



# A-OPS在分布式存储Ceph中应用研究

张道龙

软通动力openEuler研究中心

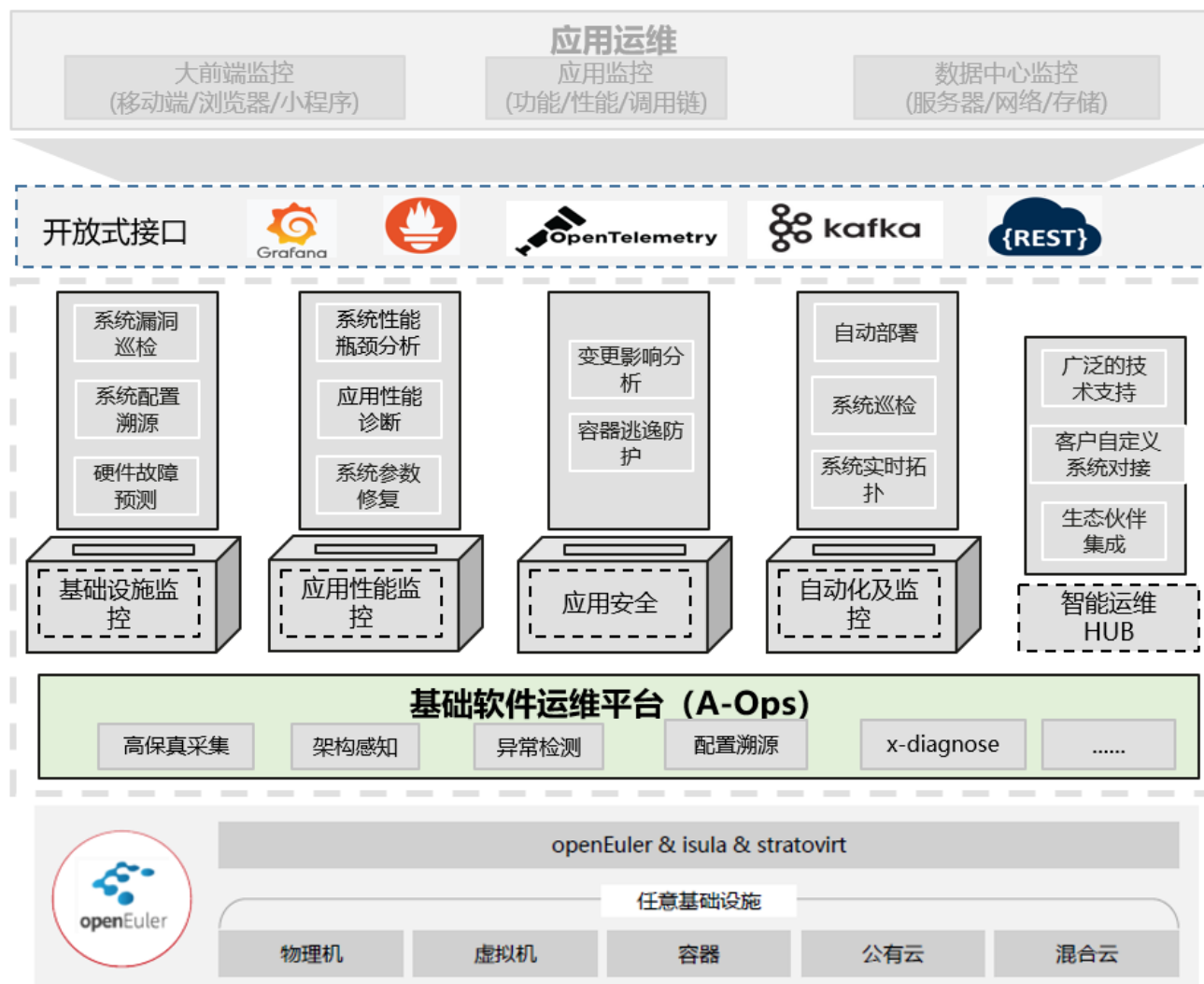
# A-Ops简介

A-Ops是一款智能运维工具，通过实现智能运维基本框架，提供配置溯源，架构感知，故障定位基础能力，支持快速排障和运维成本降低，主要功能:

- 资产管理
- CVE管理
- 异常检测
- 配置溯源

关键技术:

