京东金融

基于Ceph构建PB级对象存储实践

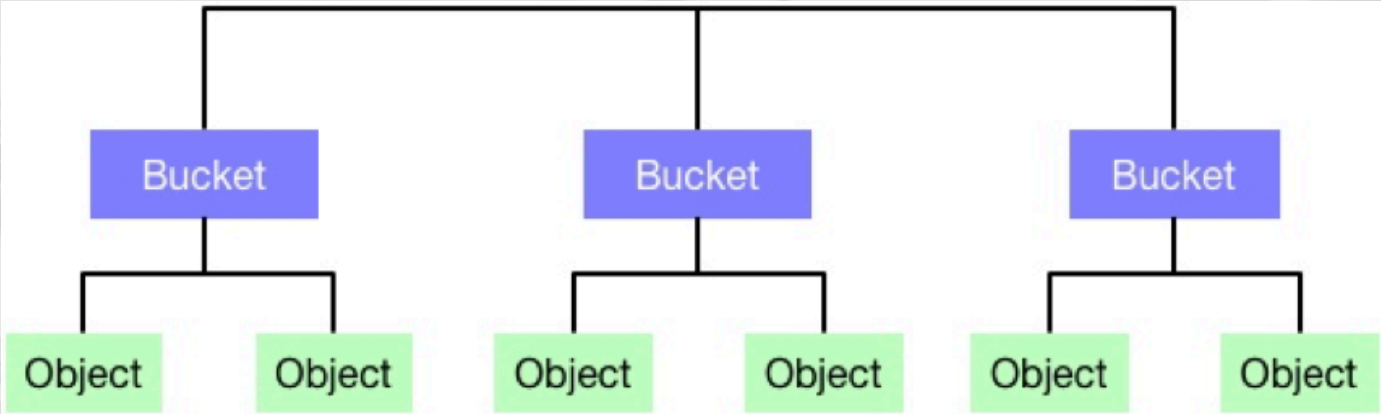
JD Finance

01 对象存储简介

对象存储迎来爆炸式发展

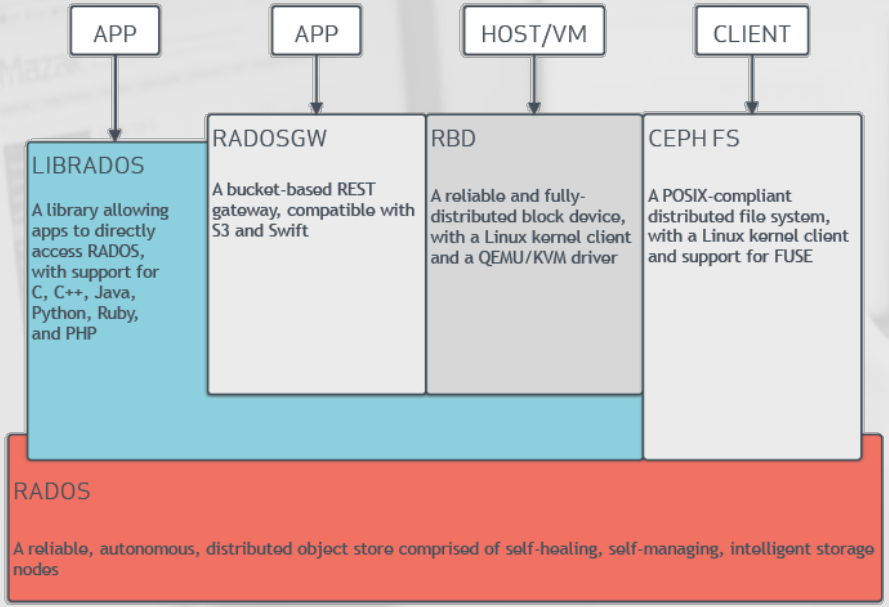
- 非结构化数据的爆炸性增长
  - 人工智能行业初露锋芒:海量训练数据存储.....
  - 互联网行业遍地开花:短视频、直播、点播、云盘、云备份.....

扁平化管理模型



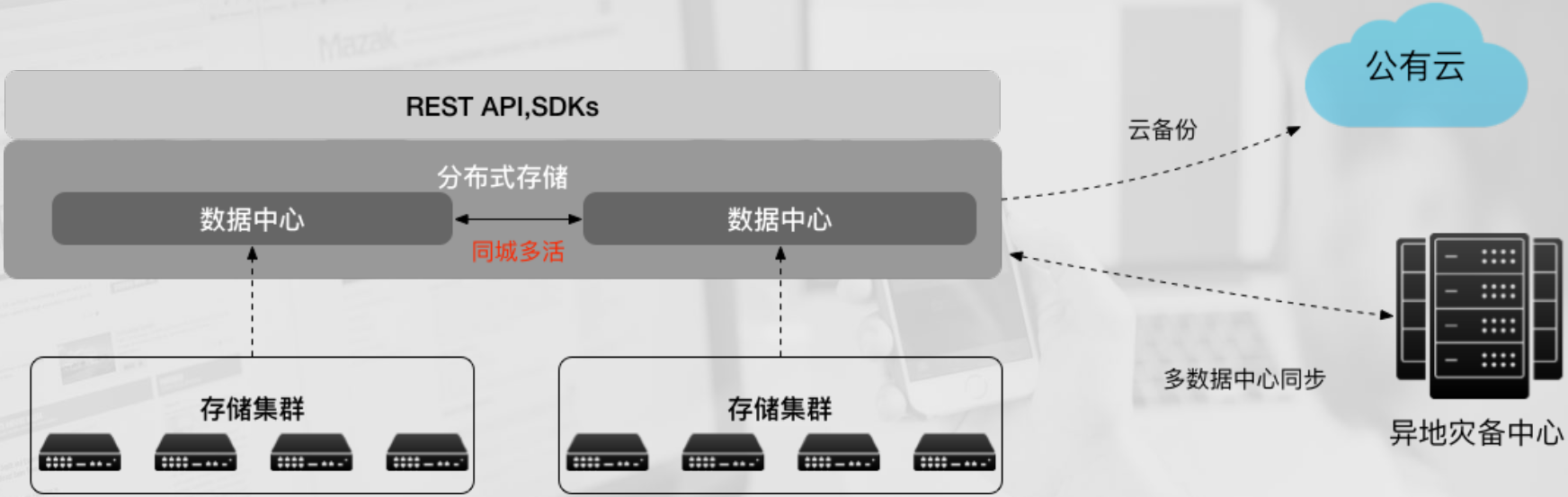
- Bucket->object 两级结构，扁平化管理
- Bucket和Object数量理论上无上限(但物理资源是有限的）
- Bucket名称全局唯一，通过Bucket name+ Key的组合来确定对象最终存储路径

