End to End tracing in Ceph

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

Scope

A feasibility study of Blkin’s replaceability by Jaeger. Replace components of Ceph using Blkin with Jaeger to see the behaviour of the components.

Accomplishments

- Deployment of Ceph on Virtual Cluster
- Study of the complete Blkin API
- Study of the Jaeger and its components
- Instrument Jaeger on a simple client server application
- Mapping of the components of Blkin and Jaeger
- Study and documentation of the complexities involved in replacement

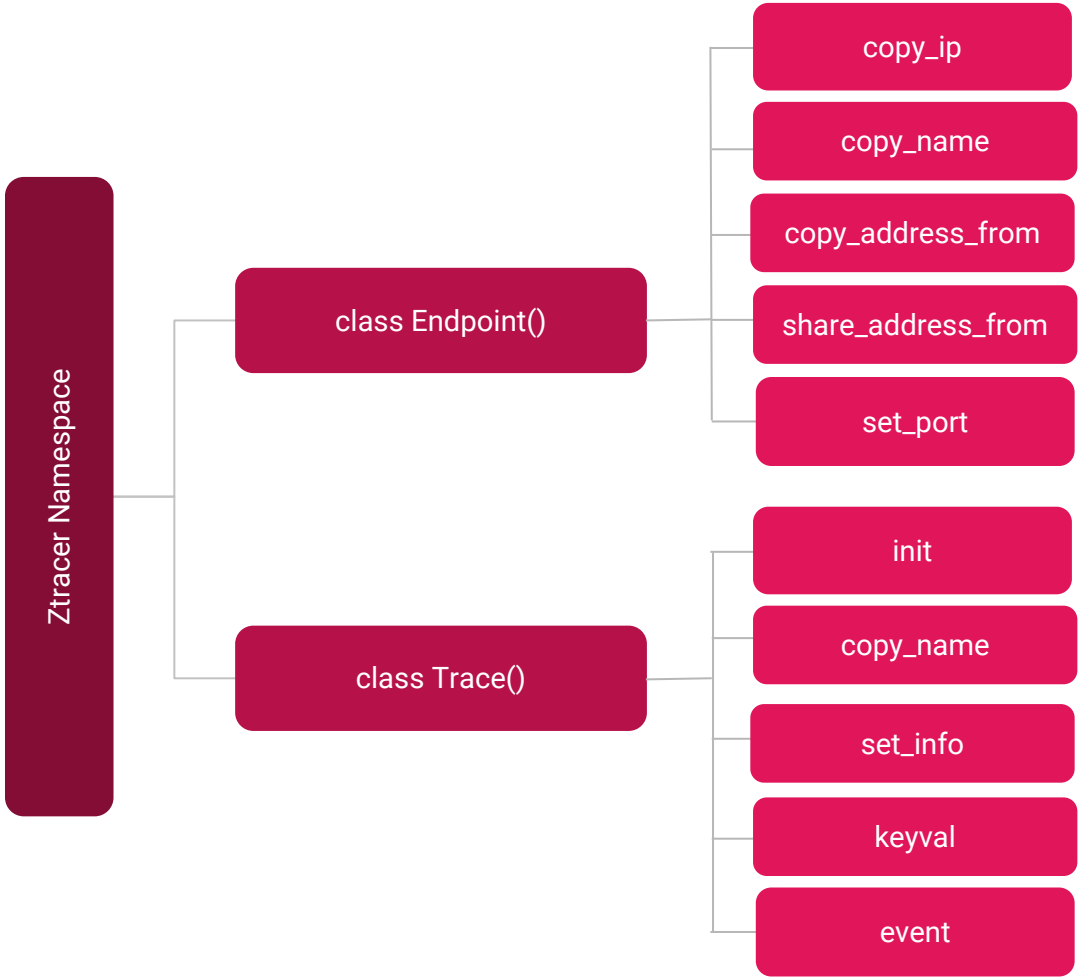
Strategy Employed

1. Mapping of the Blkin API with their correspondents in jaeger.

2. By keeping the name of the functions same, altering the body with jaeger's correspondents.

3. Ztracer can wrap around Jaeger functions without modifying tracepoints in Ceph code, at instances where the tracing functions are called in Ceph

BIkin API


Ztracer Namespace
class Endpoint()
copy_ip
copy_name
copy_address_from
share_address_from
set_port
class Trace()
init
copy_name
set_info
keyval
event

Jaeger Functions

```
32  class LogRecord {
33    public:
34      using Clock = std::chrono::system_clock;
35
36      LogRecord()
37          : _timestamp(Clock::now())
38      {
39      }
40
41      template <typename FieldIterator>
42      LogRecord(const Clock::time_point& timestamp,
43                FieldIterator first,
44                FieldIterator last)
45          : _timestamp(timestamp)
46          , _fields(first, last)
47      {
48      }
49
50      LogRecord(const opentracing::LogRecord & other)
51          : _timestamp(other.timestamp),
52            _fields(other.fields.begin(), other.fields.end())
53      {
54      }
55
56      const Clock::time_point& timestamp() const { return _timestamp; }
57
58      const std::vector<Tag>& fields() const { return _fields; }
59
```

LogRecord.h

- Purpose of Log is to hold trace and span id.
- Different constructors are defined and they serve different purposes..
- Eg. Providing current timestamp
- Yet to Map this to Blkin.