- eBPF
- AI分析

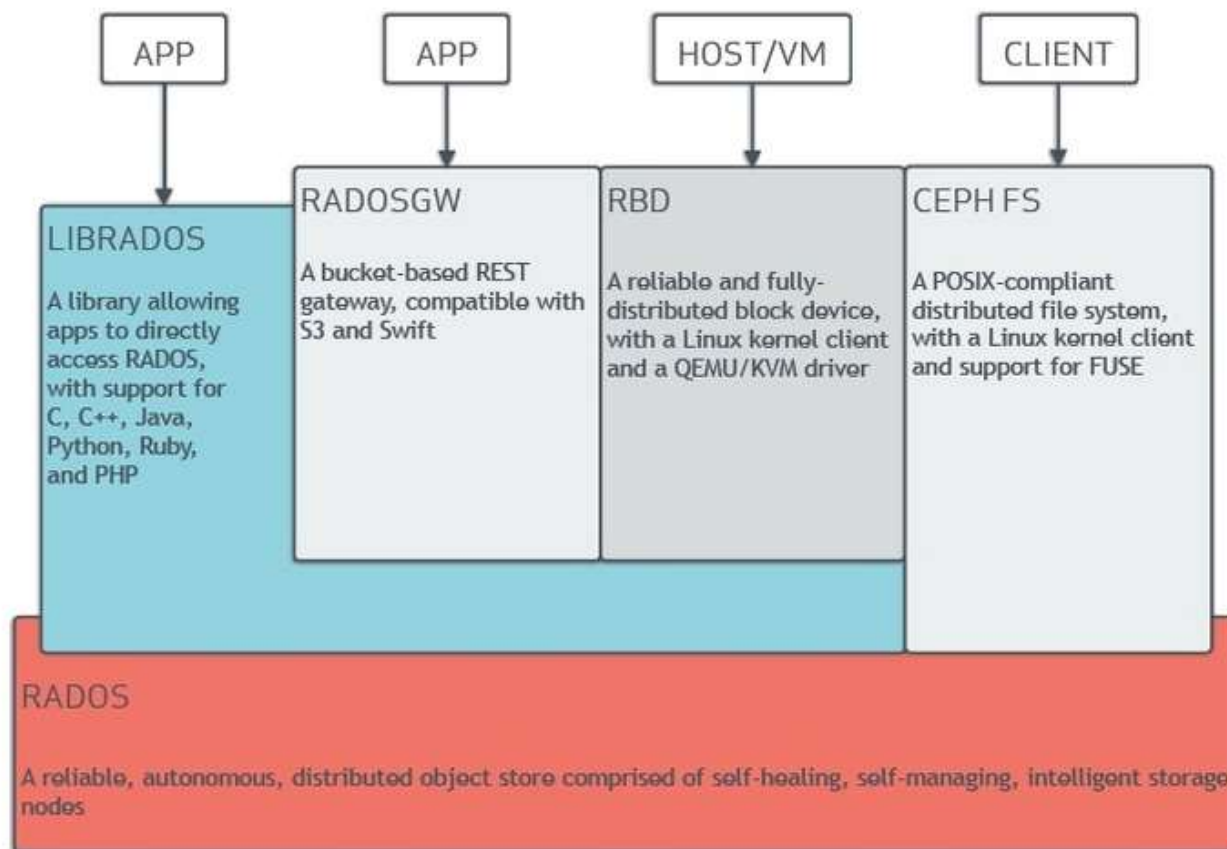


# 分布式存储Ceph简介

Ceph 是一个统一的分布式存储软件，并提供分布式、横向扩展，高度可靠性的存储系统。对外提供了：

- 对象存储
- 块设备存储
- 文件存储

主要开发语言C++， Python。



# Ceph监控和问题

Ceph是分布式存储系统，影响ceph运行资源比较多：

- ceph进程(ceph-osd,ceph-mon.....)
- 磁盘
- 网络
- CPU
- 内存

分布式存储Ceph的组件多，监控起来相对比较复杂，目前存在问题：

- 发现问题和定位问题之间衔接
- 随机性故障诊断能力不足
- 监控数据和问题衔接
- 缺乏系统性的监控分析

ceph中常见block io告警

```
1 activeunder-sized+degraded+remapped+backfilling
[root@ceph1 ~]# ceph -s
cluster:
  id:         70be1e2f-70da-4778-a226-6ad7e7a264ae
  health:     HEALTH_WARN
             Degraded data redundancy: 2188389/4412944 objects degraded (49.590%), 1 pg degraded, 1 pg undersized
             20 slow ops, oldest one blocked for 123 sec, osd.6 has slow ops

services:
  mon: 1 daemons, quorum ceph1 (age 2d)
  mgr: ceph1(active, since 2d)
  osd: 3 osds: 2 up (since 2h), 2 in (since 2h); 1 remapped pgs

data:
  pools:   1 pools, 1 pgs
  objects: 2.21M objects, 5.6 GiB
  usage:   13 GiB used, 6.5 TiB / 6.5 TiB avail
  pgs:     2188389/4412944 objects degraded (49.590%)
           1 activeunder-sized+degraded+remapped+backfilling
```



# A-Ops实现对Ceph端到端的监控

从客户端的卷，文件，对象，到服务端的磁盘，A-Ops可以实现端到端的监控。

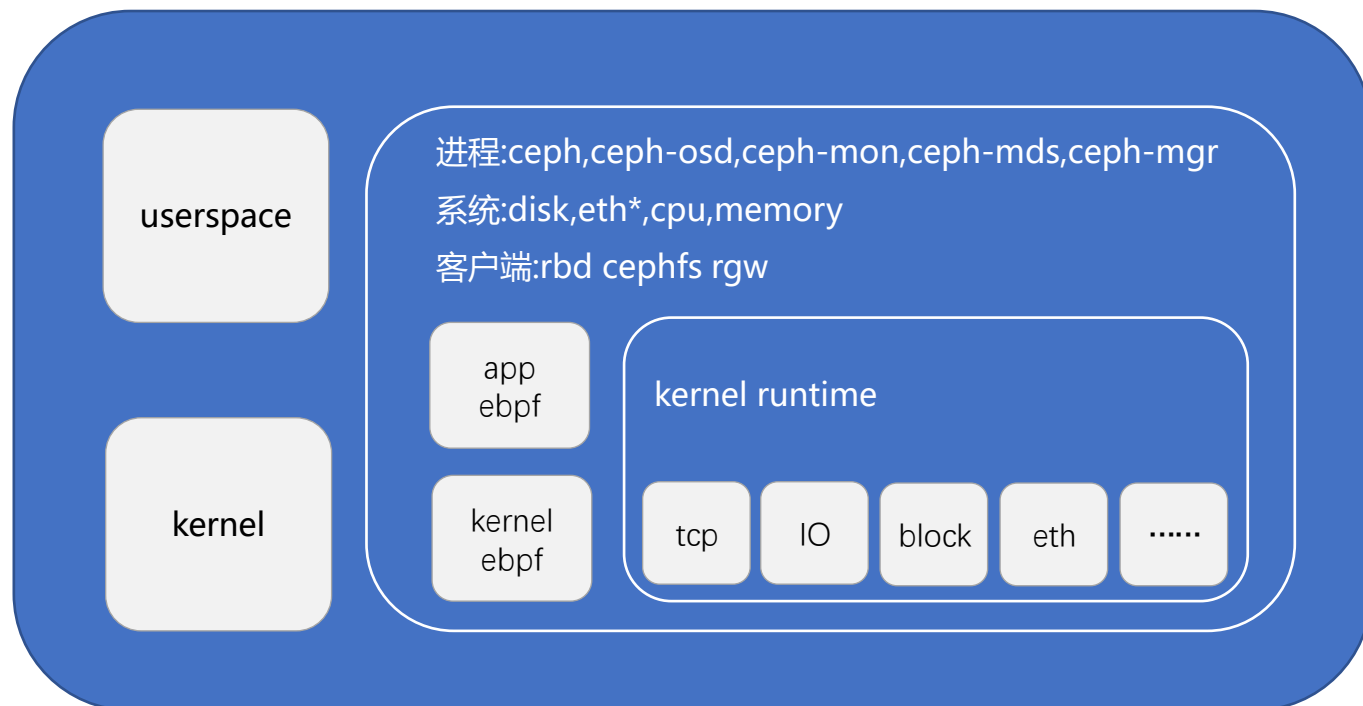
整体监控可以分为用户空间按和内核空间层面：

- 用户空间监控

aops-ceres可以收集节点基本信息。

- 内核空间监控

gala-gopher定位OS系统后台服务，提供基础软件全方位的观测能力，基于eBPF技术，持续性、低底噪的方式为采集基础软件运行时数据。



# A-Ops监控Ceph问题分析案例—ceph-osd通信故障

通过模拟Ceph 中通信故障，来具体分析A-ops监控能力，故障是通过模拟进程ceph-osd通信模块故障,延时主要是通过sleep方式实现。

```
[detached from 2237922.test]
[root@ceph1 ~]#
[root@ceph1 ~]#
[root@ceph1 ~]# ceph daemon osd.0 config set osd_debug_inject_dispatch_delay_duration 15
{
  "success": "osd_debug_inject_dispatch_delay_duration = '15.000000' (not observed, change may require restart) "
}
[root@ceph1 ~]#
[root@ceph1 ~]#
[root@ceph1 ~]# ceph daemon osd.0 config set osd_debug_inject_dispatch_delay_probability 0.5
{
  "success": "osd_debug_inject_dispatch_delay_probability = '0.500000' (not observed, change may require restart) "
}
[root@ceph1 ~]#
[root@ceph1 ~]#
```

