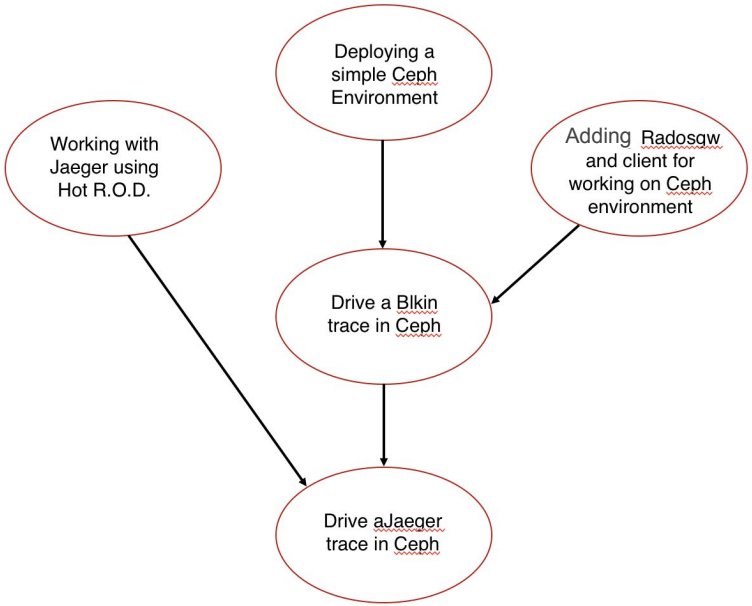
End to End tracing in Ceph

Project Developers : Golsana Ghaemi, Oindrilla Chatterjee, Aditya Singh, Bowen Song
Mentors : Mania Abdi, Raja Sambasivan, Peter Portante

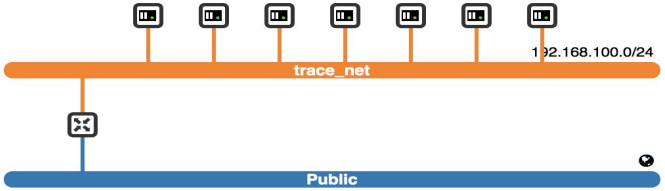
End Goal

Goal of the project: Tracing ceph using Jaeger

Outline :



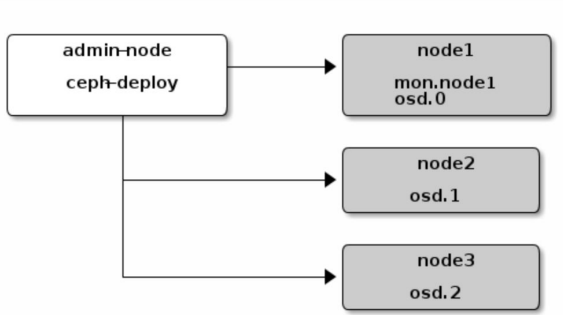
Ceph deployment



1. Created instances in Openstack : 7 Virtual machines, running CentOS
2. Created our simple ceph environment : Our ceph topology consists of one admin node, one monitor node and two OSDs
3. Why this simple environment?

   The purpose of the project is not to deploying ceph and trace it in a large environment. The goal here is to deploy a scaled down Ceph environment for having a simple testbed for understanding Blkin and Jaeger, and instrumenting Jaeger in it so that it can be scaled up further.

4. Problems we faced
   a. First using ubuntu for ceph for deployment!
   b. Ssh issues(maybe the bug of our centos image)
   c. Version of ceph : Luminous vs Jewel