Ceph构架



- http://docs.ceph.com/docs/master/architecture/

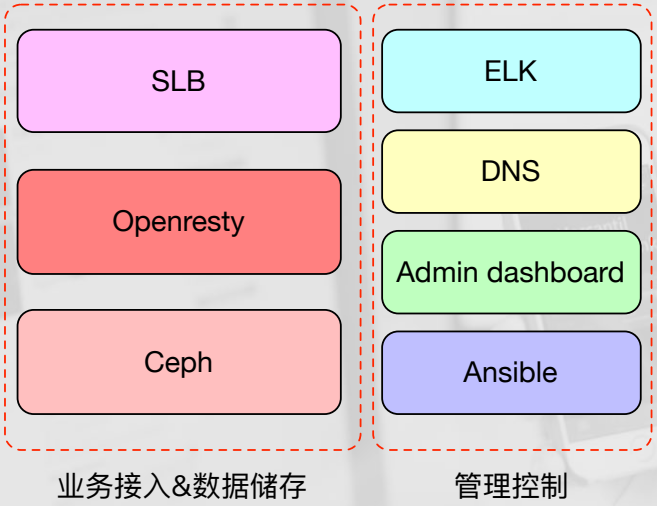
RGW同步构架



- **多集群同步**
  - 支持跨多个数据中心之间的集群多活方案(同城多活)，实现两个数据中心同时读写。
  - 支持跨数据中心的异步方式进行数据同步，实现数据中心级的异地灾备。
- **Cloud sync**
  - 支持与AWS S3等其他标准的对象存储系统进行数据同步(异步)，实现异构对象存储系统的云备份。

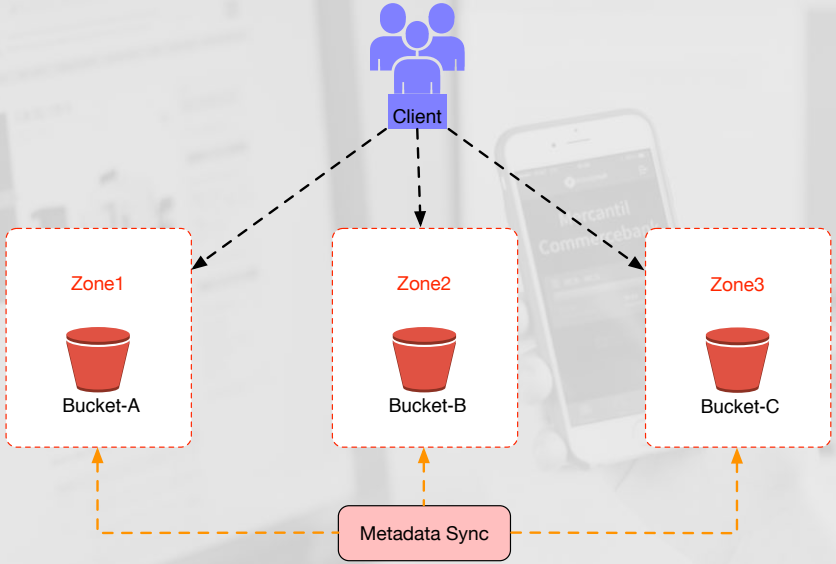
02 构架设计

功能模块组成



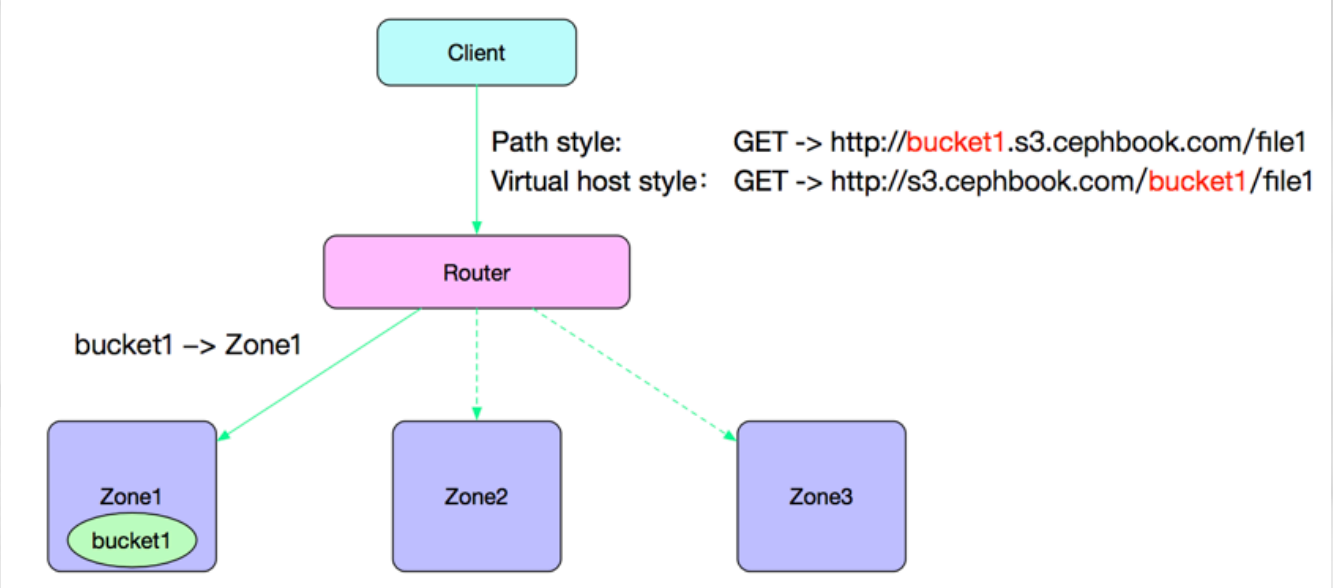
- SLB：自研基于DPDK 4层负载均衡
- Openresty：HTTP(s)请求路由分发WAF
- Ceph：Radosgw服务及数据储存
- ELK：日志处理及业务报表统计
- DNS：DNS域名解析服务
- Admin dashboard：自研Ceph运维管理平台系统
- Ansible：自动化部署及运维控制指令下发

整体构架



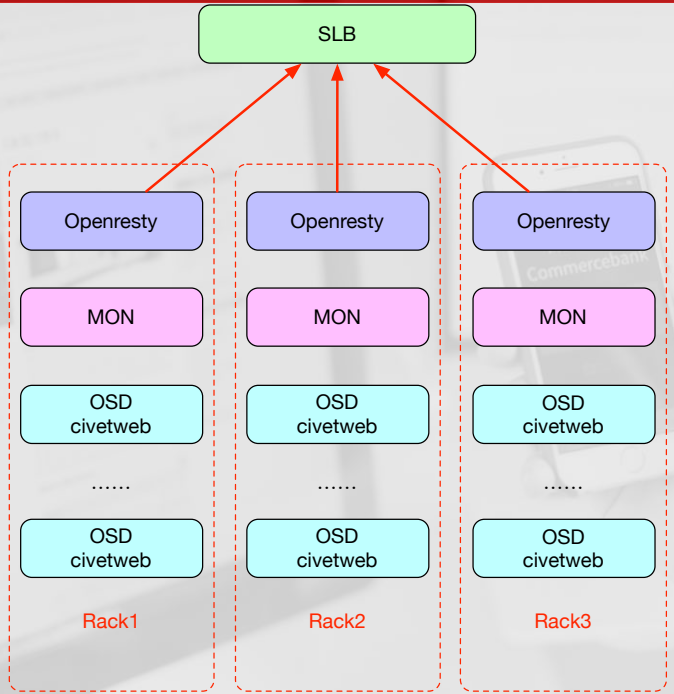
- 统一的endpoint入口，后端由多个集群(zone)组成。
- 按zone分配bucket，bucket之间无copy数据的操作。
- 按集群(zone)进行扩容，最大限度避免扩容导致的数据迁移影响业务。
- 集群之间只同步元数据信息。