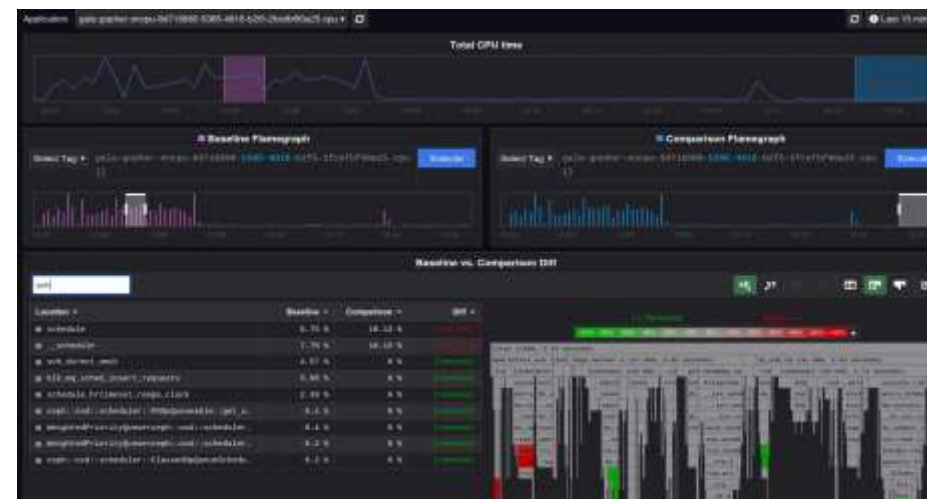
```
void maybe_inject_dispatch_delay() {
    if (g_conf()->osd_debug_inject_dispatch_delay_probability > 0) {
        if (rand() % 10000 <
            g_conf()->osd_debug_inject_dispatch_delay_probability * 10000) {
            utime_t t;
            t.set_from_double(g_conf()->osd_debug_inject_dispatch_delay_duration);
            t.sleep();
        }
    }
}
```

注入延时后，fio测试的IO延时会逐渐增加。

```
fio -filename=/dev/rbd0 -iodepth 64 -direct=1 -thread -rw=write -bs=512K -runtime=60 -name=mytest -ioengine=posixaio -numjobs=1
```

ceph集群会出现以下错误：

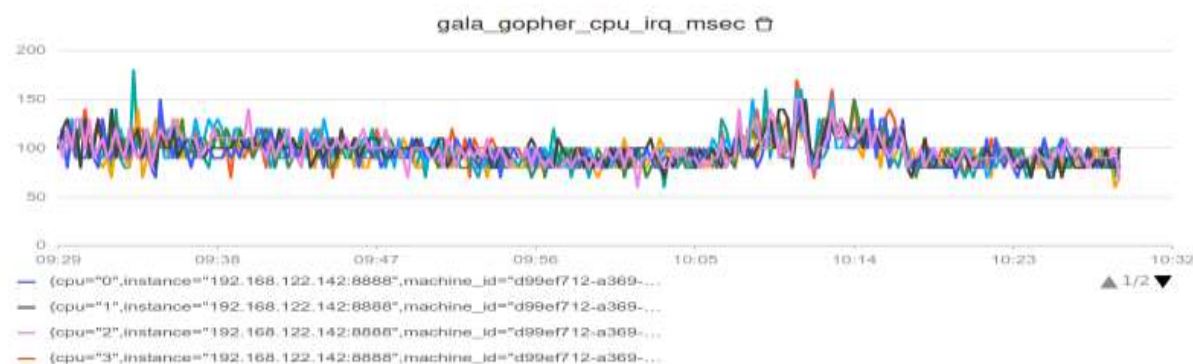
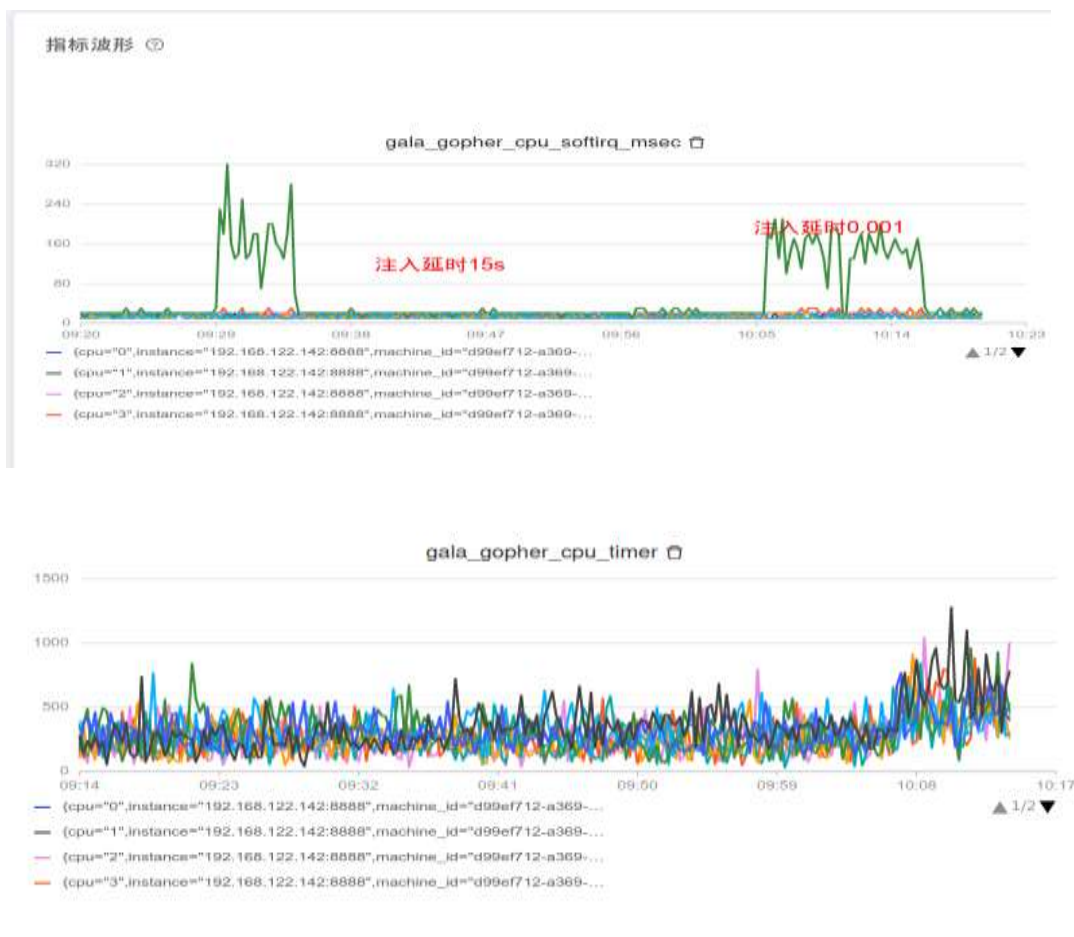
```
[root@ceph1 ~]#
[root@ceph1 ~]#
[root@ceph1 ~]# ceph health detail
HEALTH_WARN 6 slow ops, oldest one blocked for 982 sec, daemons [osd.0,osd.2] have slow ops.
[WRN] SLOW_OPS: 6 slow ops, oldest one blocked for 982 sec, daemons [osd.0,osd.2] have slow ops.
[root@ceph1 ~]# █
```

