滴滴

滴滴一下 美好出行

滴滴对象存储系统的架构演进实践

2018年10月

对象存储系统

- 存储静态数据
  - 图片，音视频，网站静态资源
  - 镜像
  - 备份数据
- 一次写，不修改，多次读，较少删除
- HTTP访问接口
- 海量文件
- 强数据可靠性的需求
- 典型系统
  - Haystack, Facebook (Finding a needle in Haystack: Facebook's photo storage, OSDI'10)
  - Azure, Microsoft (Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency, SOSP'11)
  - Ambry, LinkedIn (Ambry: LinkedIn's Scalable Geo-Distributed Object Store, SIGMOD'16)

GIFT V1

Gift，是为了解决公司小文件存储问题而开发的对象存储系统

- **访问接口**
  - http://<domain>/<bucket>/<object>，bucket全局唯一，bucket和object均为用户指定
  - 上传，删除，下载
- **架构**
  - **接入服务**为用户提供文件上传、下载、查询等功能
  - **存储服务**用于存储文件数据以及元数据
    - 数据使用SeaweedFS存储
    - 元数据使用Mysql进行存储
- SeaweedFS
  - Facebook Haystack的开源实现，实现小文件存储
  - Haystack是Facebook用于存储图片等小文件的对象存储系统
  - Haystack的核心思想是采用多对一的映射方式，把多个小文件采用追加写的方式聚合到一个POSIX文件系统上的大文件中，从而大大减少存储系统的元数据，提高文件的访问效率

SeaweedFS

Architecture

- Volume Server Cluster
  - 每个volume server对应一个磁盘，每个磁盘有一系列的volume文件，每个volume文件默认大小为30GB，用于存储小对象
  - 每个volume文件对应一个index文件，用于记录volume中存储的小对象的偏移和长度，在volume server启动时缓存在内存中
- Master Server Cluster
  - 运行raft协议，维护集群的一致性状态，一般与volume server混部
  - 负责对象写入时volume server的分配
  - 负责数据读取时volume id到volume server地址映射

GIFT V1

GIFT V1版本随着业务量不断增加存在问题也越来越多，主要表现如下：

- 随着业务量不断增加（TB -> PB）架构无法支持PB级存储
  - 数据存储SeaweedFS是有中心设计模式，集群规模增大存在数据存储容量、并发访问性能瓶颈问题
    - 读写请求必须先经过master server，再访问volume server获取数据。master server运行raft协议，只有一个leader，其它为follower。只有leader处理读写请求，存在单点性能瓶颈。在高并发的情况下，访问延迟明显的增加
  - 元数据的存储为单库设计，所有用户集群共享，QPS受限