跨集群路由设计



- 按bucket->zone对应关系来实现请求路由。
  - 通过Openresty实现path style类型的请求路由。
  - Virtual host style通过DNS解析到不同zone的入口IP来实现路由。
- 限制单个bucket的容量上限，避免超容量。
- 开发迁移工具，实现跨集群间的bucket数据迁移。

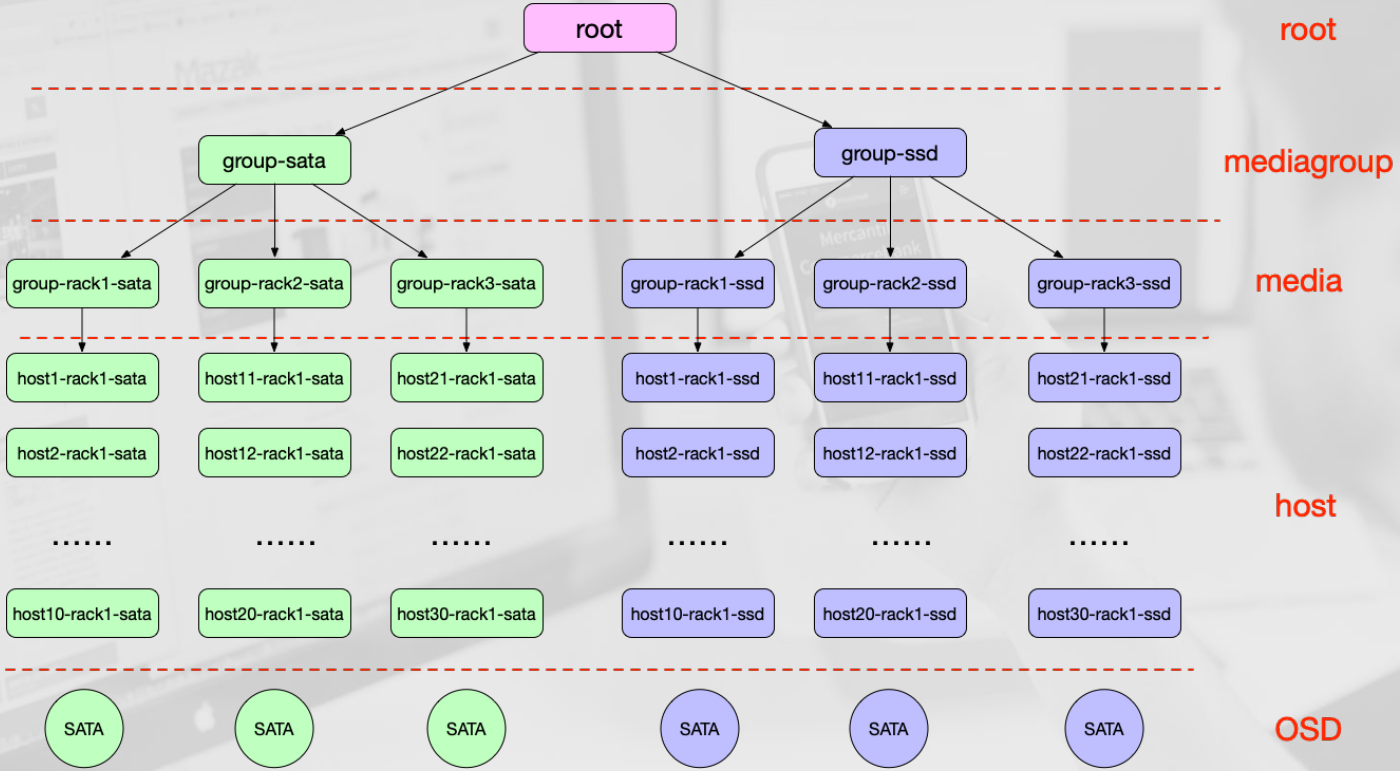
单机柜构架



- 3副本，定制crushmap实现机柜级容灾。
- OSD和RGW(civetweb)混部，提高资源利用率。
- Openresty节点通过反向代理方式连接同柜内的ciwetweb。
- 做好资源及安全隔离，拆分独立的MON、Openresty物理节点。

JD Finance

Crushmap设计

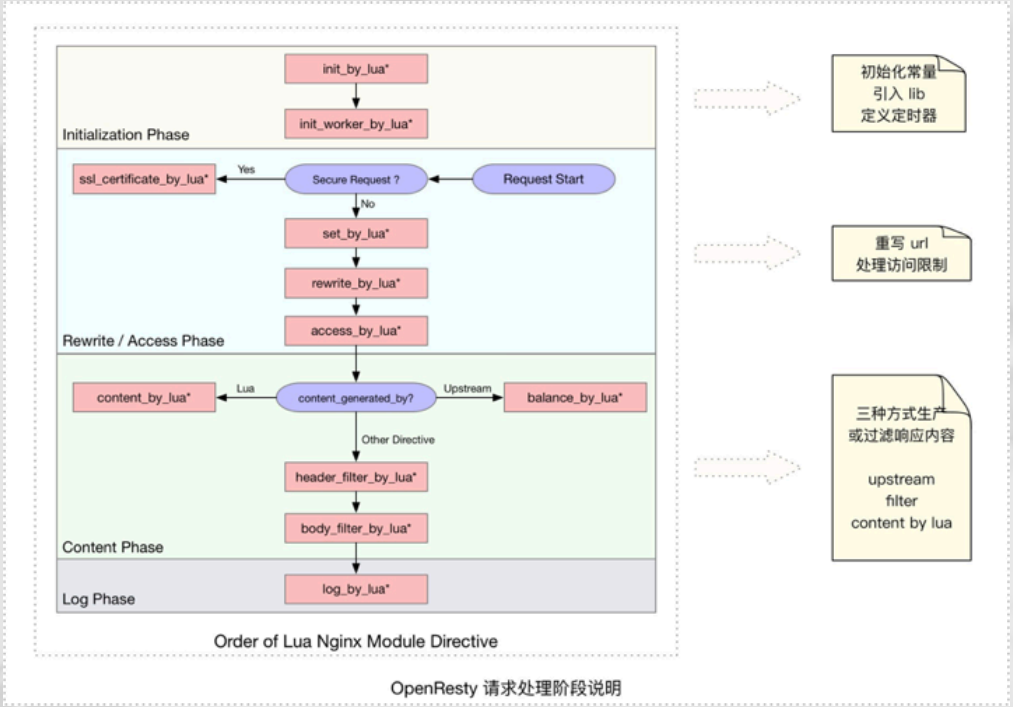


- 按储存介质拆分SATA和SSD两个类型分组
- 3副本，确保每个rack选1个副本，实现机柜级容灾。

JD Finance

02 开源工具实践

Openresty简介



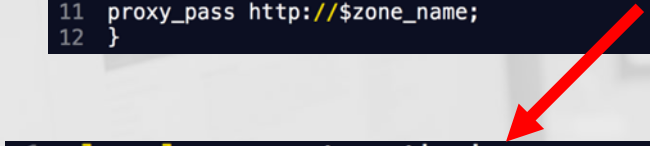
- 国人领导的开源项目，成熟稳定，大厂都在用。
- 上手快，接近C的性能，解耦神器，能把对象存储武装到牙齿！！！