[illegible]



# A-Ops监控Ceph问题分析案例—ceph-osd通信故障

ceph存在负载的情况，softirq和cputimer占用时间在一个相对稳定区间，延时注入后，ceph-osd处理速度降低，总体cpu负载也随着降低。CPU的监控指标和程序业务类型有关系，具体表现看程序业务类型。





# A-Ops监控Ceph问题分析案例—ceph-osd读故障

通过模拟ceph-osd数据校验错误，来实现模拟ceph-osd读故障。

```
[root@ceph1 ~]# ceph daemon osd.0 config set bluestore_retry_disk_reads 10
{
  "success": "bluestore_retry_disk_reads = '10' (not observed, change may require restart) "
}
[root@ceph1 ~]# ceph daemon osd.0 config set bluestore_debug_inject_csum_err_probability 1
{
  "success": "bluestore_debug_inject_csum_err_probability = '1.000000' (not observed, change may require restart) "
}
```

注入延时后，客户端跑fio。

```
# fio -filename=/dev/rbd0 -iodepth 16 -direct=1 -thread -rw=read -bs=4K -runtime=600 -name=mytest -ioengine=posixaio -numjobs=1
```

ceph集群会出现以下错误：

```
root@ceph1 ~]# ceph -s
cluster:
  id:      769e1e2f-7da0-4770-a226-6ad7e7a264ae
  health: HEALTH_WARN
           Too many repaired reads on 1 OSDs
           Degraded data redundancy: 1/1536 objects degraded (0.065%), 1 pg degraded
```

# A-Ops监控Ceph问题分析案例—ceph-osd读故障

模拟读故障实现方式是模拟读取数据crc检验失败，然后retry read。主要代码分析：

```
int BlueStore::verify_csum(OnodeRef& o,
    const bluestore_blob_t* blob, uint64_t blob_offset)
{
    const bufferlist& bl,
    uint64_t logical_offset) const
{
    int bad;
    uint64_t bad_csum;
    auto start = mono_clock::now();
    int r = blob->verify_csum(blob_offset, bl, &bad, &bad_csum);
    if (cct->conf->bluestore_debug_inject_csum_err_probability >= 65
        (rand() % 10000) < cct->conf->bluestore_debug_inject_csum_err_probability * 10000) {
        derr << "func: << "injecting bluestore checksum verification error" << endl;
        bad = blob_offset;
        r = -1;
        bad_csum = 0;
    }
    if (r < 0) {
        if (r == -1) {
            PExtentVector pex;
            blob->map(
                bad,
                blob->get_csum_chunk_size(),
                &!(uint64_t offset, uint64_t length) {
                    pex.emplace_back(bluestore_pextent_t(offset, length));
                    return 0;
                });
            derr << "func: << "bad"
                << "Checksummer::get_csum_type string(blob->csum_type)
                << "0x" << std::hex << blob->get_csum_chunk_size()
                << "checksum at blob offset 0x" << bad
                << " got 0x" << bad_csum << ", expected 0x"
                << blob->get_csum_item(bad / blob->get_csum_chunk_size()) << std::dec
                << endl;
        } else {
            derr << "func: << "failed with non-zero" << cpp_strerror(r) << endl;
        }
    }
    log_latency(
        bluestore_csum_lat,
        mono_clock::now() - start,
        cct->conf->bluestore_log_op_age);
    if (cct->conf->bluestore_ignore_data_csum) {
        return 0;
    }
    return r;
}

return 0;

int BlueStore::generate_read_result_bli(
    OnodeRef o,
    uint64_t offset,
    size_t length,
    ready_regions_t& ready_regions,
    vector<bufferlist*>& compressed_blob_bls,
    blobs2read_t& blobs2read,
    bool buffered,
    bool* csum_error,
    bufferlist& bl)
{
    // uncompress and decompress desired blobs
    auto p = compressed_blob_bls.begin();
    blobs2read_t::iterator b2r_it = blobs2read.begin();
    while (b2r_it != blobs2read.end()) {
        const BlobRef& bptr = b2r_it->first;
        regions2read_t& r2r = b2r_it->second;
        dout(20) << "func: << "blob" << "bptr" << std::hex
            << " read 0x" << r2r << std::dec << endl;
        if (bptr->get_blob().is_compressed()) {
            ceph_assert(p != compressed_blob_bls.end());
            bufferlist& compressed_bl = *p++;
            if (!verify_csum(o, &bptr->get_blob(), 0, compressed_bl,
                r2r.front().regs.front().logical_offset) < 0) {
                *csum_error = true;
                return -EIO;
            }
            bufferlist raw_bl;
            auto r = decompress(compressed_bl, &raw_bl);
            if (r < 0)
                return r;
            if (buffered) {
                bptr->shared_blob->bc.did_read(bptr->shared_blob->get_cache(), 0,
                    raw_bl);
            }
            for (auto& req : r2r) {
                for (auto& r : req.regs) {
                    ready_regions[r.logical_offset].substr_off(
                        raw_bl, r.blob_offset, r.length);
                }
            }
        } else {
            for (auto& req : r2r) {
                if (!verify_csum(o, &bptr->get_blob(), req.r.off, req.bl,
                    req.regs.front().logical_offset) < 0) {
                    *csum_error = true;
                    return -EIO;
                }
                if (buffered) {
                    bptr->shared_blob->bc.did_read(bptr->shared_blob->get_cache(),
                        req.r.off, req.bl);
                }
            }
        }
    }
}

// read raw blob data.
start = mono_clock::now(); // for the sake of simplicity
// measure the whole block below.
// The error isn't that much...
vector<bufferlist*> compressed_blob_bls;
IOContext ioc(cct, &bl, true); // allow EIO
r = _prepare_read_ioc(blobs2read, &compressed_blob_bls, &ioc);
// we always issue aio for reading, an errors other than EIO are not allowed
if (r < 0)
    return r;

int64_t num_ios = blobs2read.size();
if (ioc.has_pending_aio()) {
    num_ios = ioc.get_num_ios();
    bdev->aio_submit(&ioc);
    dout(10) << "func: << "waiting for aio" << endl;
    ioc.aio_wait();
    r = ioc.get_return_value();
    if (r < 0) {
        ceph_assert(r == -EIO); // no other errors allowed
        return -EIO;
    }
}
log_latency_fnl(
    bluestore_read_wait_aio_lat,
    mono_clock::now() - start,
    cct->conf->bluestore_log_op_age,
    &!(auto lat) { return " num_ios = " + stringify(num_ios); });

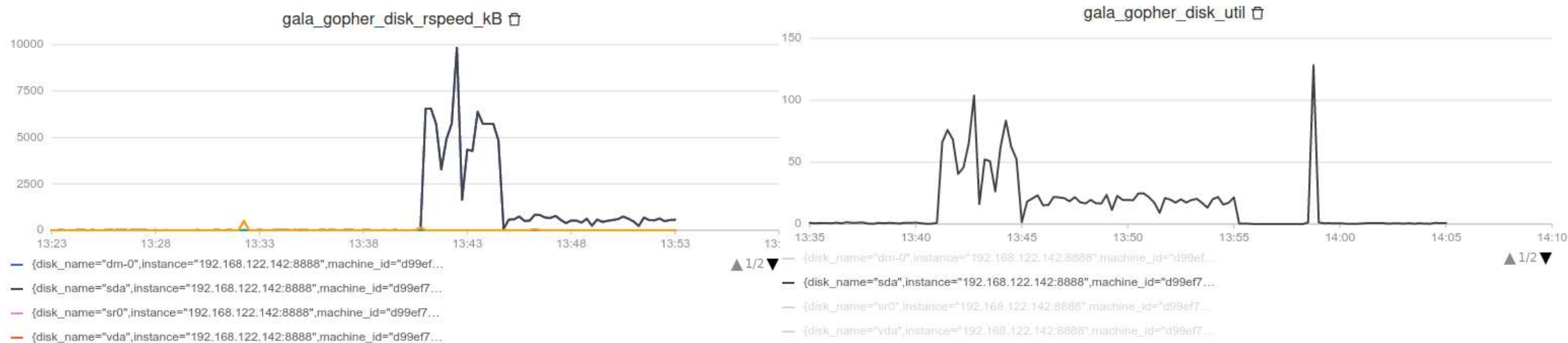
bool csum_error = false;
r = generate_read_result_bli(o, offset, length, ready_regions,
    compressed_blob_bls, blobs2read,
    buffered, &csum_error, bl);

if (csum_error) {
    // Handle spurious read errors caused by a kernel bug.
    // We sometimes get all-zero pages as a result of the read under
    // high memory pressure. Retrying the failing read succeeds in most
    // cases.
    // See also: http://tracker.ceph.com/issues/23444
    if (retry_count >= cct->conf->bluestore_retry_disk_reads) {
        return -EIO;
    }
    return _do_readic(o, offset, length, bl, op_flags, retry_count + 1);
}
r = bl.length();
if (retry_count) {
    logger->inc(bluestore_reads_with_retries);
    dout(1) << "func: << "read at 0x" << std::hex << offset << " << " << length
        << " failed " << std::dec << retry_count << " times before success"
        << endl;
    stringstream s;
}

os/bluestore/BlueStore.cc 10060,19 60% os/bluestore/BlueStore.cc 9880,12 59% os/bluestore/BlueStore.cc 10034,7 60%
```