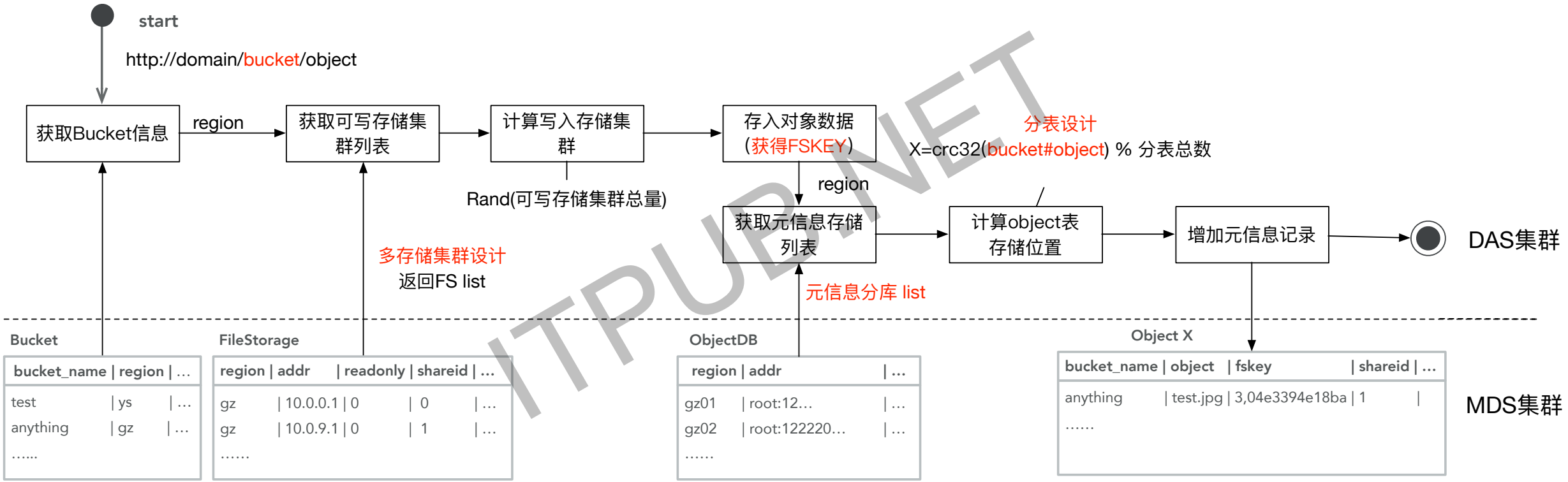
GIFT V2

V2解决方案，具体如下：

- 支持多存储子集群模式进行分流解决高并发和PB级存储问题
  - 接入服务管理多个SeaweedFS集群
  - 当一个SeaweedFS集群容量达到配置的阈值时，将其标记为只读，新数据写入其它集群
  - 支持SeaweedFS集群的动态加入
  - 集群扩容过程
    - 一个新的SeaweedFS集群会向接入服务注册
    - 接入服务将其加入可写集群列表
    - 接入服务对上传文件的bucket, object信息做hash，选取一个可写集群，进行写入操作
- 元数据支持多库分表模式，可对不同用户集群采用不同库，解决海量文件并发访问问题