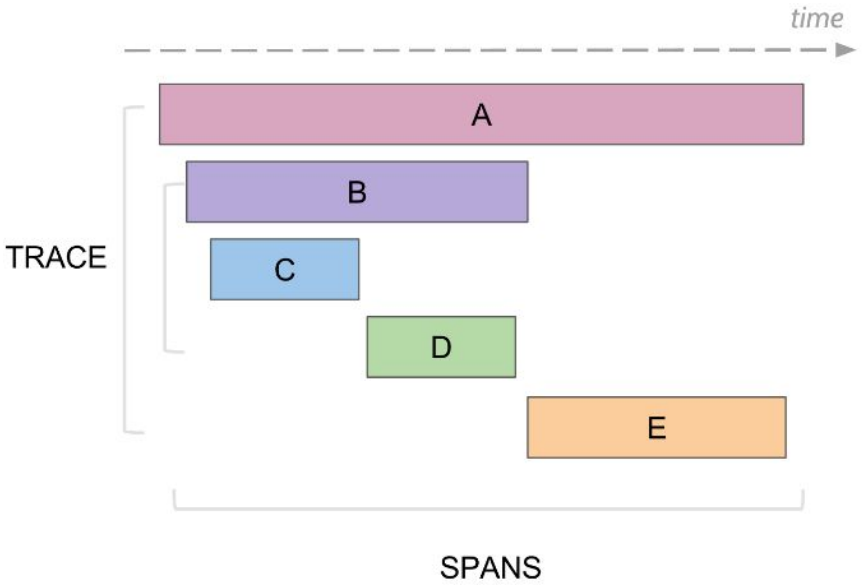


What is Tracing?

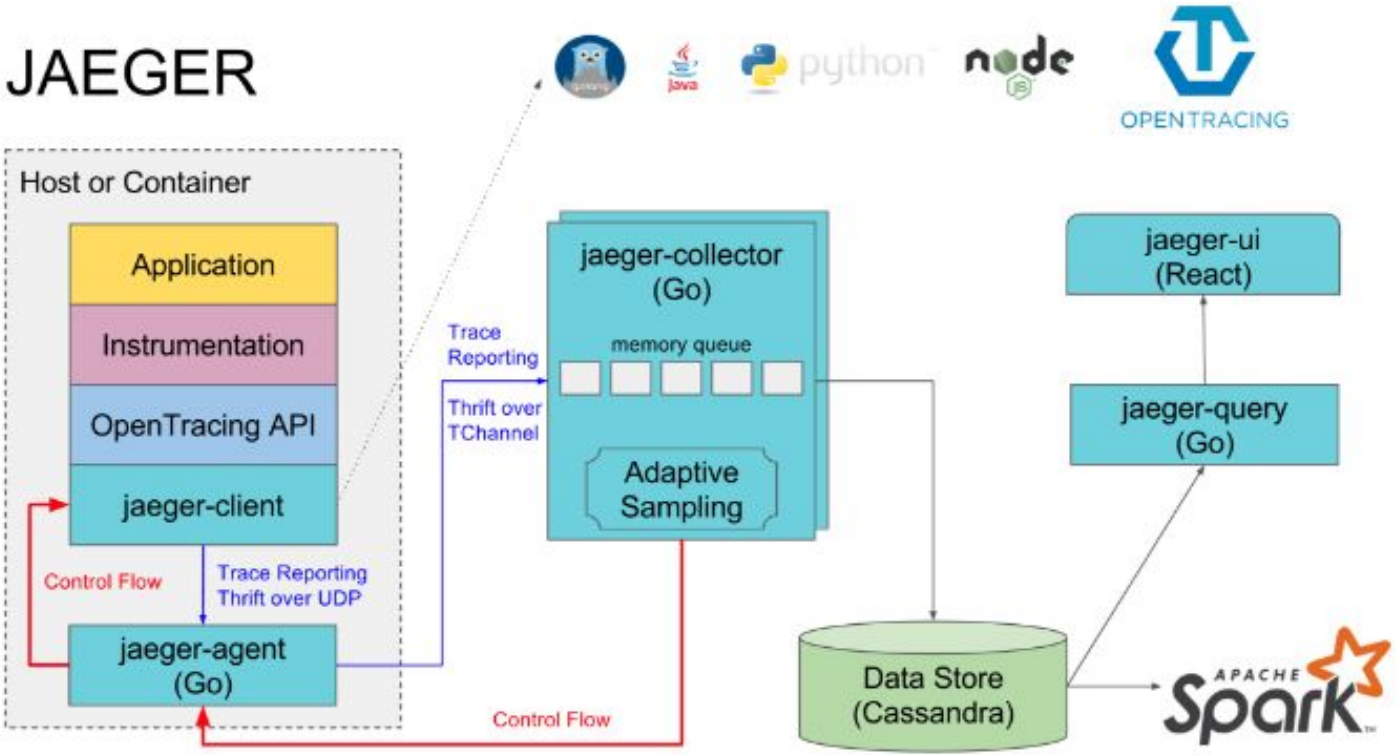
- It is a specialized use of logging to record information about a programs execution.
- Tracing is used by developers for debugging purposes and to solve common problems with software.
- In the case of Ceph which is a distributed storage system, with about a million lines of code, an effective tracing system is of utmost importance for developers.