JD Finance

Openresty用例——读写分离

```
upstream read_zone {
    server 10.1.1.251:801 weight=13;
}
upstream write_zone {
    server 10.1.1.253:801 weight=13;
}

location  / {
set $zone_name read_zone;
access_by_lua_file /etc/nginx/conf.d/access.lua;
proxy_pass http://$zone_name;
}
```



```
local request_method = ngx.var.request_method
if request_method == "PUT" then
    ngx.var.zone_name = "write_zone"
end
```

- Access阶段设置以下策略
  - 拆分后端节点，分为read和write两个zone
  - 默认请求发送到read_zone，仅当HTTP Method= PUT时将请求转发到write_zone。
  - 举一反三，可以实现后端各种差异化配置。

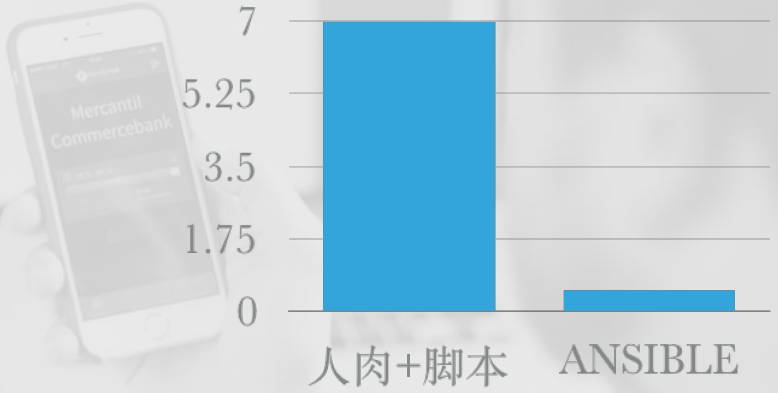
Openresty用例——WAF保护admin接口

```lua
local client_ip = ngx.var.remote_addr
local admin_from, admin_to, err = ngx.re.find(uri,"^/admin/?")
local admin_white_ip_list = {["127.0.0.1"]=true,["192.168.0.1"]=true}

if admin_from then
    if true ~= admin_white_ip_list[ip] then
        ngx.log(ngx.ERR, "Bad boy IP: ", client_ip)
        ngx.exit(ngx.HTTP_FORBIDDEN)
    end
end
```

- Access阶段设置以下策略
  - 根据客户端IP设置白名单
  - 白名单外的用户禁止访问http(s)://{hostname}/admin
  - 记录熊孩子的IP，并返回403错误

Ansible自动化部署

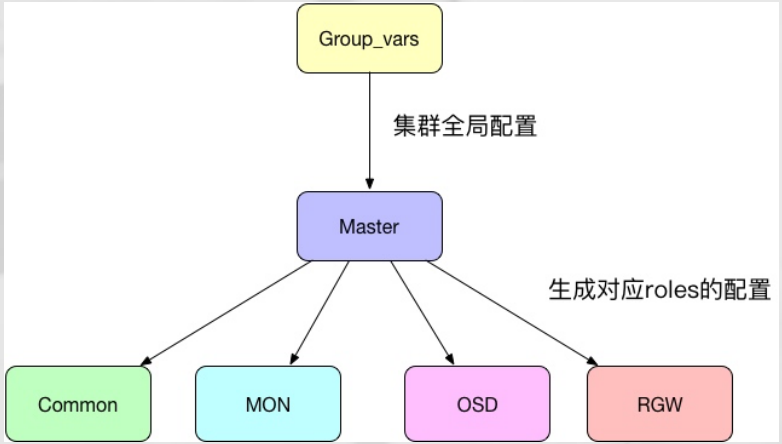


- SSH，无代理
- 功能模块丰富，避免重复造轮子
- Python，方便扩展，学习成本低...
- Ceph-deploy功能太弱，Ceph官方主推ansible部署，https://github.com/ceph/ceph-ansible

原生Ceph-ansible问题

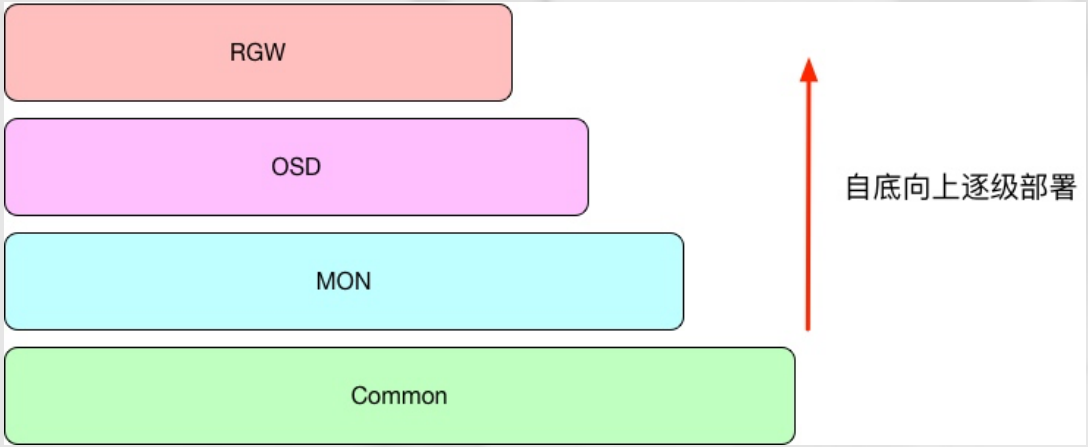
- 各种X86服务器硬件适配是问题，特别是Raid卡管理。
- 现有运维工具体系和原生的roles角色无法很好适配。
- 代码臃肿，很多功能模块用不上。
- 无法在硬件差异化的情况下区适配自定义crushmap配置
- 不同的用户习惯，与现有的运维标准操作流程适配不好。

自研ansible部署方案——角色拆分



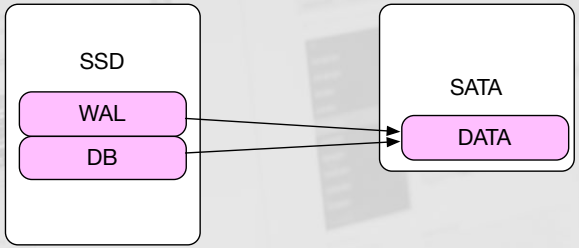
- 按功能类型拆分成Master，Common、MON、OSD、RGW几个角色，每个角色功能如下:
  - Master: 负责整个集群各个角色的初始配置文件生成
  - Common: 负责基础Ceph软件包的安装与操作系统、硬件相关配置。
  - MON: 负责MON服务的配置。
  - OSD: 负责OSD服务的配置。
  - RGW: 负责RGW服务的配置。