GIFT V2

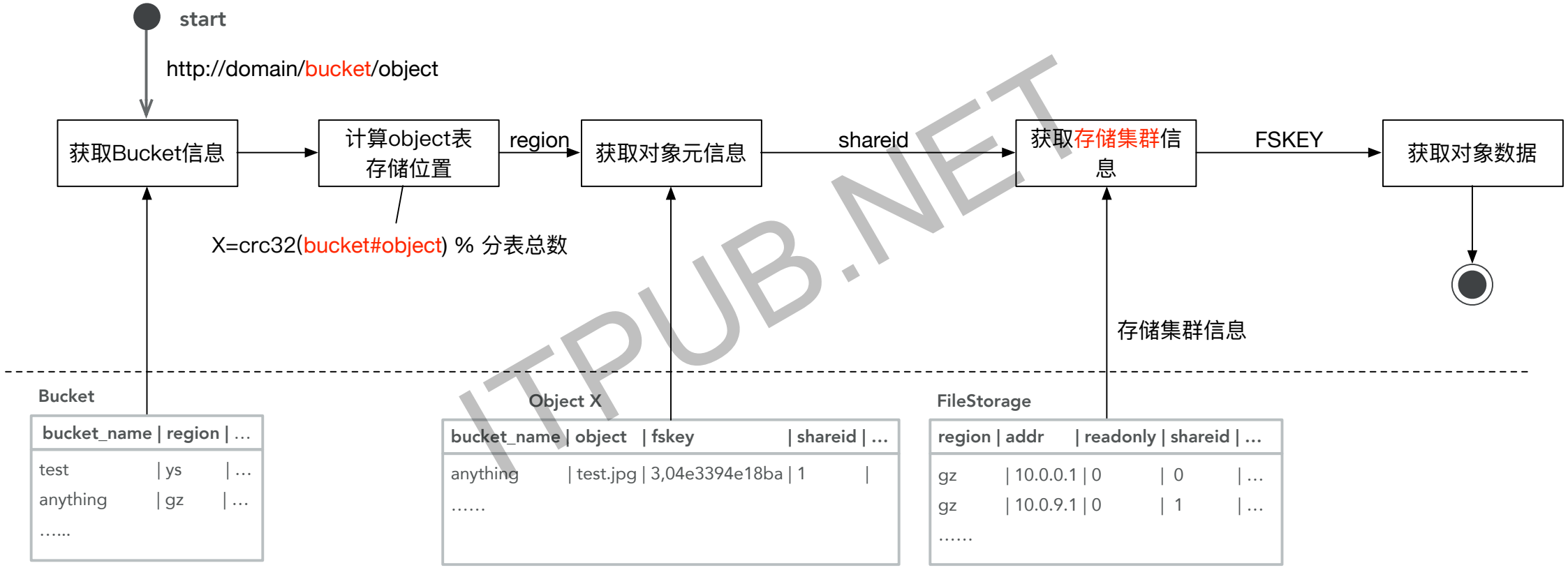
上传采用**多存储集群**进行并发分流、元数据采用**分库分表**，具体如下：



滴滴 滴滴一下 美好出行

GIFT V2

下载操作设计



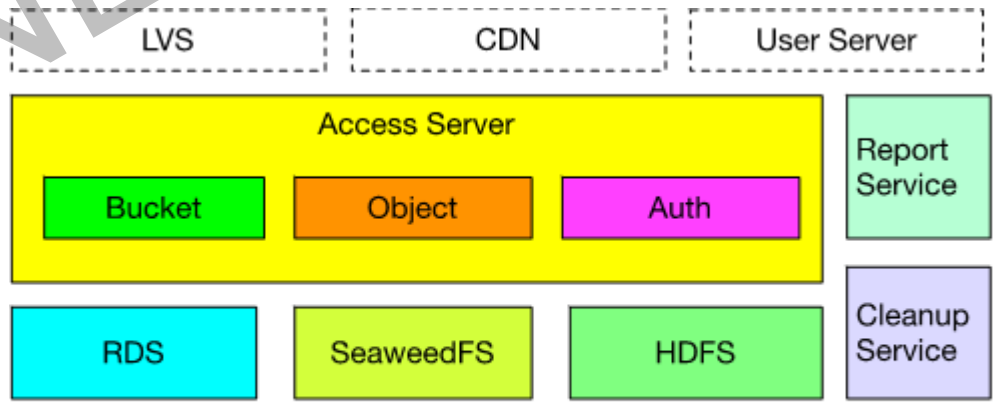
GIFT V3.0

研发动机

- 支持对外服务
- 支持大文件存储

GIFT V3.0延续GIFT V2整体架构设计，接入层重新设计

- **接入服务**提供认证和数据操作接口
- **存储服务**用于存储数据
  - 小文件存储在SeaweedFS
  - 大文件存储在HDFS
- **Report Service**用来定时推送用户统计量信息
- **Cleanup Service**用来删除过期数据
- **RDS**主要用来存储object和fskey对应关系以及配置表信息



GIFT V3 架构图

GIFT V3.0存在问题

- SeaweedFS
  - 运维复杂，不支持故障自恢复、需手动恢复数据
  - 不支持数据自动rebalance
  - 不能有效支持大文件
  - Master server维护volume id与volume server的映射关系，为单节点，所有请求经过master server
  - 不支持纠删码
- HDFS
  - 存在NN问题，中心化设计模式
  - 在GIFT应用场景下，元数据已由RDS存储处理，存储服务应尽量采用无中心化的模式
  - 不适合存储小文件
- 大小文件采用不同存储
  - 维护成本较高