Jaeger Architecture

- **Span**: A Span represents a logical unit of work in the system that has an operation name, the start time of the operation, and the duration. Trace is a directed acyclic graph of spans. Remote procedure call (RPC) is an example of a span.



Jaeger Architecture



Jaeger Architecture

Jaeger Client Libraries

- Jaeger clients are language specific implementations of the OpenTracing API. They can be used to instrument applications for distributed tracing either manually or with a variety of existing open source frameworks, such as Flask, Dropwizard, gRPC, and many more, that are already integrated with OpenTracing.

- An instrumented service creates spans when receiving new requests and attaches context information (trace id, span id, and baggage) to outgoing requests.

- Only ids and baggage are propagated with requests; all other information that compose a span like operation name, logs, etc. is not propagated. Instead sampled spans are transmitted out of process asynchronously, in the background, to Jaeger Agents.

Jaeger Architecture

Note that while all traces are generated, only few are sampled. Sampling a trace marks the trace for further processing and storage. By default, Jaeger client samples 0.1% of traces (1 in 1000), and has the ability to retrieve sampling strategies from the agent.

Agent

A network daemon that listens for spans sent over UDP, which it batches and sends to the collector. It is designed to be deployed to all hosts as an infrastructure component. The agent abstracts the routing and discovery of the collectors away from the client.

