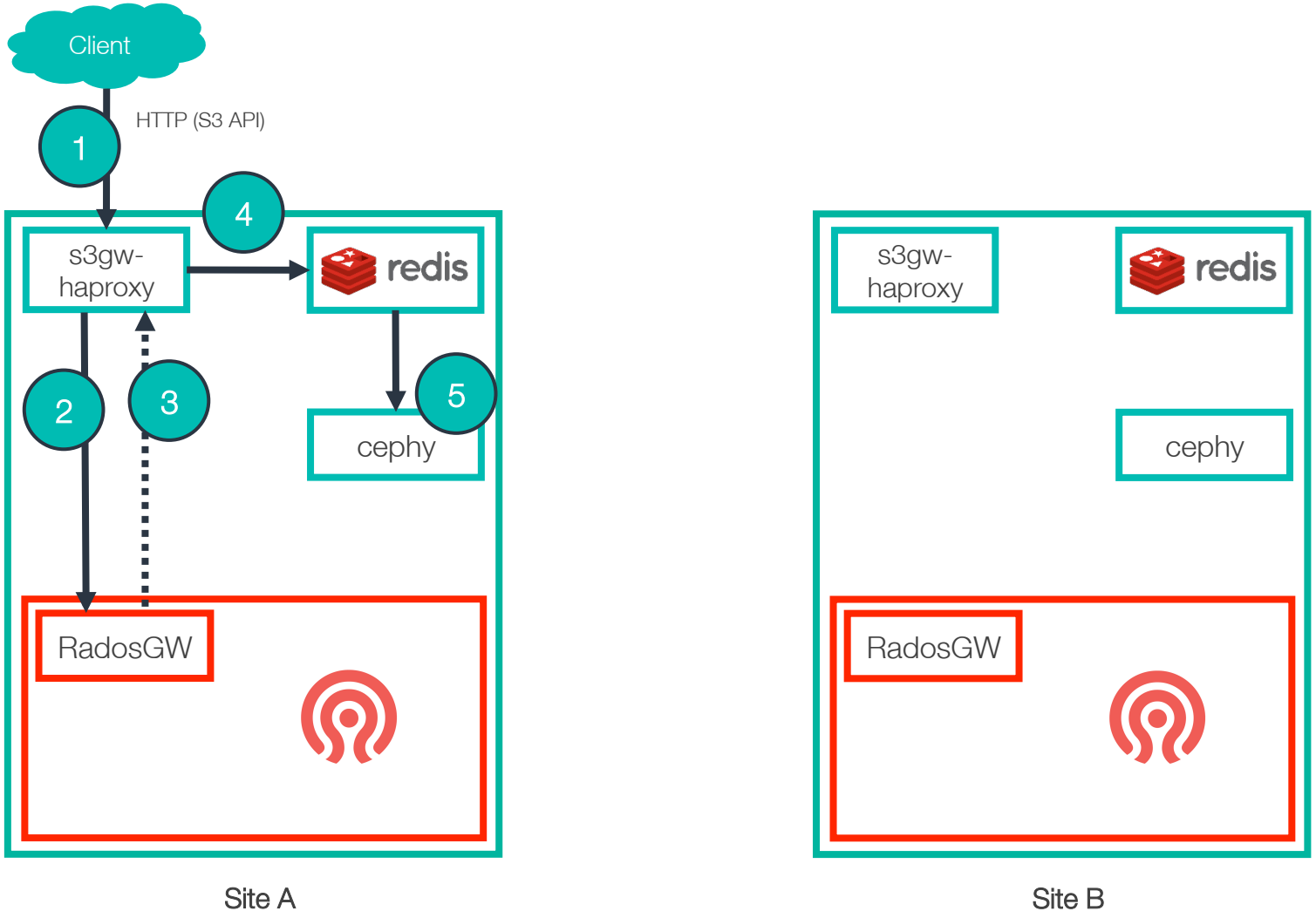
Replication in Action



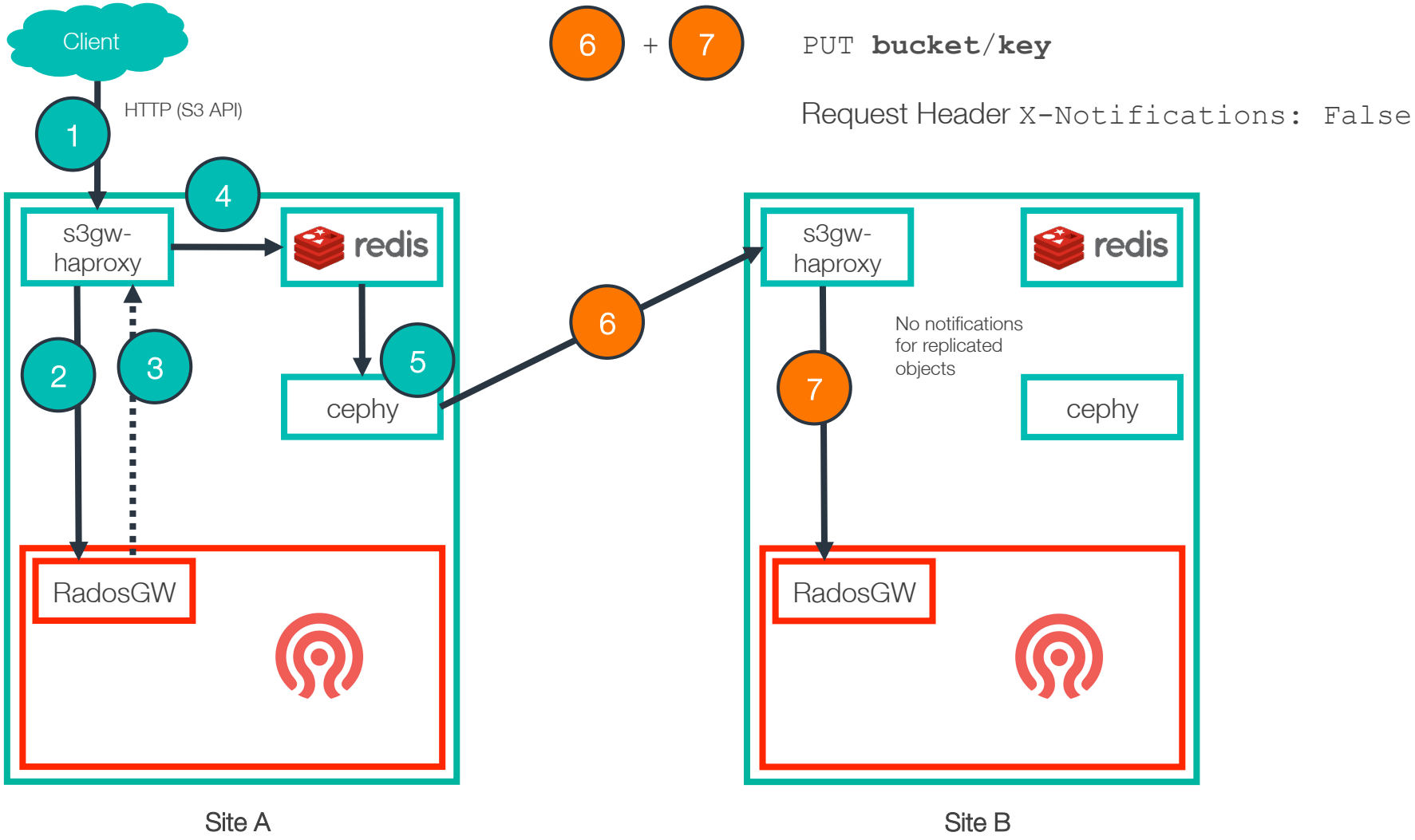
Site A

1. `PUT` **`bucket/key`** (HAProxy)
2. `PUT` **`bucket/key`** (RadosGW)
3. Response Header `Status-Code 20x`
4. Publish notification
5. Consume notification and replicate to all given targets

