BlkKin: A Low-overhead tracing infrastructure for software-defined storage systems

Marios-Evangelos Kogias

National Technical University of Athens
School of Electrical and Computer Engineering

October 14, 2014

Outline

Thesis Background

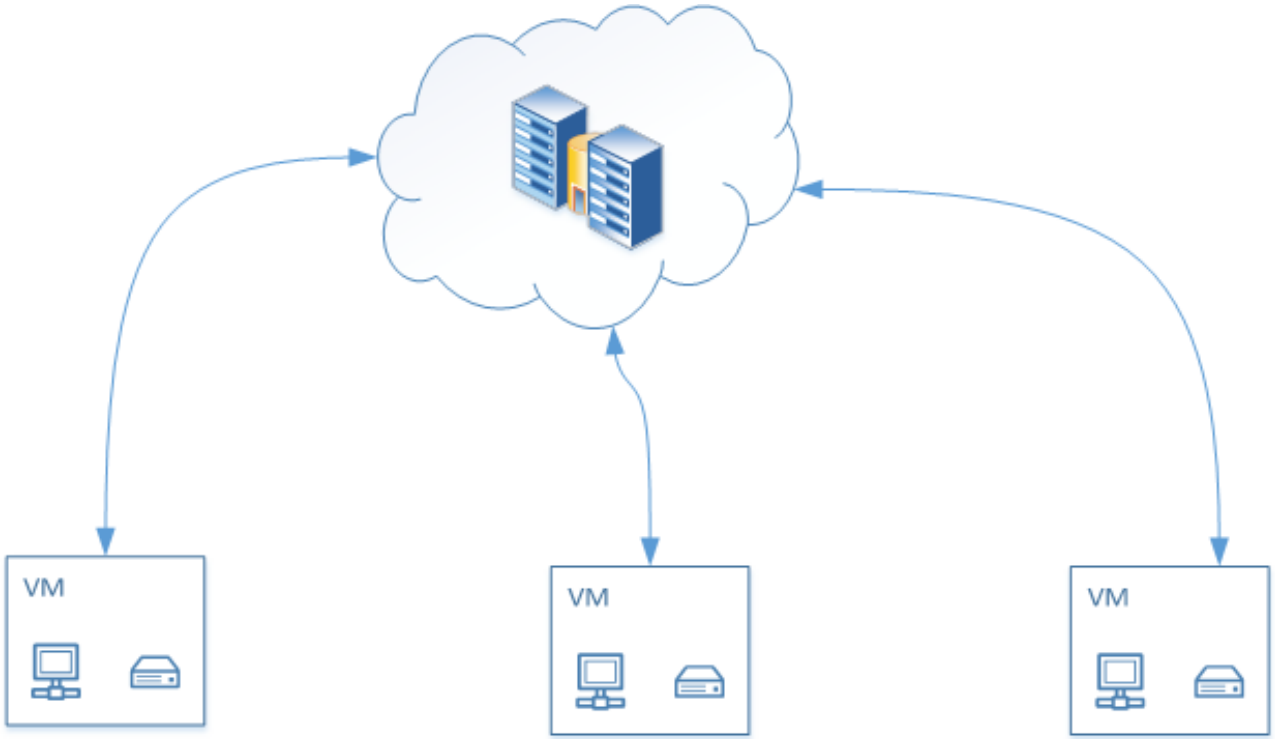
synnefo

Open source, production-ready, cloud software.
Designed since 2010 by GRNET.

okeanos

- IaaS service
- Targeted at the Greek Academic and Research Community
- Designed by GRNET
- In production since 2011

VM Volume storage

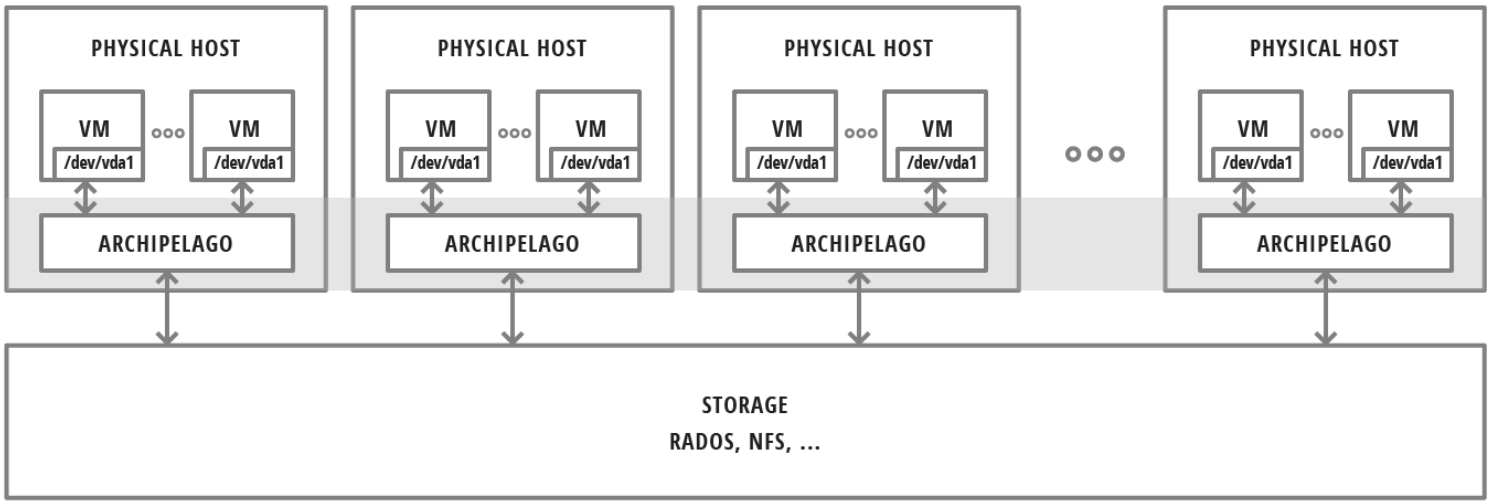


Archipelago I

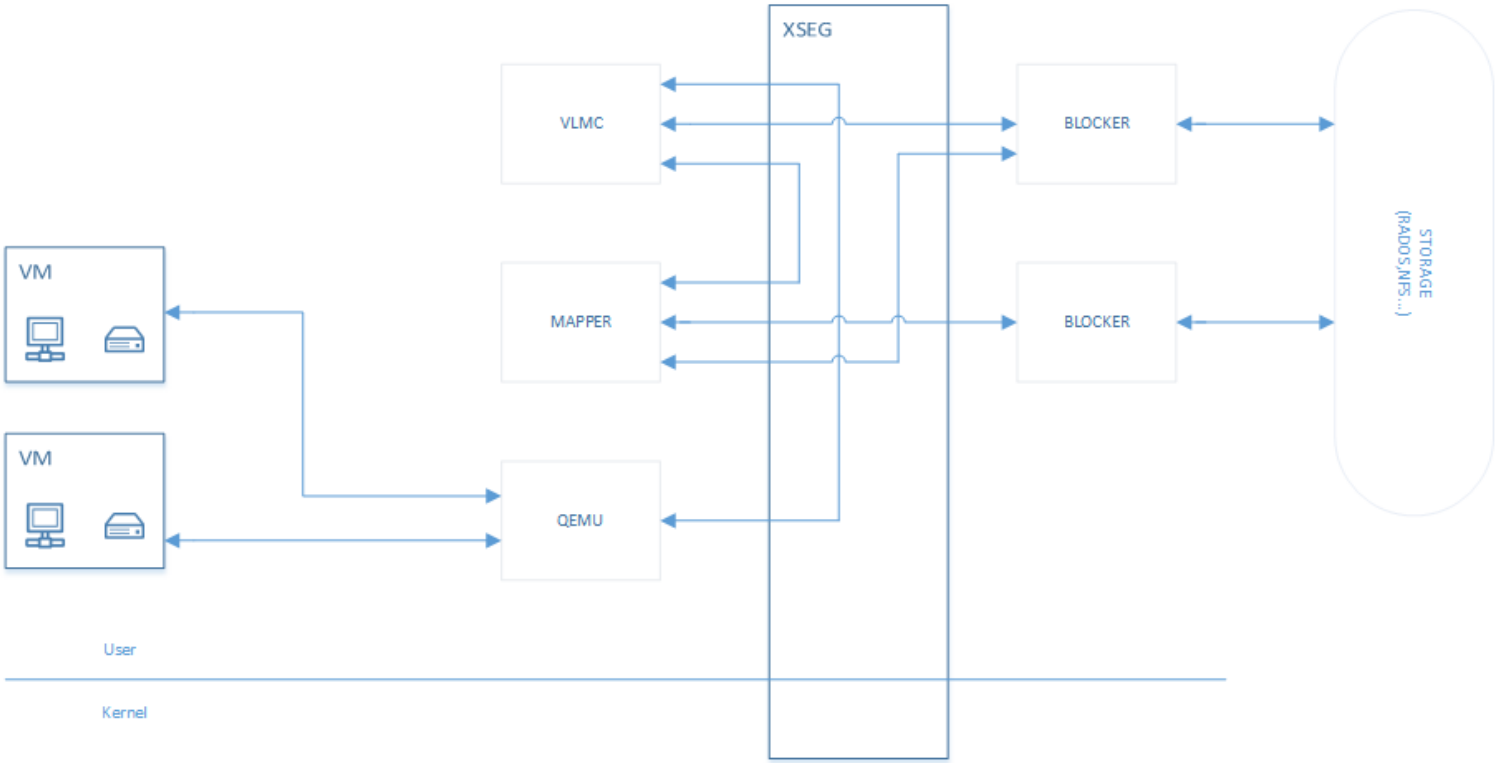
A thin distributed storage layer aiming to:

- Decouple storage logic from the actual data store
- Provide logic for thin cloning and snapshotting
- Provide logic for deduplication
- Provide different endpoint drivers to access Volumes and Files
- Provide backend drivers for different storage technologies

Archipelago II



Archipelago III



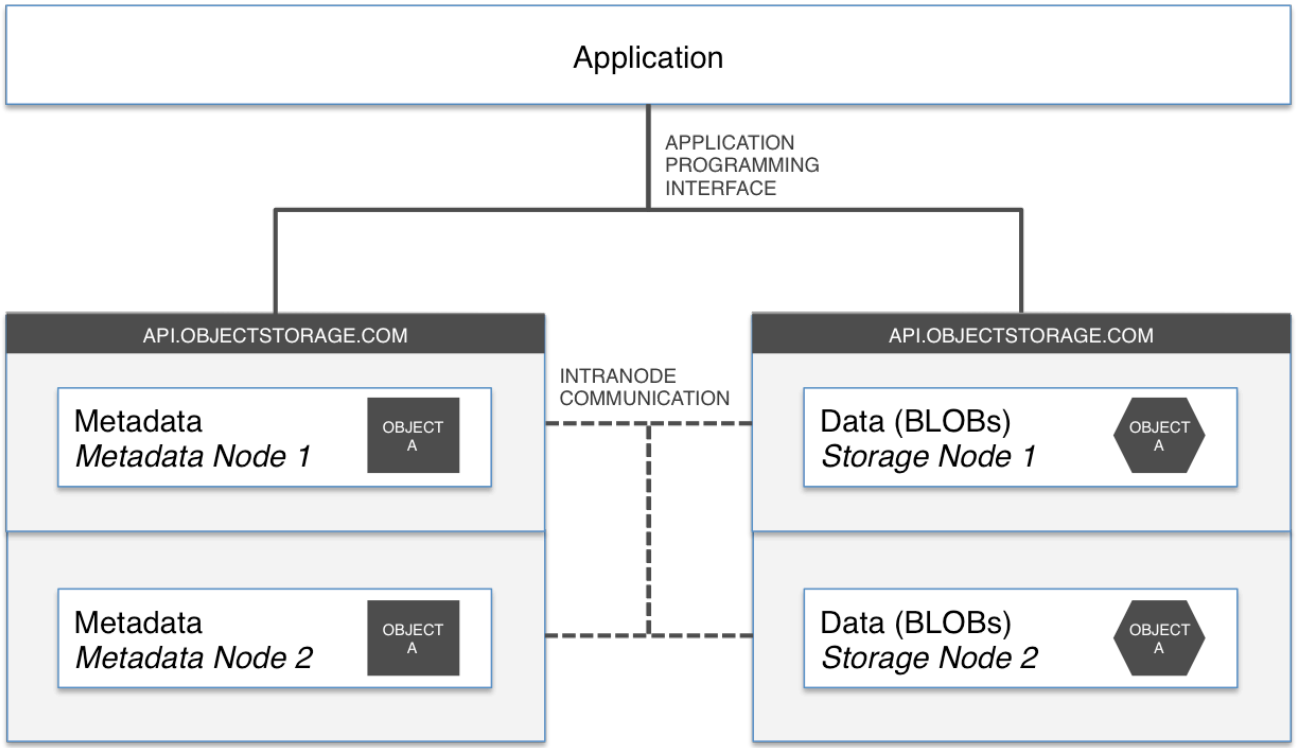
RADOS

is the block storage component of Ceph

RADOS basic characteristics are:

- *Replication*
- *Fault tolerance*
- *Self-management*
- *Scalability*

Storage Abstraction



The Problem

- Complex service oriented architectures
- Difficult debugging
- Difficult monitoring
- Non-deterministic execution
- Context-bound faults

Solution

Distributed end-to-end tracing

&

Central data collection

BlkKin

A distributed tracing infrastructure to track the IO request from Qemu until RADOS

BlkKin main characteristics:

- low-overhead tracing
- live-tracing
- End-to-end tracing of causal relationships
- User interface

Main Challenges

- Meaningful and easily correlated tracing data
- Low overhead tracing backend

Schools of thought

black-box schemes

They assume there is no additional information other than the message record described above and use statistical regression techniques to infer that association.

annotation-based schemes

They rely on applications or middleware to explicitly tag every record with a global identifier that links these message records back to the originating request.

The Dapper System

- Large scale distributed systems tracing infrastructure created by Google
- Annotation-based tracing scheme
- Common libraries instrumentation
  - RPC System
  - Control Flow
- BigTable backend
- Closed-source

Dapper tracing concepts

annotation	The actual information being logged. Either *timestamp* or *key-value*
span	The basic unit of the process tree. Can represent a subsystem or a function call. To depict causal relationship each span has a parent span or is a *root* span.
trace	A different trace id is used to group data related to he same initial request