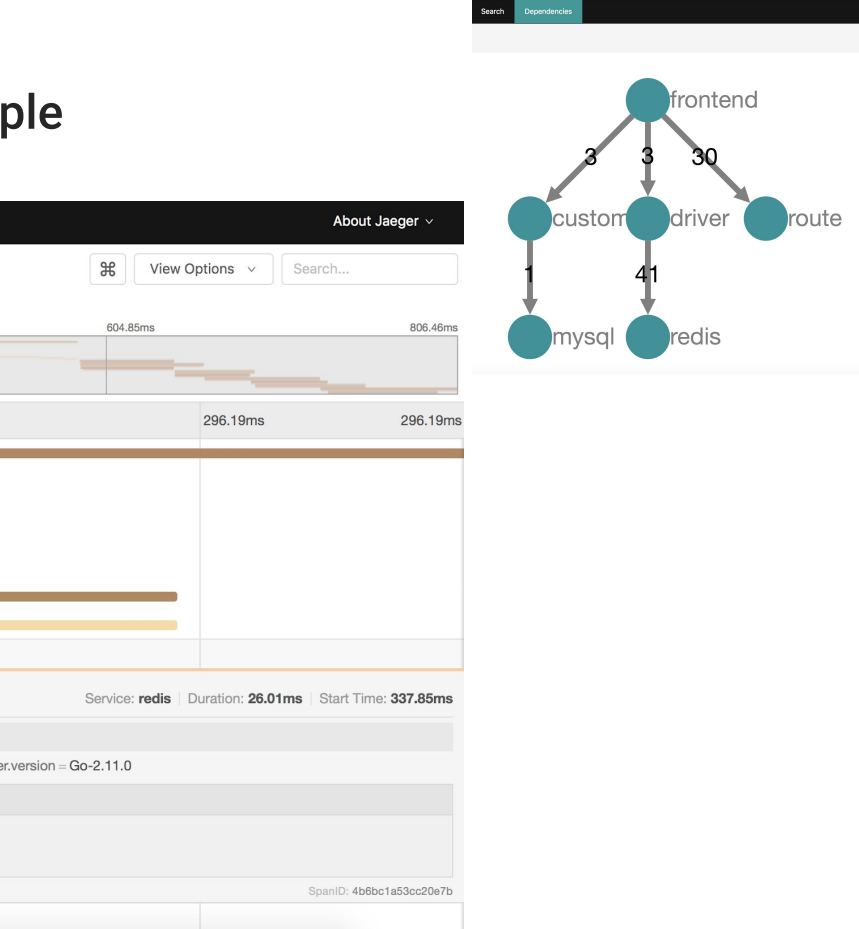
Collector

The collector receives traces from Jaeger agents and runs them through a processing pipeline. Currently Jaeger pipeline validates traces, indexes them, performs any transformations, and finally stores them. Jaeger storage is a pluggable component which currently supports Cassandra and ElasticSearch.