# A-Ops监控Ceph问题分析案例—ceph-osd读故障

a-ops监控信息，13:45后磁盘的带宽和使用都在下降。



Application: `gala-gopher-oncpu-8d716880-5385-4818-b2f5-2fcef90a25.cpu` 🔄 🕒 Last 15 minutes

**Total CPU time**

**Baseline Flamegraph**

Select Tag: `gala-gopher-oncpu-8d716880-5385-4818-b2f5-2fcef90a25.cpu` Execute

**Comparison Flamegraph**

Select Tag: `gala-gopher-oncpu-8d716880-5385-4818-b2f5-2fcef90a25.cpu` Execute

**Baseline vs. Comparison Diff**

read

Location	Baseline	Comparison	Diff
<code> aio_read</code>	9.48 %	3.01 %	(-66.25%)
<code> __read</code>	9.21 %	15.25 %	(+65.00%)
<code> blkdev_read_iter</code>	9.21 %	3.01 %	(-67.32%)
<code> generic_file_read_iter</code>	9.21 %	3.01 %	(-67.32%)
<code> pthread_cond_broadcast</code>	7.08 %	3.01 %	(-57.49%)
<code> ksys_read</code>	5.74 %	12.99 %	(+125.28%)
<code> vfs_read</code>	5.21 %	12.99 %	(+149.33%)
<code> new_sync_read</code>	4.27 %	12.99 %	(+206.56%)
<code> sock_read_iter</code>	3.87 %	12.81 %	(+230.23%)
<code> OSD::dequeue_op(boost::intrusive_ptr&lt;PG&gt;, _</code>	2.94 %	0 %	(removed)
<code> read events</code>	2.67 %	1.32 %	(-50.56%)

**Stack Diff**

Total (100%, 12.98 seconds)

Before: `log_mgr-worker-2` (23.12s, 4.24 seconds) `sp_read_tp` (42.42s, 5.43 seconds)