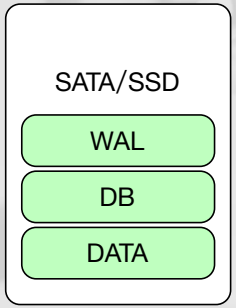
自研ansible部署方案——部署流程

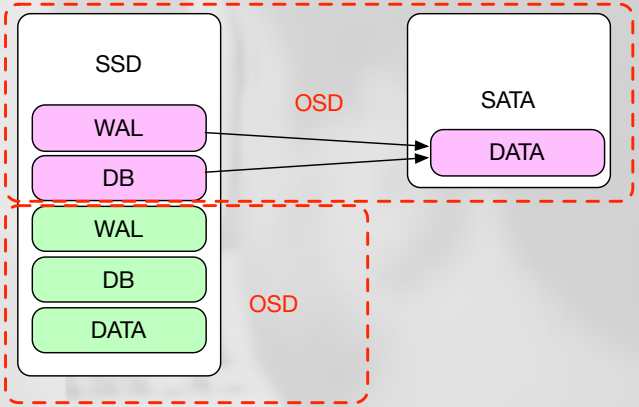


- 通过master角色生成Common、MON、OSD、RGW几个角色的配置文件
- 遵循Ceph的部署过程，从下自上依次运行Common、MON、OSD、RGW完成对应服务的部署

自研ansible部署方案——OSD部署模式







- 支持Bluestore和Filestore两种类别的SATA、SSD混合部署
  - 混合模式1：SSD作为WA和DB，SATA作为DATA
  - 混合模式2：在混合模式1的基础上再多加一组OSD
  - SATA/SSD 独占模式，划分WAL、DB、DATA

JD Finance

自研ansible部署方案——磁盘分区管理

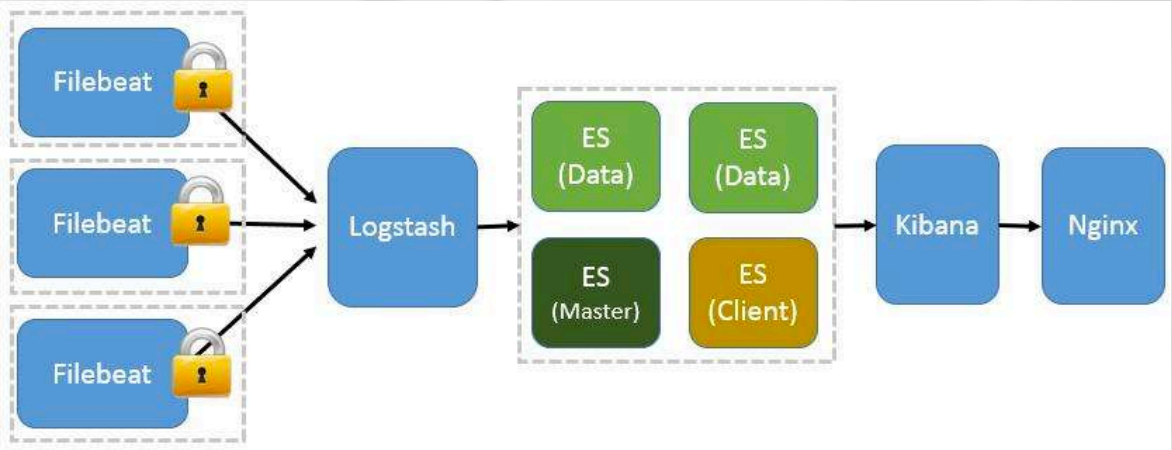
```
[root@JXQ-97-8-41 supdev]# blkid
/dev/sda5: UUID="e5ea6129-5272-4064-8020-b8b52ac87847" TYPE="ext4" PARTLABEL="primary" PARTUUID="9505b77f-3070-4246-8c49-9ca252b120bb"
/dev/sda2: UUID="4404e242-6700-47bf-9373-0a855c44c2e4" TYPE="swap" PARTLABEL="primary" PARTUUID="8ebec8d4-7d6e-483b-85c4-7daf3bc2f93d"
/dev/sda3: UUID="b9607b87-9416-4419-8497-ab22385ee93d" TYPE="ext4" PARTLABEL="primary" PARTUUID="6e93340b-de88-4197-a03e-3810df90c063"
/dev/sda4: UUID="a9c570b5-78ba-484a-a54d-b4a8ed7d370d" TYPE="xfs" PARTLABEL="primary" PARTUUID="7ef0c2cc-9009-4454-809a-8b5eafb9e984"
/dev/sdb6: UUID="de001899-804e-4fdb-99fb-71ff8d22b062" TYPE="xfs" PARTLABEL="108-data" PARTUUID="41cf61f4-4ccf-5795-95ca-ffe5a31b3329"
/dev/sdc6: UUID="b89b83e0-f958-449a-8909-76bd9d817a8f" TYPE="xfs" PARTLABEL="109-data" PARTUUID="14b559fd-8e28-5f58-91bd-5cd435edff2d"
/dev/sdd1: UUID="f7a788f4-388d-4f8a-8719-d20ab4b8c83b" TYPE="xfs" PARTLABEL="100-data" PARTUUID="e678ada9-8f4d-56d9-8f78-d942276c2504"
/dev/sdf1: UUID="8db03fc1-03f0-4b9e-aa22-360c4b5dfdea" TYPE="xfs" PARTLABEL="102-data" PARTUUID="ae07da0f-3c07-5f9e-bc9d-ecd15d63dc77"
/dev/sdg1: UUID="65b1e076-2c30-438a-b043-a6639910a39a" TYPE="xfs" PARTLABEL="103-data" PARTUUID="ccf75dfe-07cc-57d2-8f31-d6ab743c14ba"
/dev/sdh1: UUID="a033dd46-c28c-4d49-828a-ebc177e10925" TYPE="xfs" PARTLABEL="104-data" PARTUUID="ae0de0ec-0fe8-56f7-81d5-986e02f50777"
/dev/sdj1: UUID="14055a87-3fcd-437e-a7e1-0e52f6b26e3d" TYPE="xfs" PARTLABEL="106-data" PARTUUID="f7eb6e95-4fd1-5f96-b4c3-78b3cef4426d"
/dev/sdk1: UUID="2b859c8d-c276-4649-8d78-dcd739b7ebf6" TYPE="xfs" PARTLABEL="107-data" PARTUUID="6c66a831-a6db-539c-9701-b844a233f23d"
/dev/sde1: UUID="972e4460-a561-4653-abc2-b3f266c99859" TYPE="xfs" PARTLABEL="101-data" PARTUUID="8e8801be-3d8e-530e-b072-9d22da18fbc1"
/dev/sdi1: UUID="678b6e5b-d156-4dd7-a33d-4ebccd11b46c" TYPE="xfs" PARTLABEL="105-data" PARTUUID="20a9773a-f8c1-544a-adb3-a9de8a755603"
/dev/sda1: PARTLABEL="primary" PARTUUID="77c76ab6-327d-4255-a726-cbbb0d7847a8"
/dev/sdb1: PARTLABEL="100-journal" PARTUUID="b581b78a-32bd-5c22-81ed-b62a61de3ec4"
/dev/sdb2: PARTLABEL="101-journal" PARTUUID="9a79f93c-89c0-5d18-af55-e901016a4123"
/dev/sdb3: PARTLABEL="102-journal" PARTUUID="f5594678-72d4-5b2e-91d4-7528667e8a48"
/dev/sdb4: PARTLABEL="103-journal" PARTUUID="87bd6ca3-bedb-55d2-82d6-f16e4d4a3c61"
/dev/sdb5: PARTLABEL="108-journal" PARTUUID="0d1a7874-3e68-5591-84d7-89a9d4ac78bc"
/dev/sdc1: PARTLABEL="104-journal" PARTUUID="ce8a08ad-4cac-5a0b-b7b6-63931bd5a990"
/dev/sdc2: PARTLABEL="105-journal" PARTUUID="c0164189-08ce-5159-aa1f-faeaacf6eb8a"
/dev/sdc3: PARTLABEL="106-journal" PARTUUID="bb9e6934-a1bd-50c4-bfd6-1630fcaadd71"
/dev/sdc4: PARTLABEL="107-journal" PARTUUID="1020422a-cb35-5669-b724-393c5543d17b"
/dev/sdc5: PARTLABEL="109-journal" PARTUUID="e860ebff-b733-5062-aabd-9e4db238b3f5"
```