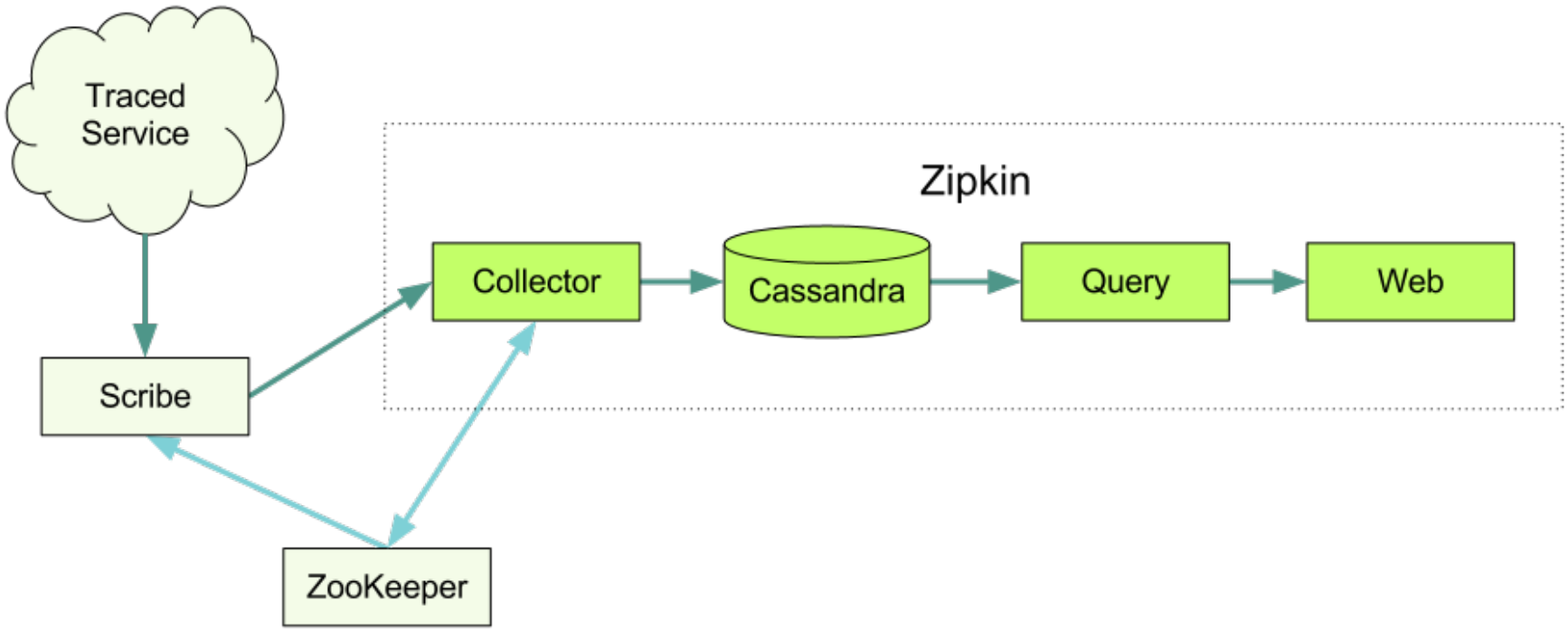
Zipkin

An open-source Scala implementation of the Dapper paper by Twitter

Zipkin services:

- Data collector
- Database service
- Web UI

Zipkin Architecture



Scribe

Scribe is a scalable and reliable logging server created by Facebook

- Written in C++
- Directed graph architecture
- Batch messaging
- HDFS support
- Based on Apache Thrift

Thrift

A software framework for scalable cross-language services development.

Includes a code generation engine to create RPC services across programming languages based on a Thrift file

Sample target languages: C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, OCaml

Zipkin sum up

Zipkin	is a full stack tracing system using
Scribe	as its logging server using
Thrift	as its transport protocol

Tracing

> "Tracing is a specialized use of logging to record information about a program's execution"
>
> Wikipedia

Tracing characteristics:

- Tracing can be low level (eg. kernel tracing, access to performance counters)
- Tracing has mostly debugging purposes and performance tuning
- Tracing may produce notoriously bulky output

Tracing Systems

DTrace

Released by Sun Microsystems in 2005

SystemTap

Released by Red Hat in 2005

Advantages:

- Dynamic Instrumentation
- User and kernel tracing

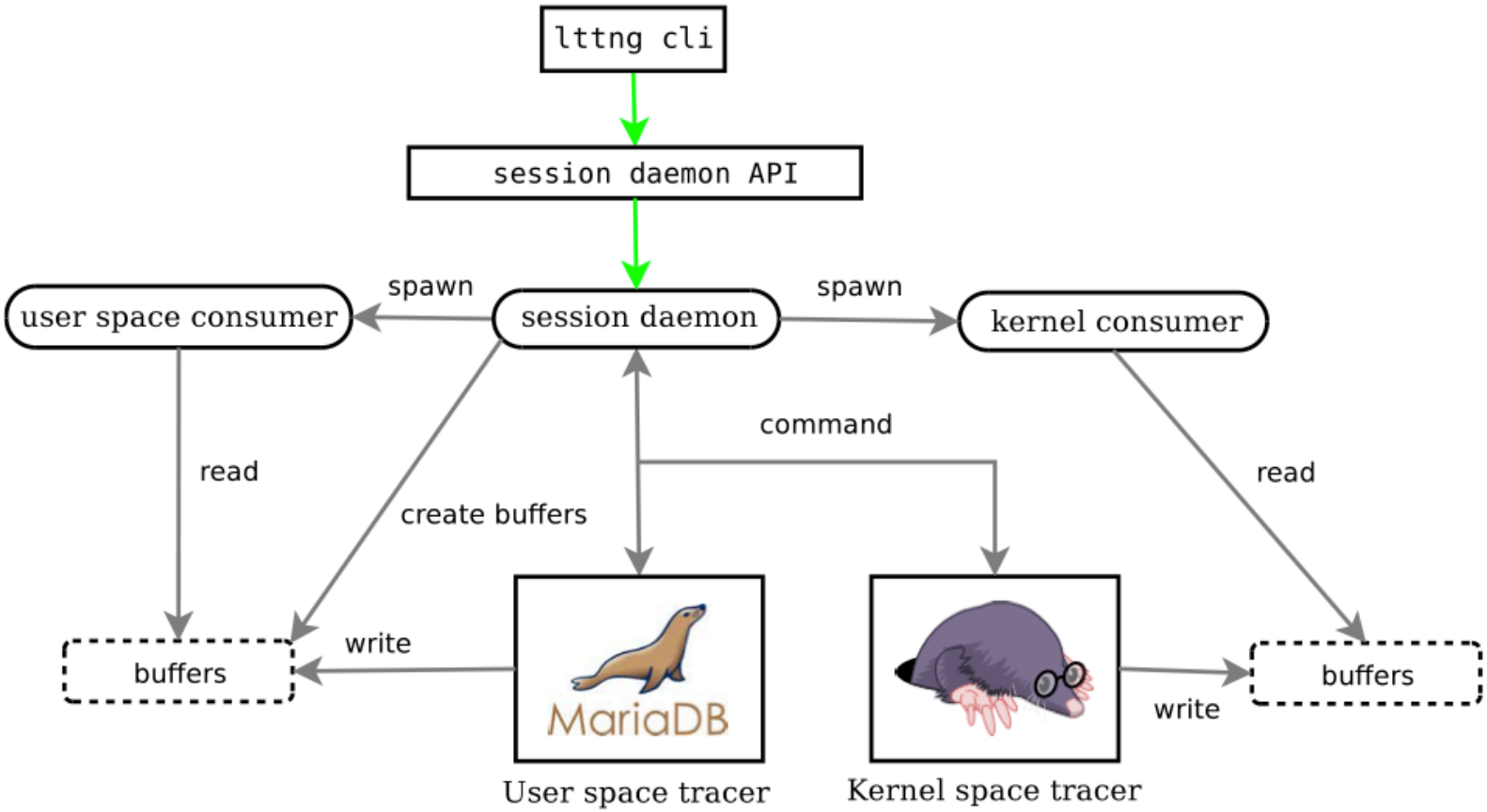
Disadvantages:

- User tracing is based or system calls or breakpoints
- Significant performance overhead
- Inappropriate for live tracing