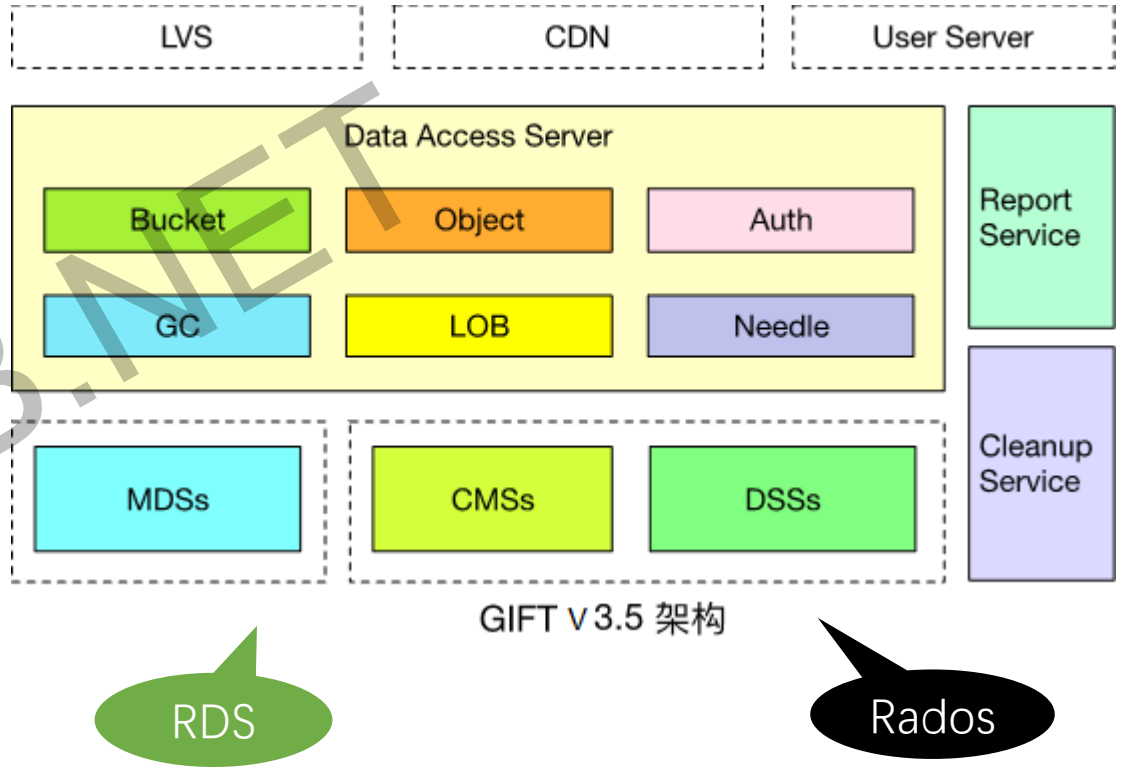
GIFT V3.5

- 存储底层采用Ceph rados object storage
- 同时支持大小文件
  - 小文件合并存储
  - 大文件分片存储
  - 异步GC
- 支持热升级
- 支持异地备份
- 集群扩容数据不迁移