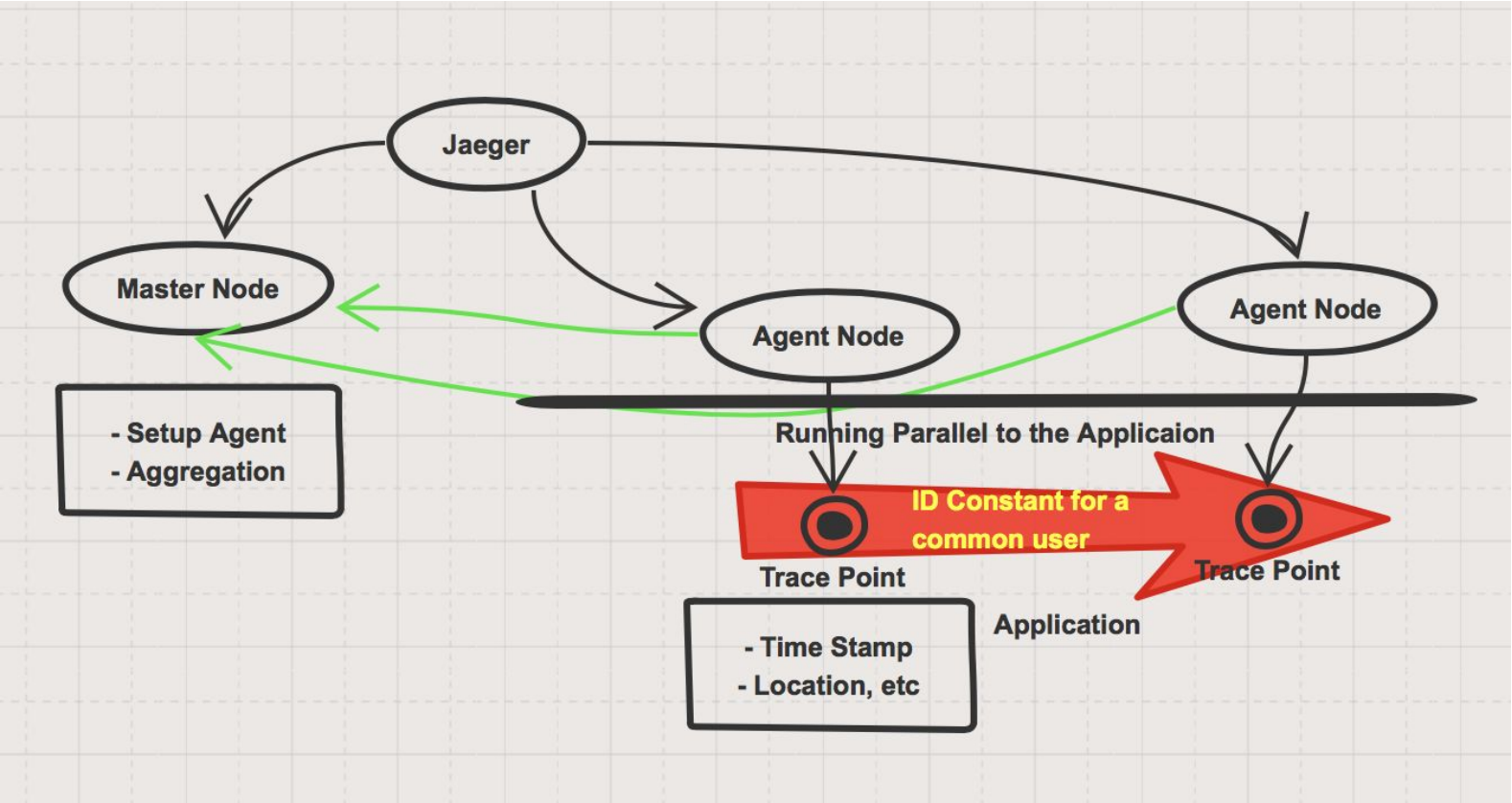
Query

Query is a service that retrieves traces from storage and hosts a UI to display them.

Jaeger Local Deployment and HotROD Example



Jaeger Tracing Applicaition



Planning for Sprint 3

1. Adding Radosgateway
2. Adding client to use Ceph via Radosgateway
3. Learning the Blkin architecture
4. Compile Ceph with Blkin

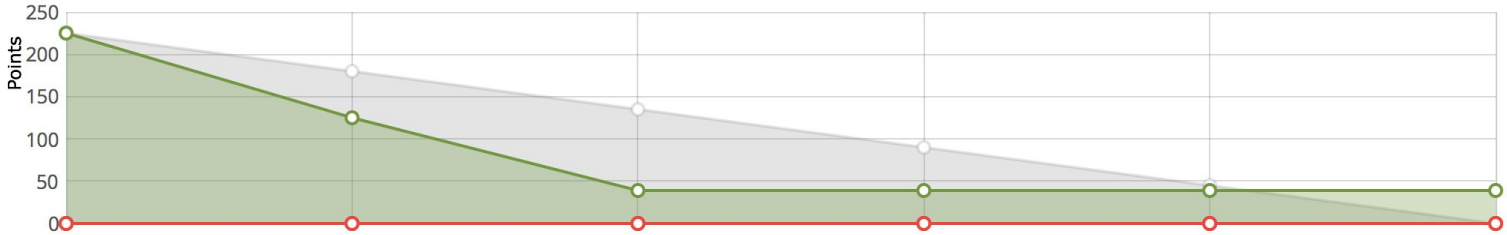
User Stories

Sprint 2(Current)

- As a product owner I should set up the environment on our machines collectively in a team so that we have a collective and better understanding of the tools
- As a product owner, I should learn to deploy Ceph on the environment, so that I can further collect traces using ceph and go on to understanding the existing tracing infrastructure
- As a product owner, I should compile a project presentation, so that the audience and stakeholders have a good understanding of the progress and workflow of our team

Sprint 3(Next)

- As a product owner I should set the Radosgw client so that we can demonstrate the metrics of the Ceph deployment and its interaction with the clients
- As a product owner, I should learn the Blkin architecture and how it is going about the tracing in Ceph currently and its procedure for implementing the three parts of tracing
- As a product owner, I should compile a project presentation, so that the audience and stakeholders have a good understanding of the progress and workflow of our team
- As a product owner, I would like to compile Ceph with Blkin so that I understand the existing limitation



References

[1] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, “Dapper, a large-scale distributed sys-tems tracing infrastructure,” Google, Inc., Tech. Rep., 2010. [Online]. Available: https://research.google.com/archive/papers/dapper-2010-1.pdf

[2] Red Hat, Inc. (2017) Ceph homepage. [Online]. Available: https://ceph.com

[3] ——. (2016) Intro to Ceph Ceph Documentation. [Online]. Available: http://docs.ceph.com/docs/master/start/intro/

[4] Sage A. Weil, et al., “Ceph: a scalable, high-performance distributed file system,” OSDI 06 Proceedings of the 7th symposium on Operating systems design and implementation, 2006.

[5] Raja R. Sambasivan, et al., “So, you want to trace your distributed system? Key design insights from years of practical experience,” Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-14-102, April, 2014.

[6] Red Hat, Inc. (2016) Tracing Ceph With BlkKin Ceph Documentation. [Online]. Available: http://docs.ceph.com/docs/master/dev/blkin/

Thank You!