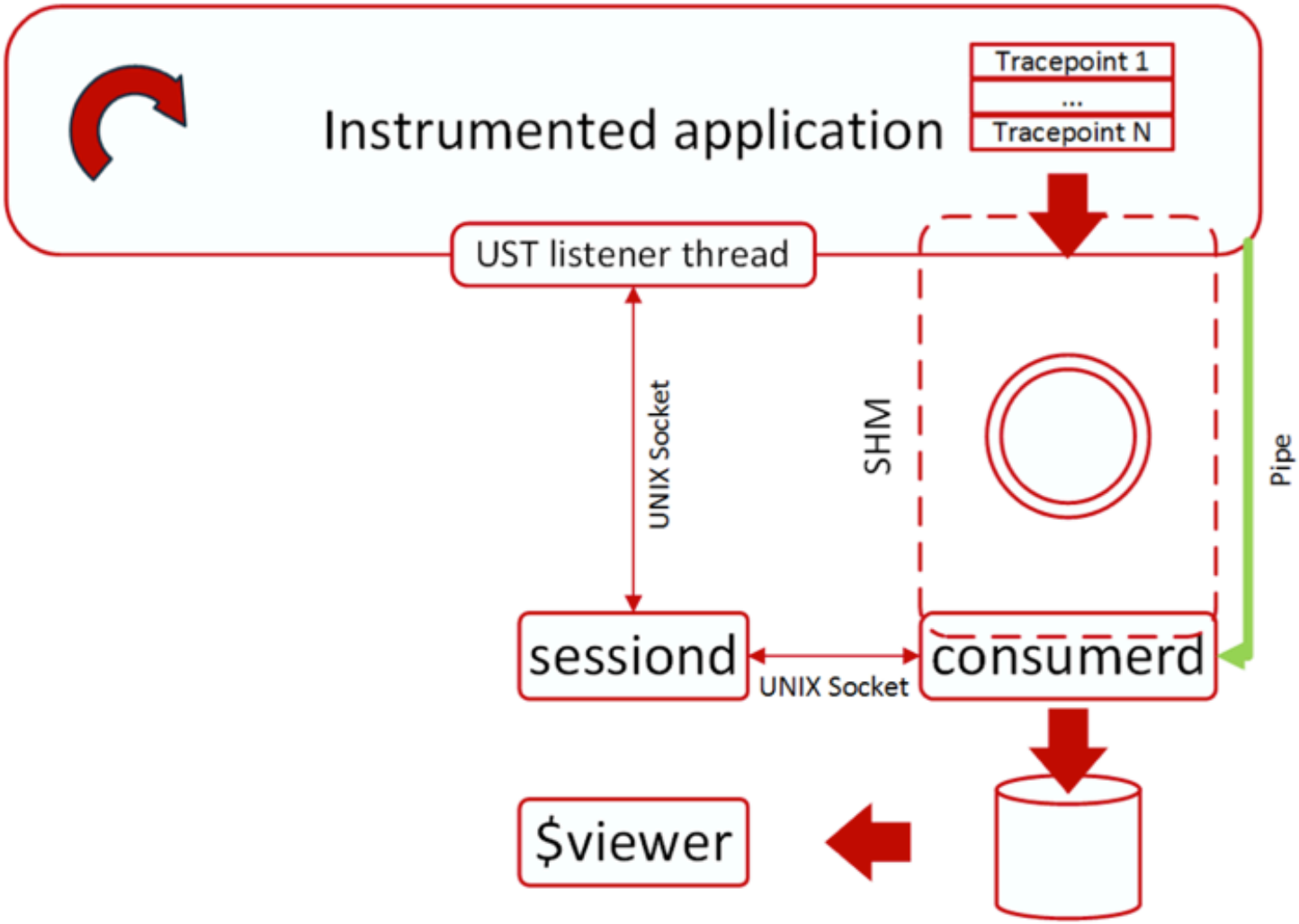
Linux Trace Toolkit - next generation

- successor of Linux Trace Toolkit
- Mathew Desnoyers PhD dissertation in Ecole Polytechnique de Montreal
- maintained by EfficiOS Inc1and the DORSAL lab in Ecole Polytechnique de Montreal.
- Unified user and kernel tracing
- Low overhead tracing based on Tracepoints
- Static instrumentation
- Live tracing

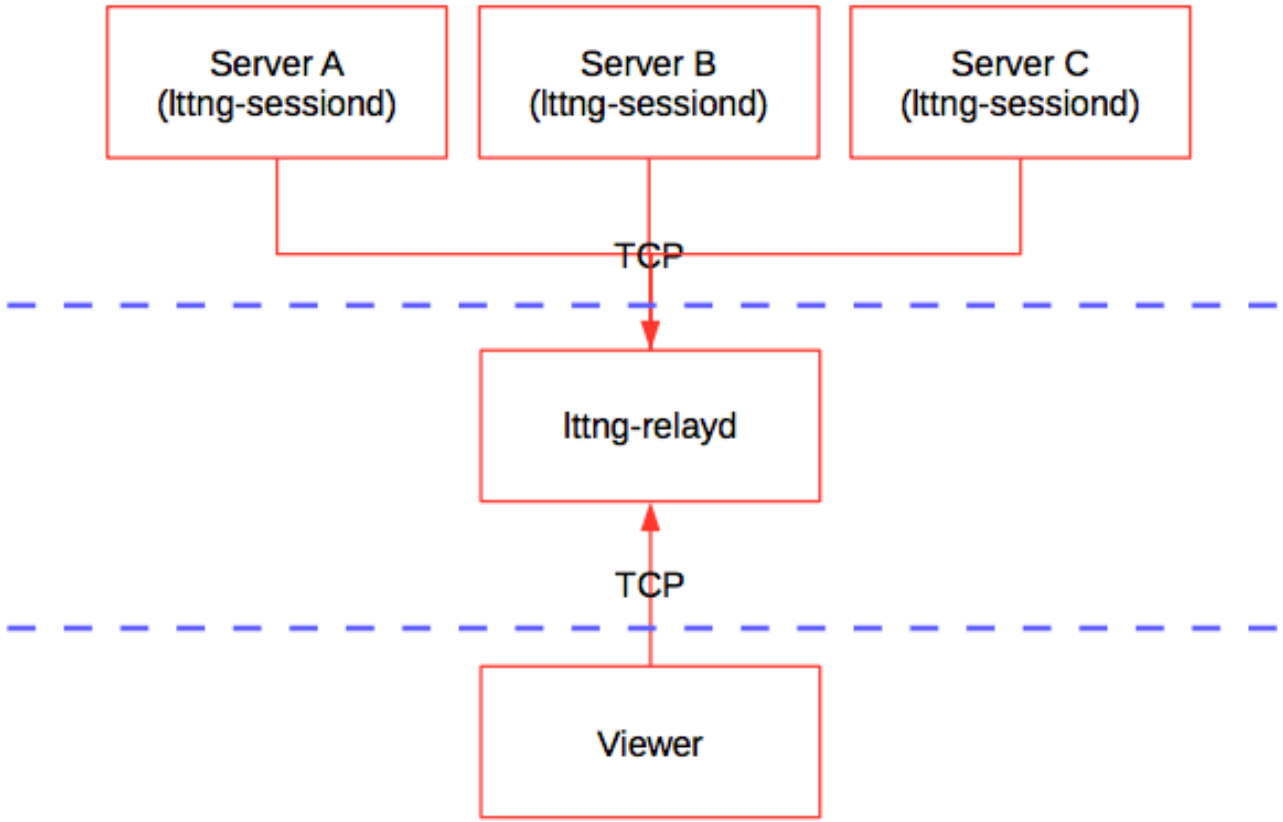
LTTng Architecture



UST architecture



LTTng live-tracing



CTF

Common Trace Format:

- Aimed to cover tracing needs from versatile communities
- Collaboration between the Multicore association and the Linux Community
- Based on Trace Stream Description Language
- Separation between data and metadata
- Separation between event and event-context
- Variety of data types and type inheritance
- Live tracing

Babeltrace

- Trace reader/writer
- Trace converter
- Babeltrace plugins
- Exports `libbabeltrace`
- Python bindings

Complete environments based on Babeltrace: Trace Compass (Java tool), Eclipse LTTng plugin

BlkKin = Block Storage + Zipkin

BlkKin is:

- end-to-end tracing infrastructure
- implementing Dapper semantics
- based on Zipkin and LTTng
- providing live tracing