Jaeger Functions

```
Span(const Span& span)
{
    std::lock(_mutex, span._mutex);
    std::lock_guard<std::mutex> lock(_mutex, std::adopt_lock);
    std::lock_guard<std::mutex> spanLock(span._mutex, std::adopt_lock);

    _tracer = span._tracer;
    _context = span._context;
    _operationName = span._operationName;
    _startTimeSystem = span._startTimeSystem;
    _startTimeSteady = span._startTimeSteady;
    _duration = span._duration;
    _tags = span._tags;
    _logs = span._logs;
    _references = span._references;
}

// Pass-by-value intentional to implement copy-and-swap.
Span& operator=(Span rhs)
{
    swap(rhs);
    return *this;
}

~Span() { Finish(); }
```

Span.h

- This function gives us information about the Span.
- Eg. Span duration, Start times, Span Tags, Span duration etc

.

Jaeger Functions

```
void swap(SpanContext& ctx)
{
    using std::swap;
    swap(_traceID, ctx._traceID);
    swap(_spanID, ctx._spanID);
    swap(_parentID, ctx._parentID);
    swap(_flags, ctx._flags);
    swap(_baggage, ctx._baggage);
    swap(_debugID, ctx._debugID);
}

friend void swap(SpanContext& lhs, SpanContext& rhs) { lhs.swap(rhs); }

const TraceID& traceID() const { return _traceID; }

uint64_t spanID() const { return _spanID; }

uint64_t parentID() const { return _parentID; }

const StrMap& baggage() const { return _baggage; }

SpanContext withBaggage(const StrMap& baggage) const
{
    return SpanContext(
        _traceID, _spanID, _parentID, _flags, baggage, _debugID);
}
```

SpanContext.h

- Key value annotations are used to define applications specific information in traces.
- Object ctx of class Span Context is defined.
- Eg. Swap function is used which determines whether trace_id variable needs to be used or ctx.trace_id needs to be used.

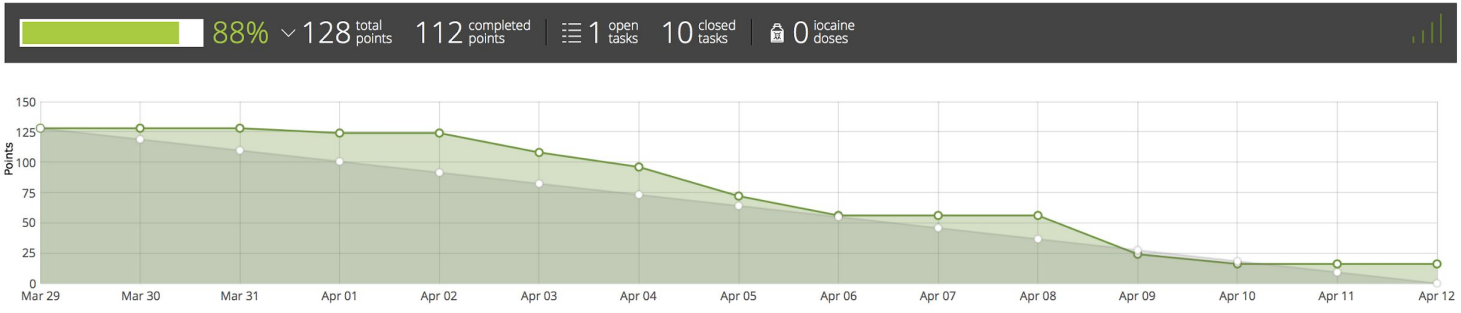
.

Mapping

Further Steps

1. Replacement of Ztracer namespace in Blkin
2. Testing and compiling of Ceph on Virtual cluster

Burndown



Sprint 5 ending with demo(5)

112 closed

29 Mar 2018-12 Apr 2018

128 total

#50 As a product developer, I would like to set-up a simple client-server TCP connection and instrument Jaeger tracepoints on it	24
#64 As a product developer, I need to find if there are more objects of Trace and Endpoint class which are being initialized throughout the code, are they initialized once or in more than one locations in the code	8
#60 As a product developer, I would like to find the functions in Blkin relevant to tracing so that I can draw equivalences with Jaeger functions	16
#59 As a product developer, I would like to list the functions present in the jaeger API and list the roles of each so that I can compare the with the Blkin functions to start replacement	16
#57 As a product developer, I would like to draw equivalences between Blkin and Jaeger tracing from reading from the codes so that I can start the replacement	16
#49 As a product developer, I would like to read through the Blkin code and debug it so that I understand the thread local storage of traces	8
#55 As a product developer, I would like to understand the Blkin code in depth so that replacement with Jaeger becomes fairly simple	24
#66 As a product developer, I would like to replace Blkin functions by jaeger functions appropriately so that I can start replacing the tracing infrastructure	16

Thank You!