Replication in Action



Replication in Action



Geo-replication

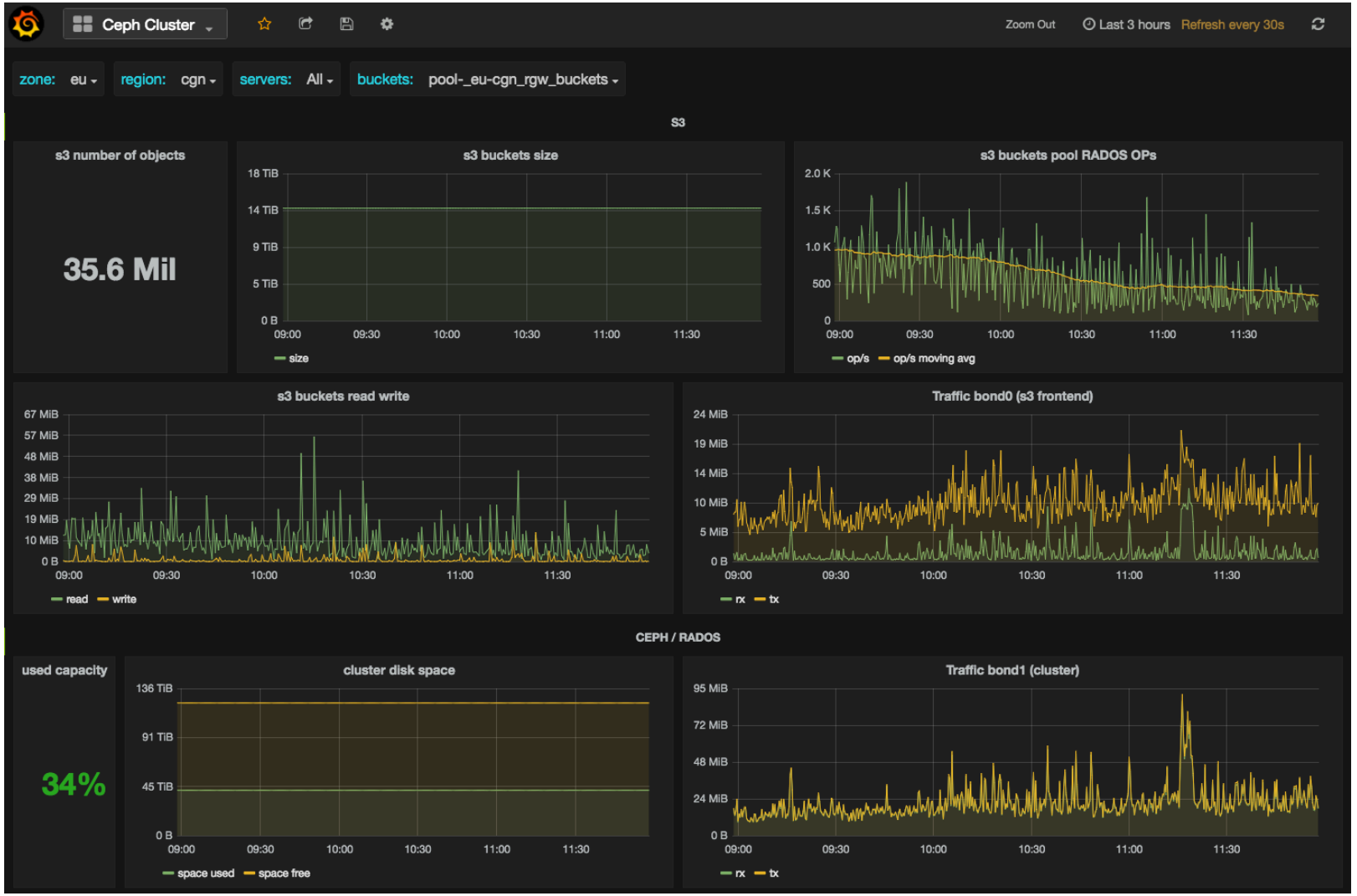
Roadmap

- More servers / locations / replication targets
- 100M+ objects
- Global gateway
  - “Object inventory”
  - More intelligent scrubbing
  - Location awareness
  - Easily bootstrap a new cluster
  - Statistics
- Potentially make `cephy` open source