BIkKin Contribution

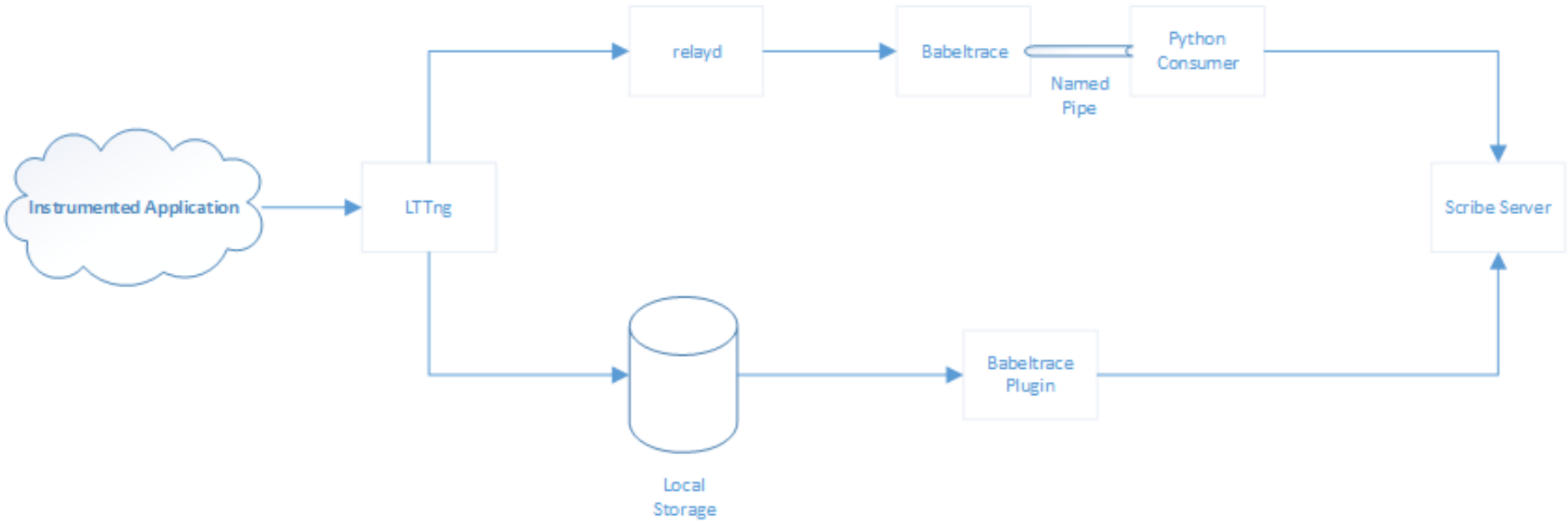
BIkKin instrumentation library:

- C/C++ instrumentation library
- implementing Dapper tracing semantics
- LTTng backend
- sampling

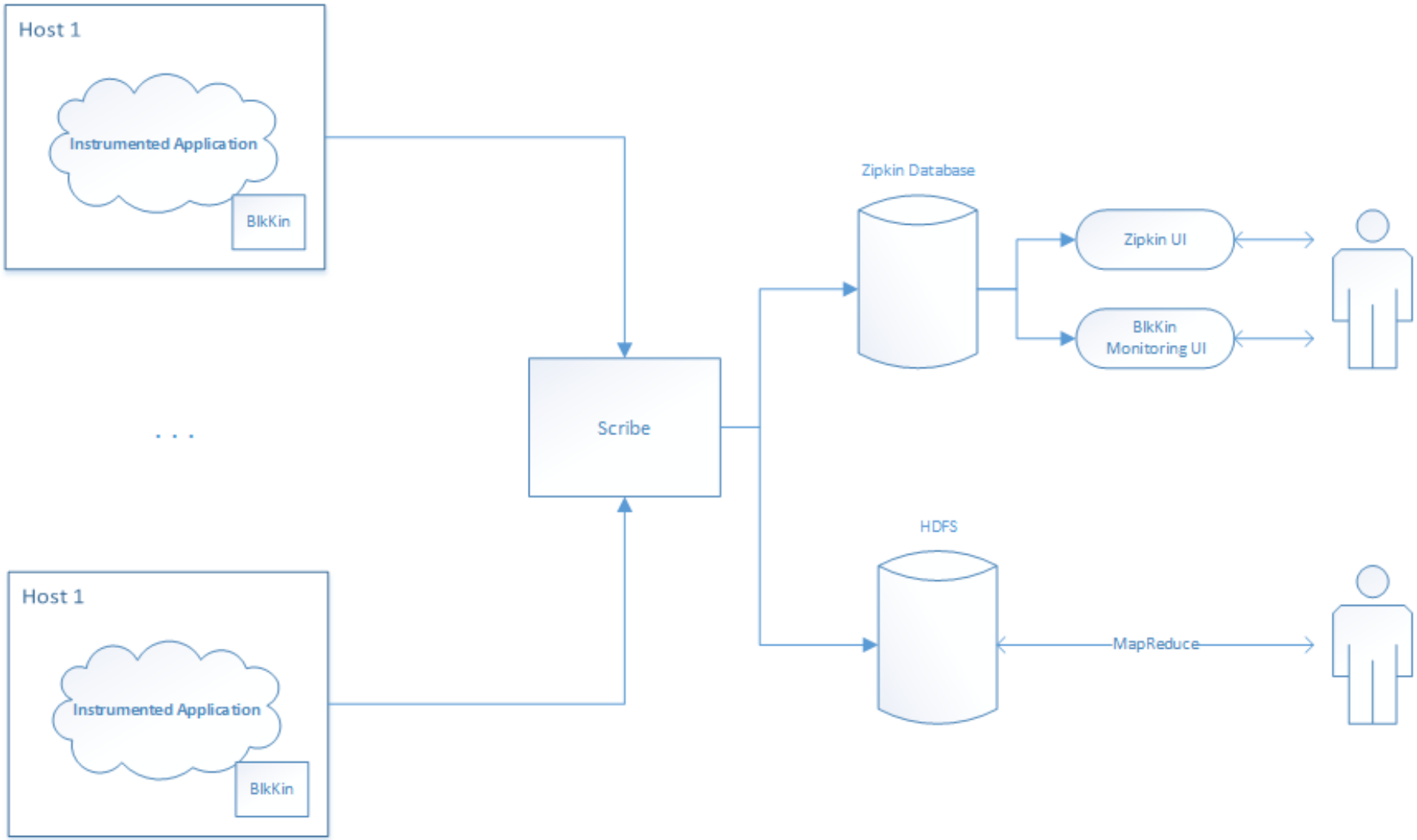
CTF-to-Scribe Babeltrace plugins:

- based on Python bindings
- two output formats:
  - JSON format (generic)
  - Zipkin Thrift (Zipkin specific)