- 标识出每块磁盘及分区的类型，方便换盘
- 通过PARTUUID标识磁盘及启动OSD，避免盘符乱序

自研ansible部署方案——部署流程

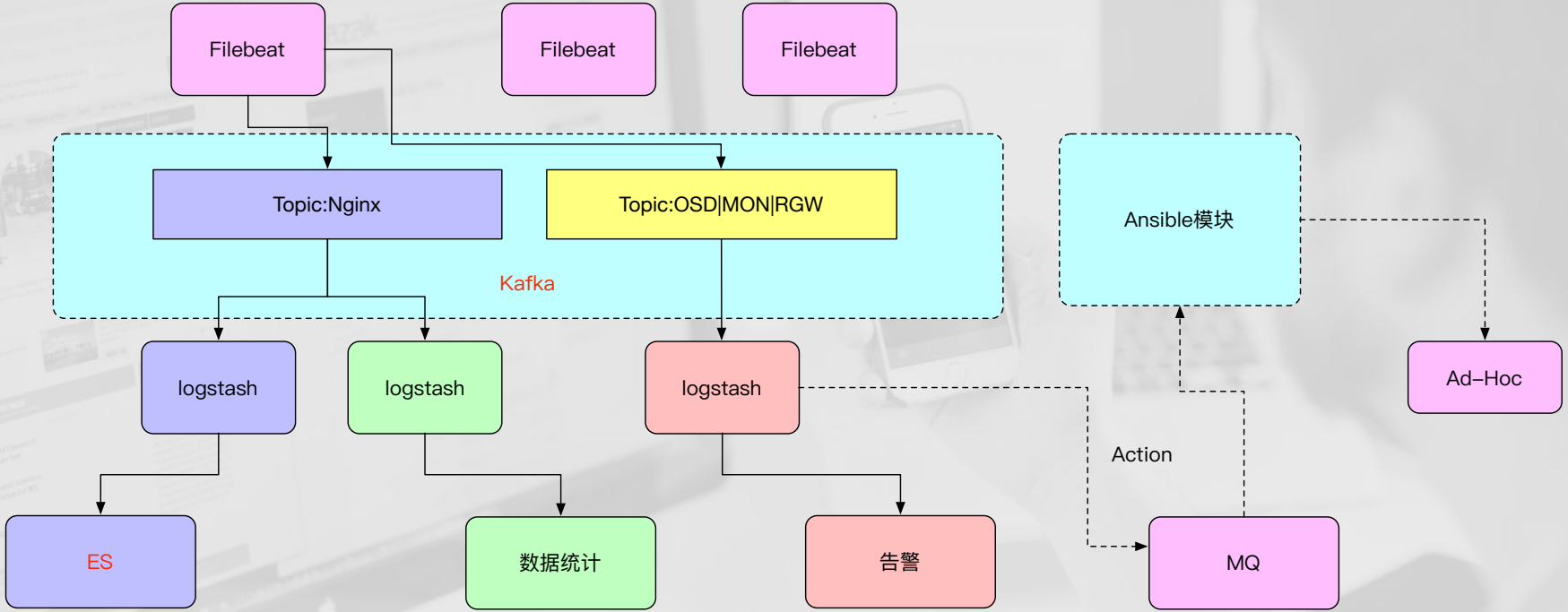
```
/dev/sdb11: UUID="5dRXvA-5mpb-3n1l-Gxyc-zuPZ-ZFNV-zaFJc1" TYPE="LVM2_member" PARTLABEL="osd-8-data" PARTUUID="dc5f9268-c672-5ac9-a01f-821abcc6cd92"
/dev/sdc11: UUID="3IbjBF-w1c5-56Yo-yZTT-5npf-YmFu-zcBvJn" TYPE="LVM2_member" PARTLABEL="osd-9-data" PARTUUID="7813a8ae-f809-5aaa-8f58-cabdfac2931e"
/dev/sda1: PARTLABEL="primary" PARTUUID="05c3067c-ff98-4bd3-bed4-ccabca8d15e9"
/dev/sdb1: PARTLABEL="osd-0-wal" PARTUUID="34ab4d2f-0c7d-543a-a0ed-0f1451bcd2c8"
/dev/sdb2: PARTLABEL="osd-0-db" PARTUUID="4d51a388-a74f-5016-852c-19999cdf0cc9"
/dev/sdb3: PARTLABEL="osd-1-wal" PARTUUID="fad5f808-b643-51e3-98e8-33cf4ed859d5"
/dev/sdb4: PARTLABEL="osd-1-db" PARTUUID="72afe6cc-b0aa-5d6a-a591-8dc3e6bf7e06"
/dev/sdb5: PARTLABEL="osd-2-wal" PARTUUID="962cf5a2-7964-5322-b9c7-520222a9a413"
/dev/sdb6: PARTLABEL="osd-2-db" PARTUUID="6e77a7f1-a125-55d0-b467-2d0553776d9c"
/dev/sdb7: PARTLABEL="osd-3-wal" PARTUUID="25703087-e26d-57b5-ac97-2efa616de610"
/dev/sdb8: PARTLABEL="osd-3-db" PARTUUID="1d5221a2-228e-55d4-a265-a4ad91361dcf"
/dev/sdb9: PARTLABEL="osd-8-wal" PARTUUID="15938517-8156-5ec9-b2f6-9d957ece91d7"
/dev/sdb10: PARTLABEL="osd-8-db" PARTUUID="71e8c94f-1ed3-5c66-97db-665e91823ca9"
/dev/sdc1: PARTLABEL="osd-4-wal" PARTUUID="26fd18e5-7a23-506f-9944-5f855e355a6a"
/dev/sdc2: PARTLABEL="osd-4-db" PARTUUID="ad25f9a6-69af-5ac7-9c08-49239567c7d9"
/dev/sdc3: PARTLABEL="osd-5-wal" PARTUUID="653a9752-8ddc-532e-9d27-91804d13ef8b"
/dev/sdc4: PARTLABEL="osd-5-db" PARTUUID="9789b0c7-abd3-544c-a400-6e78306557ee"
/dev/sdc5: PARTLABEL="osd-6-wal" PARTUUID="e7c53743-e026-5275-8365-4ddce10bc74a"
/dev/sdc6: PARTLABEL="osd-6-db" PARTUUID="ec314eeb-e502-53c5-9e28-1a1be535fd62"
/dev/sdc7: PARTLABEL="osd-7-wal" PARTUUID="fbd3ebbb-b80f-528e-b456-15a36eef9693"
/dev/sdc8: PARTLABEL="osd-7-db" PARTUUID="1ca8cba0-452e-5350-9367-f9812482845e"
/dev/sdc9: PARTLABEL="osd-9-wal" PARTUUID="7621e527-30e8-5012-b941-baf494b3ba7d"
/dev/sdc10: PARTLABEL="osd-9-db" PARTUUID="b4688200-187f-51ff-aa6c-7ecf44a7829e"
```