Lessons learned

Lessons learned

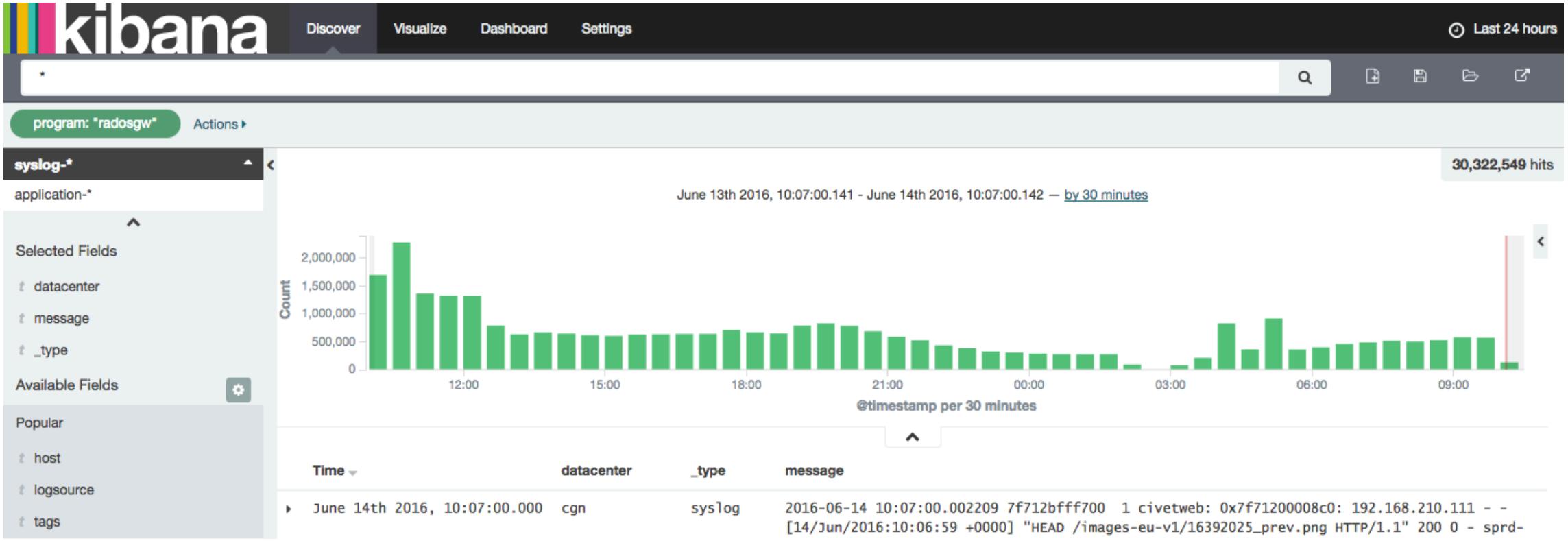
- Grafana is awesome!



Lessons learned

RadosGW is quite chatty btw.

- Kibana is awesome, too!



Lessons learned

- Upgrades can fail
  - Build scripts to get a dummy setup and test it in QA
  - Story of a zombie cluster

Lessons learned

- Move index data to SSD*

```
ceph osd crush add-bucket ssd root

ceph osd crush add-bucket stor01_ssd host

ceph osd crush move stor01_ssd root=ssd


ceph osd crush rule create-simple ssd_rule ssd host


ceph osd pool set .eu-cgn.rgw.buckets.index crush_ruleset 1
```

CRUSH rule numbers are tricky

* https://www.sebastien-han.fr/blog/2014/01/13/ceph-managing-crush-with-the-cli/

Lessons learned

- Know your libraries and tools
  - Pain points:
    - Ceph Object Gateway (before Jewel) only supports V2 signatures
      - Official SDKs from AWS: Java, Go, …
      - Fun with Python Boto2/3
    - Custom (other than AWS) S3 endpoints might not be supported or even ignored/ overridden
    - Beware of file system wrappers (e.g. Java `FileSystemProvider`)
      - They do funny things that you don't necessarily expect
        e.g. `DELETE` requests when copying files with `REPLACE_EXISTING` option
      - Can cause much more requests than you might think

Lessons learned

- Plan your scrubbing
  - 35.2M objects / 4.400 PGs = 8.000 files per scrub task
  - Distribute deep scrubs over a longer timeframe

    `osd deep scrub interval = 2592000` (30 days)

  - Watch it closely

```
ceph pg dump pgs -f json | jq .
[].last_deep_scrub_stamp | tr -d '"' | awk '{print
$1}'| sort -n | uniq -c
```

Questions?

Thank You

ajaz@spreadshirt.net

jns@spreadshirt.net

Links

- http://ceph.com/
- https://github.com/spreadshirt/s3gw-haproxy
- http://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html
- http://redis.io/
- http://www.haproxy.org/
- https://www.sebastien-han.fr/blog/2014/01/13/ceph-managing-crush-with-the-cli/