BlkKin Architecture



BlkKin Architecture



Evaluation Environment

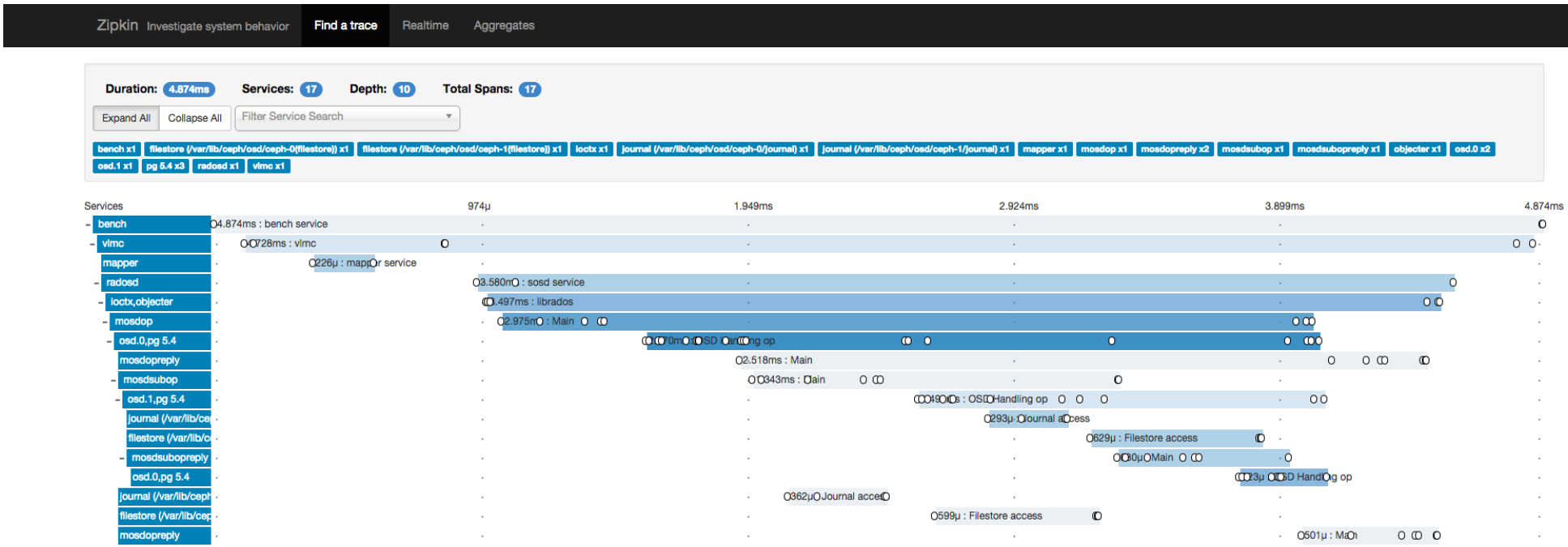
Instrumented:

- QEMU Archipelago driver
- Archipelago and libxseg
- RADOS

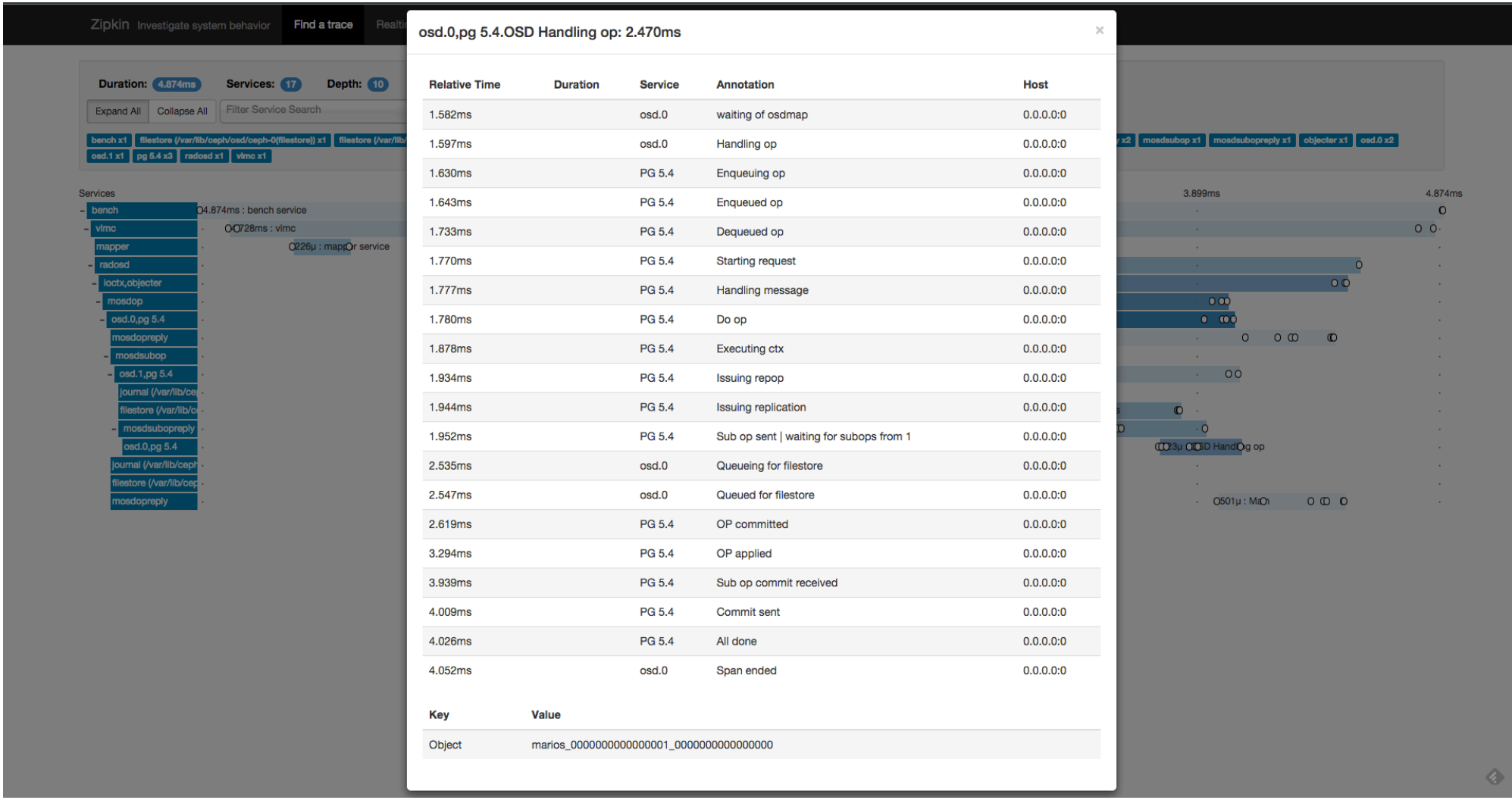
Testbed:

- 2 hosts LAN interconnected
- 2 OSDs per host
- 1 host running QEMU and Archipelago

Zipkin UI


Zipkin
Investigate system behavior
Find a trace
Realtime
Aggregates
Duration: 4.874ms
Services: 17
Depth: 10
Total Spans: 17
Expand All
Collapse All
Filter Service Search
bench x1
filestore (/var/lib/ceph/osd/ceph-0/filestore)) x1
filestore (/var/lib/ceph/osd/ceph-1/filestore)) x1
ioctx x1
journal (/var/lib/ceph/osd/ceph-0/journal) x1
journal (/var/lib/ceph/osd/ceph-1/journal) x1
mapper x1
mosdop x1
mosdopreply x2
mosdsubop x1
mosdsubopreply x1
objecter x1
osd.0 x2
osd.1 x1
pg 5.4 x3
radosd x1
vlmc x1
Services
974µ
1.949ms
2.924ms
3.899ms
4.874ms
bench
4.874ms : bench service
vlmc
728ms : vlmc
mapper
226µ : mapper service
radosd
3.580m : sosd service
ioctx,objecter
.497ms : librados
mosdop
2.975m : Main
osd.0,pg 5.4
mosdopreply
2.518ms : Main
mosdsubop
343ms : Main
osd.1,pg 5.4
journal (/var/lib/ce
filestore (/var/lib/c
mosdsubopreply
osd.0,pg 5.4
journal (/var/lib/ceph
filestore (/var/lib/ce
mosdopreply
293µ : Journal access
629µ : Filestore access
362µ Journal acce
599µ : Filestore access