ELK简介



- 开源日志管理首选方案
  - https://www.elastic.co/cn/products
- 成熟稳定，经过大规模生产验证
- 图形图表可定制化强

基于ELK的自动化运维构架



JD Finance

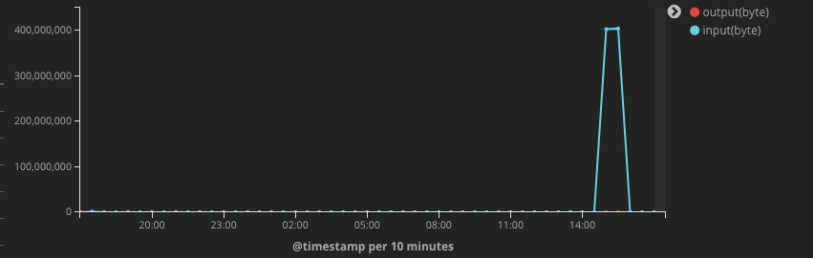
基于ELK服务质量监控

请求状态码分布

Table | Request | Response | Statistics

status: Descending	Count
200	3,158
404	509
405	18
400	11
204	2

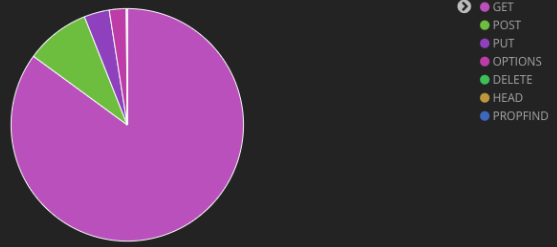
进出流量累计



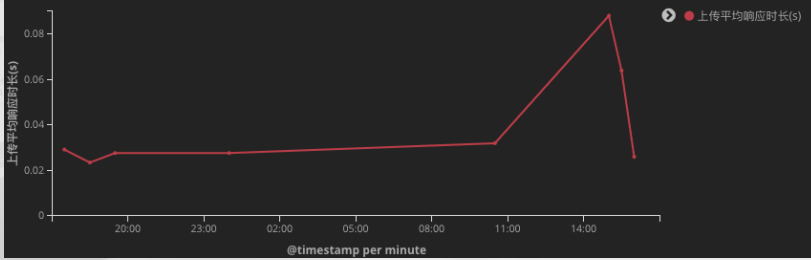
请求类别统计

request_method.keyword: Descending	Count
GET	3,145
POST	331
PUT	130
OPTIONS	87
DELETE	2