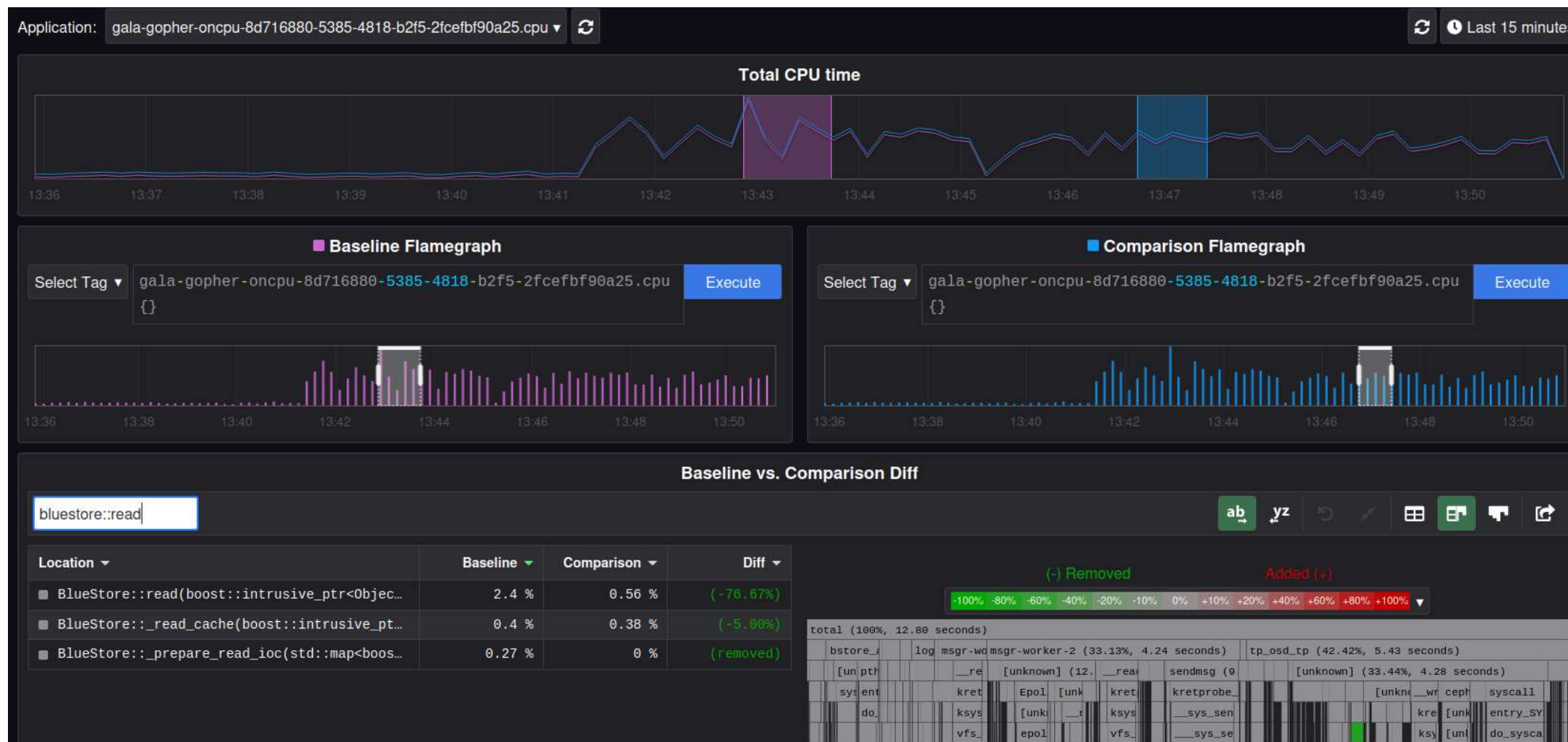
Legend: (-) Removed, (+) Added

Scale: 100%, 80%, 60%, 40%, 20%, 0%, +10%, +20%, +40%, +60%, +80%, +100%



# A-Ops监控Ceph问题分析案例—ceph-osd读故障

read上层调用, bluestore::read在单位时间内占用时间在相对在减少。



# A-Ops监控Ceph问题分析案例—TC注入网络丢包

read上层调用, bluestore::read在单位时间内占用时间在相对在减少。

```
[root@ceph1 ~]# tc qdisc add dev ens3 root netem loss 30% 25%
```

注入丢包前fio性能:

```
[root@worker3 ~]# sleep 60; fio -filename=/dev/rbd0 -iodepth 16 -direct=1 -thread -rw=read -bs=4K -runtime=300 -name=mytest -ioengine=posixaio -numjobs=1
mytest: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=posixaio, iodepth=16
fio-3.29
Starting 1 thread
Jobs: 1 (f=1): [R(1)][100.0%][r=8640KiB/s][r=2160 IOPS][eta 00m:00s]
mytest: (groupid=0, jobs=1): err= 0: pid=21910: Wed Apr 19 16:45:41 2023
read: IOPS=2062, BW=8249KiB/s (8447kB/s)(2417MiB/300005msec)
  slat (nsec): min=110, max=199590, avg=879.42, stdev=737.71
  clat (usec): min=2003, max=83925, avg=7750.24, stdev=5099.56
  lat (usec): min=2003, max=83926, avg=7751.12, stdev=5099.64
  clat percentiles (usec):
    | 1.00th=[ 2311], 5.00th=[ 5145], 10.00th=[ 5407], 20.00th=[ 5800],
    | 30.00th=[ 6063], 40.00th=[ 6390], 50.00th=[ 6652], 60.00th=[ 6980],
    | 70.00th=[ 7430], 80.00th=[ 8029], 90.00th=[10421], 95.00th=[15795],
    | 99.00th=[23987], 99.50th=[48497], 99.90th=[66847], 99.95th=[71828],
    | 99.99th=[76022]
  bw ( KiB/s): min= 3192, max=19552, per=100.00%, avg=8256.22, stdev=2003.11, samples=599
  iops        : min= 798, max= 4888, avg=2063.99, stdev=500.77, samples=599
  lat (msec)  : 4=2.83%, 10=86.53%, 20=9.11%, 50=1.07%, 100=0.46%
```

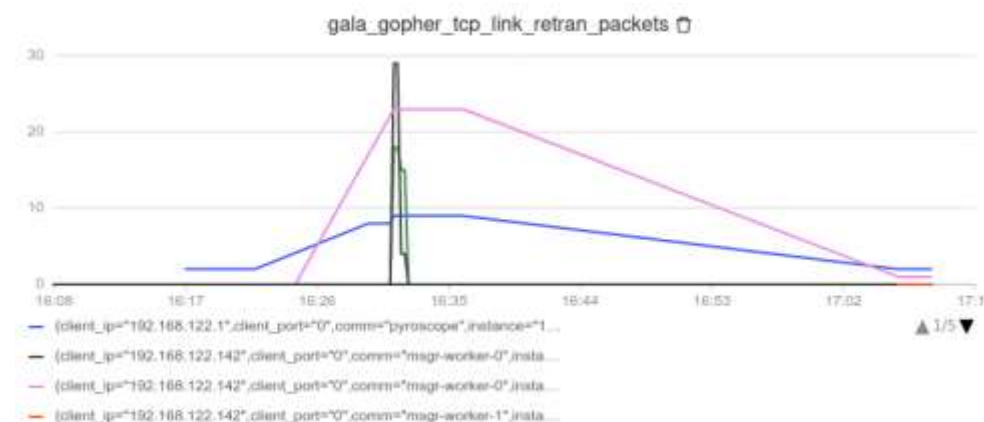
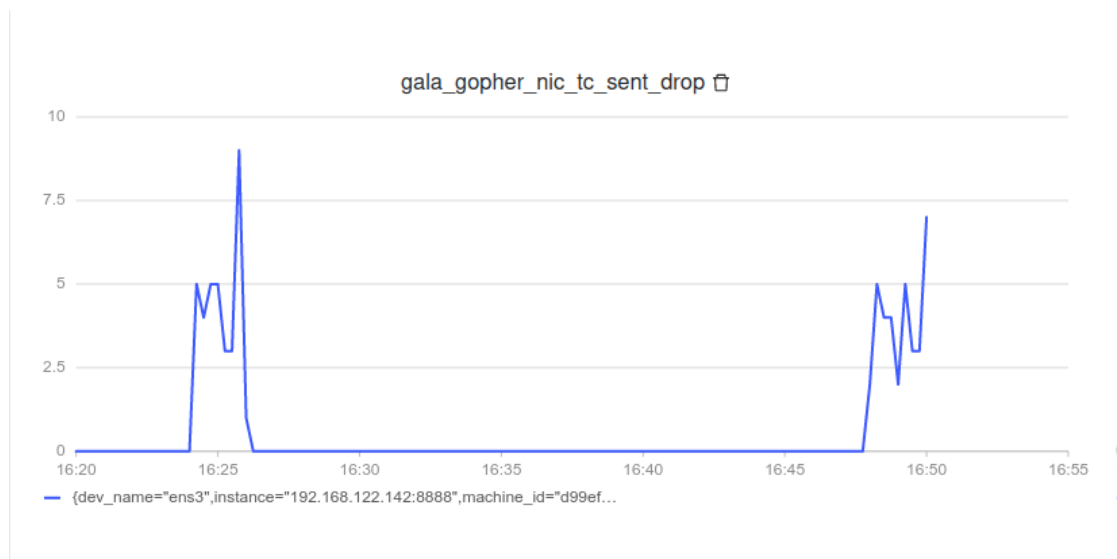
注入丢包后fio性能:

```
[root@worker3 ~]# sleep 60; fio -filename=/dev/rbd0 -iodepth 16 -direct=1 -thread -rw=read -bs=4K -runtime=300 -name=mytest -ioengine=posixaio -numjobs=1
mytest: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=posixaio, iodepth=16
fio-3.29
Starting 1 thread
Jobs: 1 (f=1): [R(1)][0.0%][r=8392KiB/s][r=2098 IOPS][eta 04m:42s]
Jobs: 1 (f=1): [R(1)][100.0%][r=8720KiB/s][r=2180 IOPS][eta 00m:00s]
mytest: (groupid=0, jobs=1): err= 0: pid=22071: Wed Apr 19 16:53:09 2023
read: IOPS=2044, BW=8177KiB/s (8373kB/s)(2396MiB/300007msec)
  slat (nsec): min=132, max=263364, avg=857.48, stdev=763.07
  clat (usec): min=1709, max=265346, avg=7818.73, stdev=6952.27
  lat (usec): min=1710, max=265347, avg=7819.59, stdev=6952.33
  clat percentiles (msec):
    | 1.00th=[ 3], 5.00th=[ 6], 10.00th=[ 6], 20.00th=[ 6],
    | 30.00th=[ 6], 40.00th=[ 7], 50.00th=[ 7], 60.00th=[ 8],
    | 70.00th=[ 8], 80.00th=[ 9], 90.00th=[ 11], 95.00th=[ 16],
    | 99.00th=[ 25], 99.50th=[ 53], 99.90th=[ 72], 99.95th=[ 88],
    | 99.99th=[ 220]
  bw ( KiB/s): min= 3224, max=20464, per=100.00%, avg=8182.13, stdev=2081.00, samples=599
  iops        : min= 806, max= 5116, avg=2045.47, stdev=520.26, samples=599
  lat (msec)  : 2=0.07%, 4=2.70%, 10=86.70%, 20=9.06%, 50=0.93%
  lat (msec)  : 100=0.49%, 250=0.05%, 500=0.01%
```