Zipkin UI - Annotations

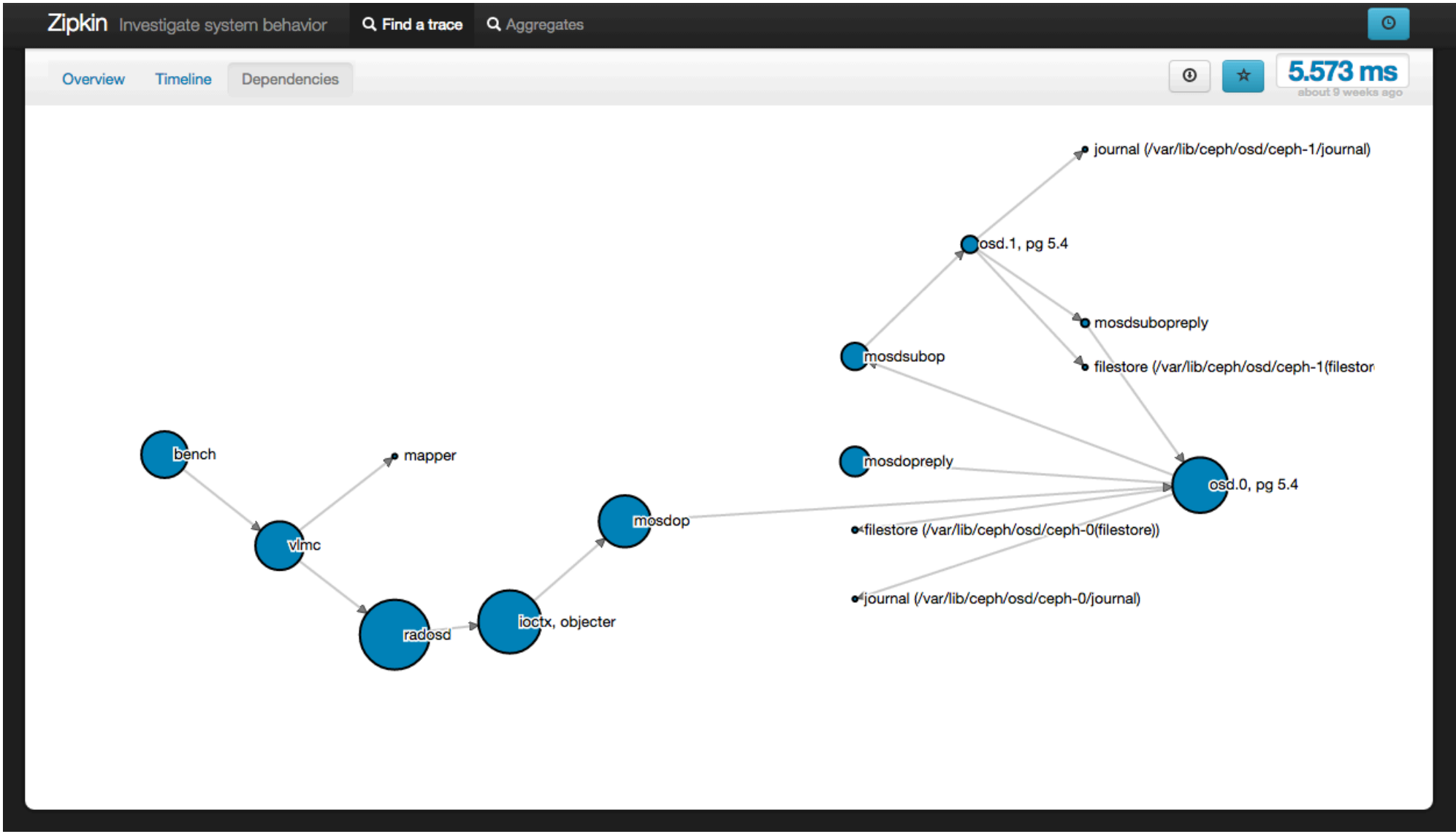


osd.0,pg 5.4.OSD Handling op: 2.470ms

Relative Time	Duration	Service	Annotation	Host
1.582ms		osd.0	waiting of osdmap	0.0.0.0:0
1.597ms		osd.0	Handling op	0.0.0.0:0
1.630ms		PG 5.4	Enqueuing op	0.0.0.0:0
1.643ms		PG 5.4	Enqueued op	0.0.0.0:0
1.733ms		PG 5.4	Dequeued op	0.0.0.0:0
1.770ms		PG 5.4	Starting request	0.0.0.0:0
1.777ms		PG 5.4	Handling message	0.0.0.0:0
1.780ms		PG 5.4	Do op	0.0.0.0:0
1.878ms		PG 5.4	Executing ctx	0.0.0.0:0
1.934ms		PG 5.4	Issuing repop	0.0.0.0:0
1.944ms		PG 5.4	Issuing replication	0.0.0.0:0
1.952ms		PG 5.4	Sub op sent | waiting for subops from 1	0.0.0.0:0
2.535ms		osd.0	Queueing for filestore	0.0.0.0:0
2.547ms		osd.0	Queued for filestore	0.0.0.0:0
2.619ms		PG 5.4	OP committed	0.0.0.0:0
3.294ms		PG 5.4	OP applied	0.0.0.0:0
3.939ms		PG 5.4	Sub op commit received	0.0.0.0:0
4.009ms		PG 5.4	Commit sent	0.0.0.0:0
4.026ms		PG 5.4	All done	0.0.0.0:0
4.052ms		osd.0	Span ended	0.0.0.0:0

Key	Value
Object	marios_0000000000000001_0000000000000000

Zipkin UI - Dependencies



IO Loads - Scenarios

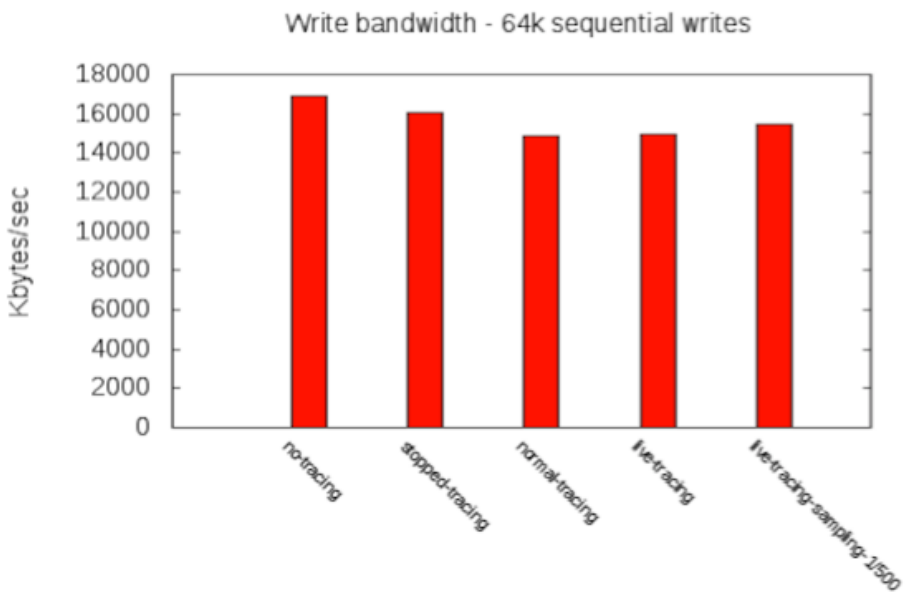
IO loads created within VM using `fio`:

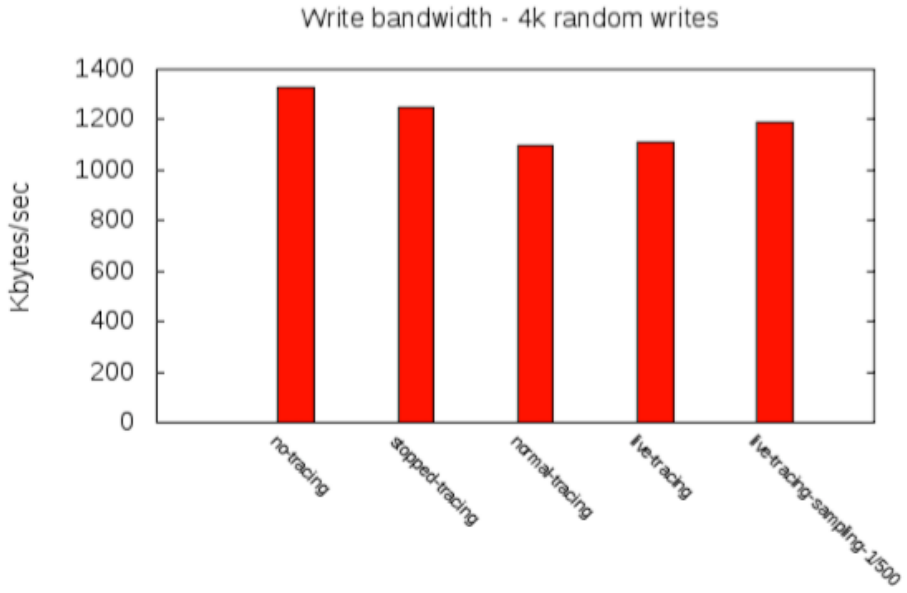
- 4k random writes
- 64k sequential writes