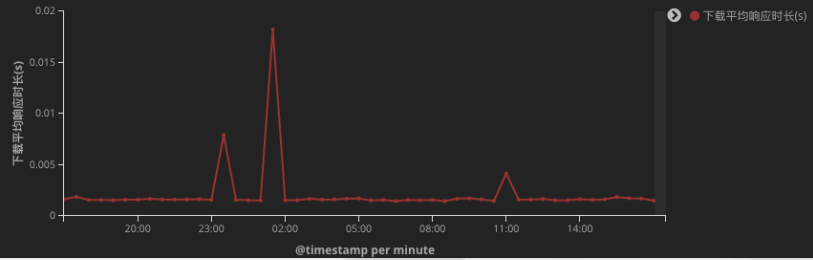
请求类型分布



上传平均响应时长



下载平均响应时长



nance

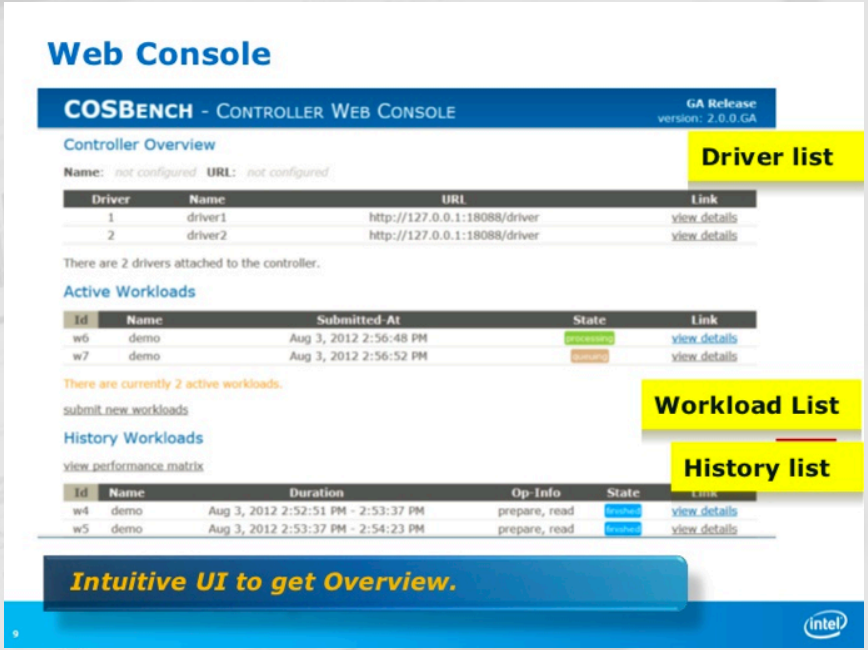
日志可视化

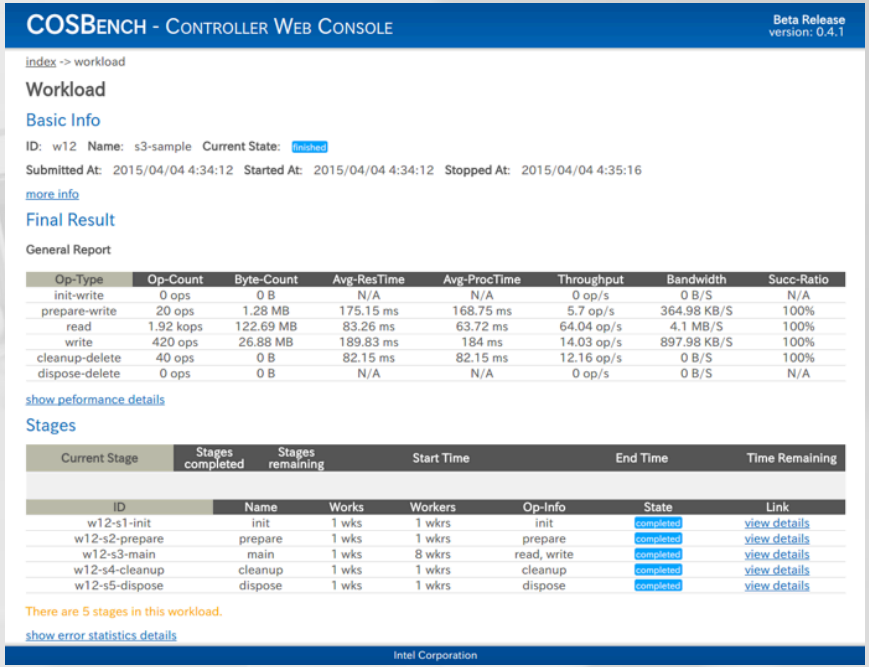
Table | JSON

@timestamp		November 2nd 2018, 17:03:04.000
_id		AWbTqWPE8GvhV4mwSmaq
_index		-2018.11
_score		-
_type		doc
body_bytes_sent		0
bucket_name		null
content_length		0
http_host		-
http_referer		-
http_user_agent		-
http_x_forwarded_for		-
realm_name		cn
remote_addr		[redacted]
request		HEAD / HTTP/1.0
request_length		19
request_method		HEAD
request_time		0.011
request_uri		/
scheme		http
server_addr		[redacted]
server_protocol		HTTP/1.0
source		[redacted]
status		200
time_local		[02/Nov/2018:17:03:04 +0800]
upsteam_response_time		0.003
upstream_addr		127.0.0.1:9000
x_jrss_request_id		tx0000000000000003a140d-005bdc12c8-1aeae-cn-seer-1
zone_name		cn-seer-1
zonegroup_name		cn-seer

JD Finance

性能压测





- Intel开源项目(基本已经停止维护)
  - https://github.com/intel-cloud/cosbench

JD Finance

04 一些经验分享

OSD分布不均——PG调优(Jewel之前版本)

◆调优前:

Crush算法并不能确保PG绝对均匀的分布到所有OSD中。

OSD磁盘容量的利用率低下，经常出现部分OSD出现near full(≥85%)，而部分OSD磁盘利用率居然40%不到。

除了容量分布不均匀以外，磁盘的IO性能也同样分布不均，部分OSD磁盘会经常出现slow Request。

◆调优后:

PG分布非常均匀，基本上在3~5%波动。

磁盘容量平均利用率问题基本上解决。

集群整体性能也有5~10%左右提升。

OSD分布不均——PG调优(Jewel之前版本)

◆1. 安装软件包

pip install crush

◆2 . 新建pool，注意集群状态需要为Health OK，同时必须使用straw2算法

ceph osd pool create {pool-name} {pg-num} {pgp-num}

◆3. 导出report

ceph report > report.json

◆4. 调优ID为3的pool的pg分布

crush optimize --crushmap report.json --out-path optimized.crush --pool 3

◆5. 导入调优后的结果

ceph osd setcrushmap -i optimized.crush

注意调优操作最好在上线之前进行，不要在运行中的生产系统操作。

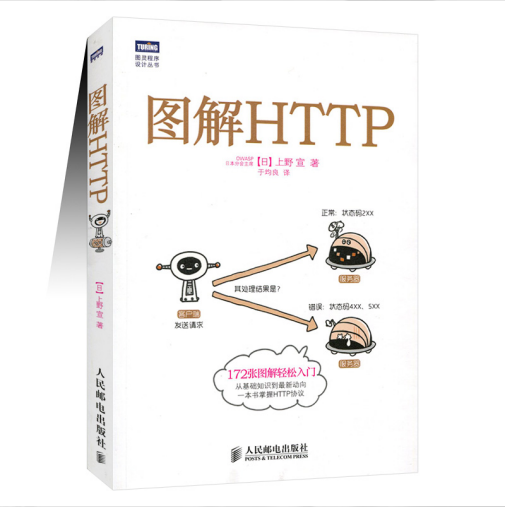
参考: http://crush.readthedocs.io/en/latest/ceph/optimize.html

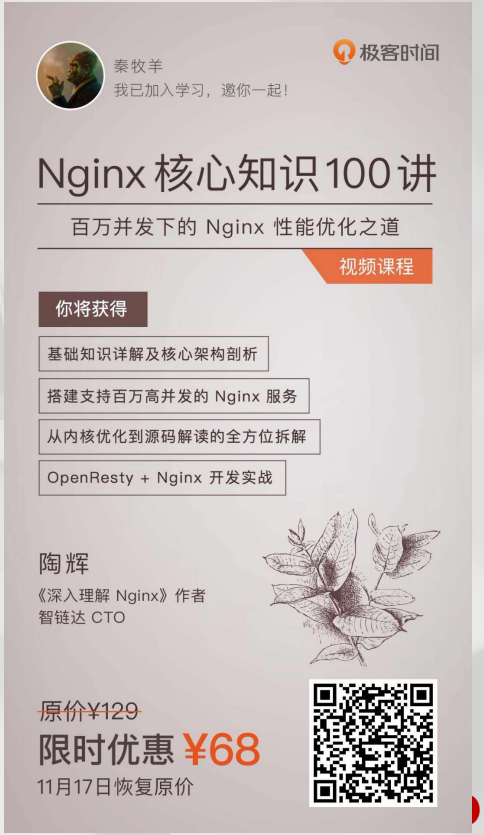
Luminous以上版本调优可以参考: http://url.cn/5rLLEZE

经验分享

- 一定要有SSD作为index pool
- 每个bucket的shard数量要提前规划好（每个shard大概承载20W左右的objects数量），最好不要去reshard
- 每个OSD的omap(RocksDB)控制在10G以内
- Auto reshard有坑，最好不要用
- 如果是用到multisite，最好把log pool也设置成SSD。

学习资料推荐





Finance

其他

- 京东数科云团队招聘分布式储存工程师（北京、杭州），欢迎邮件到 jrguofeng@jd.com
- 其他cepha相关技术内容可以关注下面微信公众号

JD Finance

Thanks