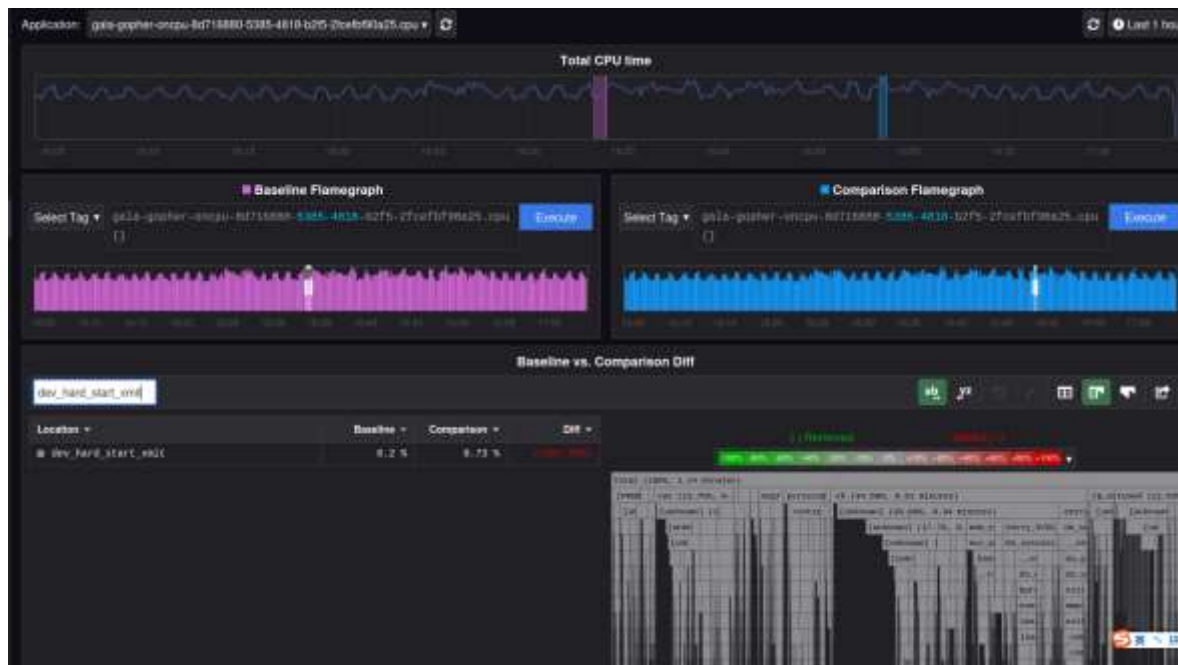
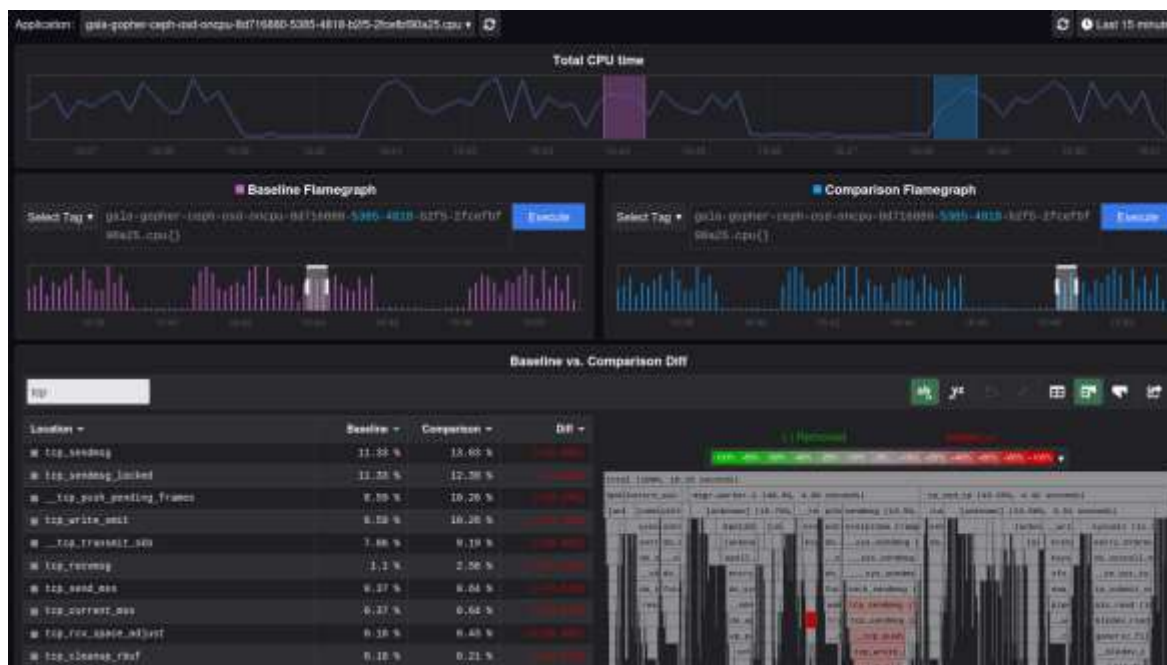
# A-Ops监控Ceph问题分析案例—TC注入网络丢包