Scenarios:

- no tracing
- stopped tracing
- normal tracing
- live tracing without sampling
- live tracing with 1/500 sampling

Bandwidth Overhead

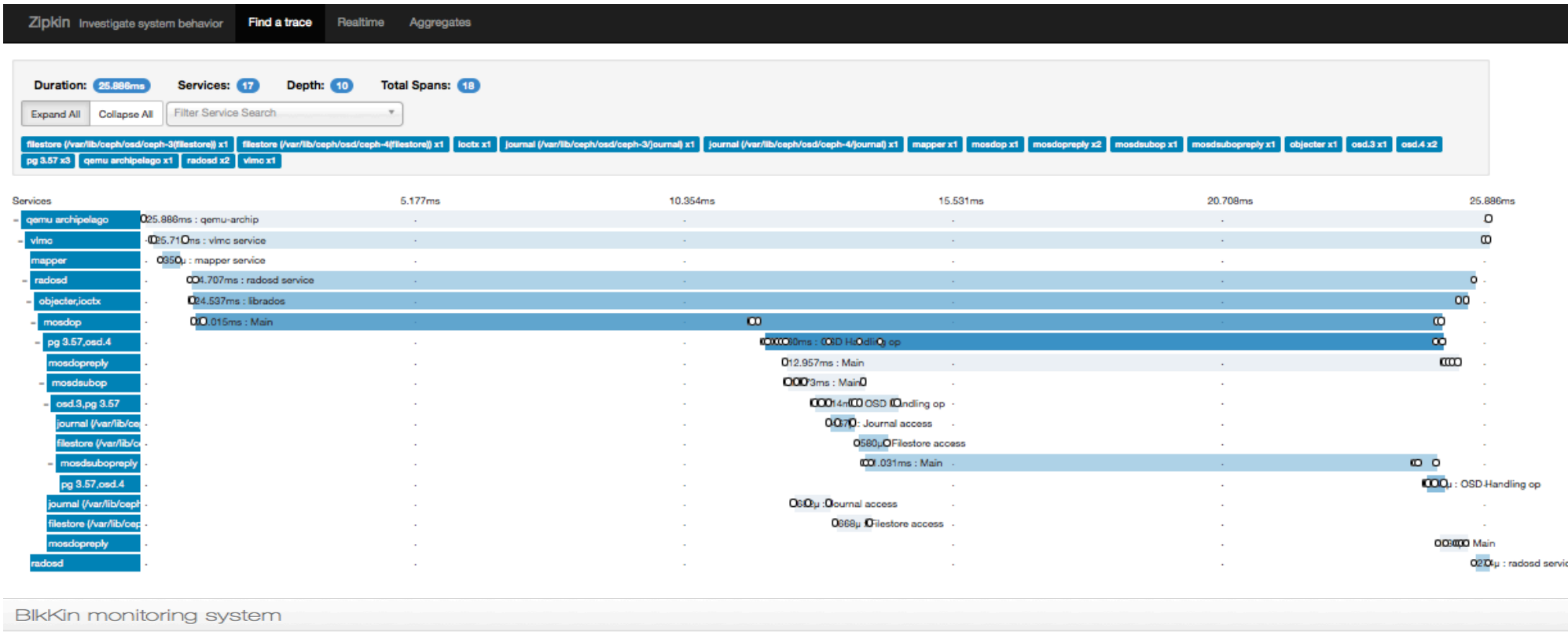




Fault injection

1. Calculate threshold values using MapReduce in HDFS
2. Simulate faulty environment:
   - Network faults: using `tc`
   - Disk faults: adding extra IO load using `fio`
3. Detect faults using BlkKin monitoring UI

Network fault



BlkKin monitoring system

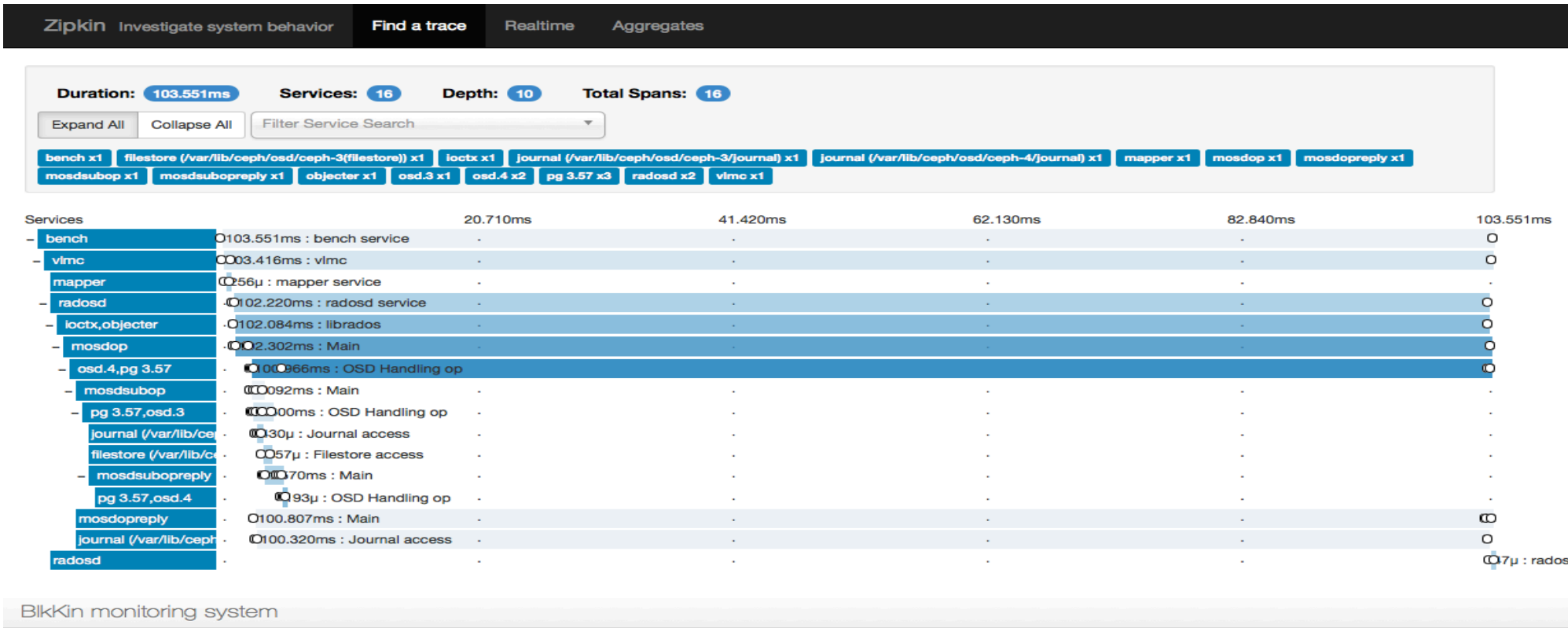
Network monitor

Type	Value
Communication time	3960.1623

OSD monitor

OSD Name	Journal Access
ceph-3	329.708
ceph-4	284.5664

Disk fault



Network monitor

Type	Value
Communication time	298.8313

OSD monitor

OSD Name	Journal Access
ceph-3	303.2
ceph-4	29692.8947

Summarizing

BlkKin:

- end-to-end tracing infrastructure
- based on Zipkin and LTTng
- targeting storage systems tracing
- used in QEMU, Archipelago and RADOS instrumentation

Questions?

"Judge a man by his questions rather than by his answers."

—Voltaire

Thank you