read上层调用, bluestore::read在单位时间内占用时间在相对在减少。



# A-Ops监控Ceph问题分析案例—TC注入网络丢包

ceph-osd和系统的火焰图，网络发送包处理函数tcp\_sendmsg，dev\_hard\_start\_xmit占用时间在增加。



# A-Ops拓扑功能在Ceph中应用

## A-Ops拓扑图构建：

根据gala-gopher 采集的所有观测对象实例的数据，并计算它们之间的拓扑关系，将生成的拓扑图保存到图数据库 arangodb 中。拓扑类型可分为：

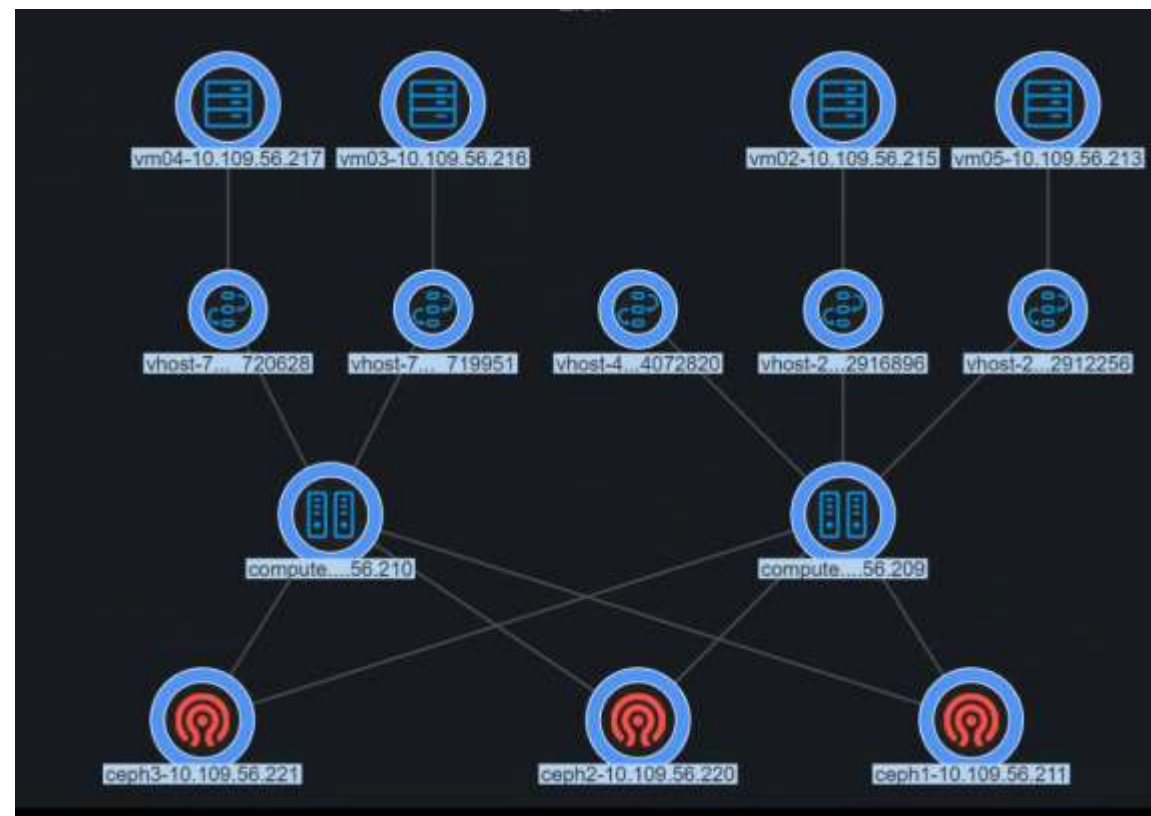
- host 级别拓扑
- 应用级别拓扑
- 从属拓扑（比如主机和主机的磁盘）

在Ceph集群中，构建OS或者应用之间关系拓扑，可以对ceph进程问题溯源。

### 问题

Ceph涉及的进程比较多，在一个节点上可能存在上十万个链接，一个集群几十个节点，如果是app级别这里是否存在性能问题。

## host级别的拓扑



# A-Ops的智能分析功能在Ceph应用

A-Ops的智能分析功能：

- 异常检测

针对操作系统，提供异常检测能力，能够及时发现潜在影响客户

端时延的系统级异常，快速跟踪并解决问题。

- 异常上报

当发现异常行为，平台能够实时上报至Kafka，运维人员只需订阅Kafka消息队列，了解当前系统是否潜在风险。

- AI分析

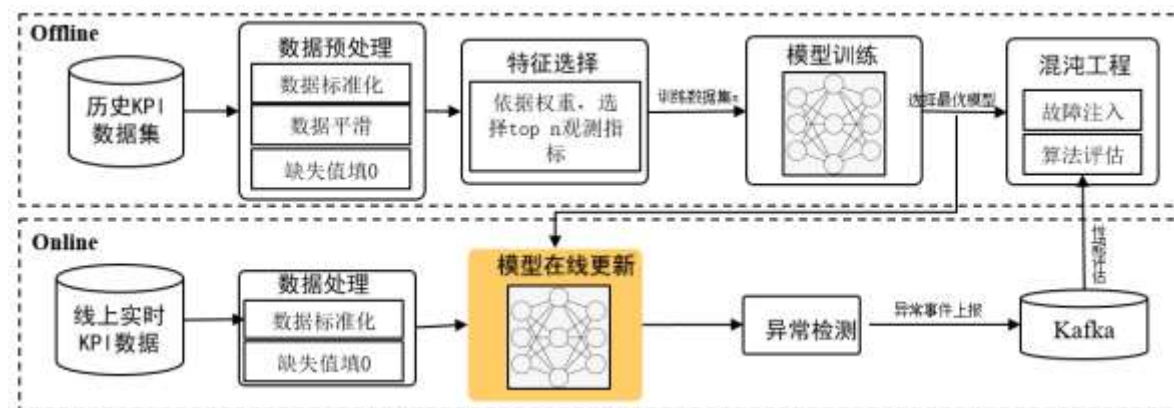
基于线下预训练、线上模型的增量学习与模型更新，能够很好地适应于多维多模态数据故障诊断。

利用A-Ops的AI对Ceph以下资源监控预测分析：

- 磁盘坏盘的预测
- 分析ceph-osd性能
- 分析集群性能瓶颈

## 问题

去噪问题，比如客户端争抢问题。



# 我们的目标

我们正在做的：

- 部署工具开发
- 主机sos信息收集
- 整合pyroscope提供火焰图
- 等等...

A-Ops改进：

- 可视化加强,比如页面提供火焰图
- 具体应用场景功能强化
- 磁盘生命周期状态监控
- 等等...