GIFT V3.5架构

GIFT V3.5架构延续GIFT V3.0整体架构设计，底层存储使用**RADOS组件**。具体如下：

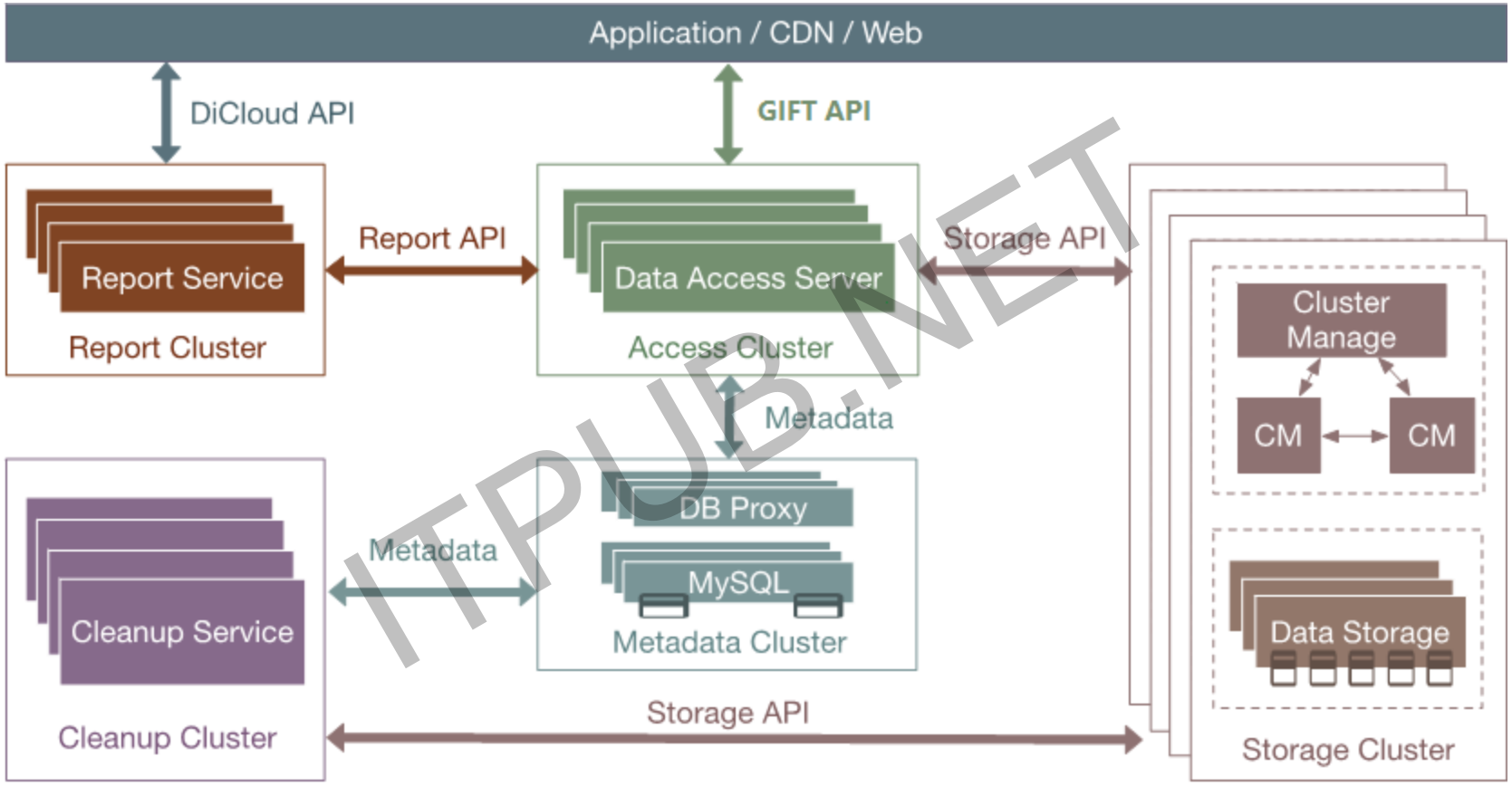
- DAS：接入服务提供认证和文件访问协议处理和小文件聚合、大文件分片、资源GC等等
- RPS：上报服务用来定时推送用户统计量信息
- CUS：清理服务提供清理过期数据
- CMS （RADOS Mon组件） ：集群管理服务是整个系统的管理者，维护了CRUSH map、PG map、OSD map等信息，保证系统整体状态的一致性
- DSS（RADOS OSD组件）：数据存储服务主要的功能是存储数据，处理数据的复制、恢复、平衡，并通过检查其它OSD 守护进程的心跳来向CMS提供一些监控信息
- MDS（RDS组件）：元数据服务



滴滴
滴滴一下 美好出行

GIFT V3.5

- 架构



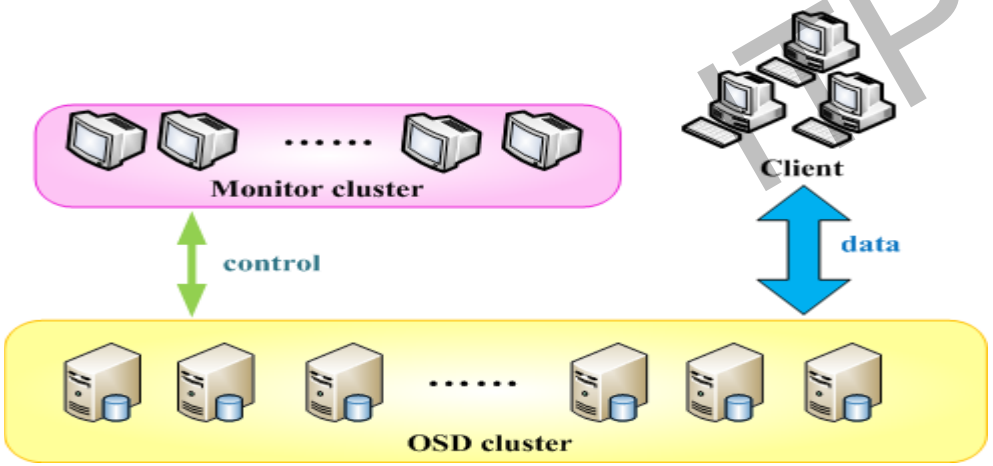
Why RADOS

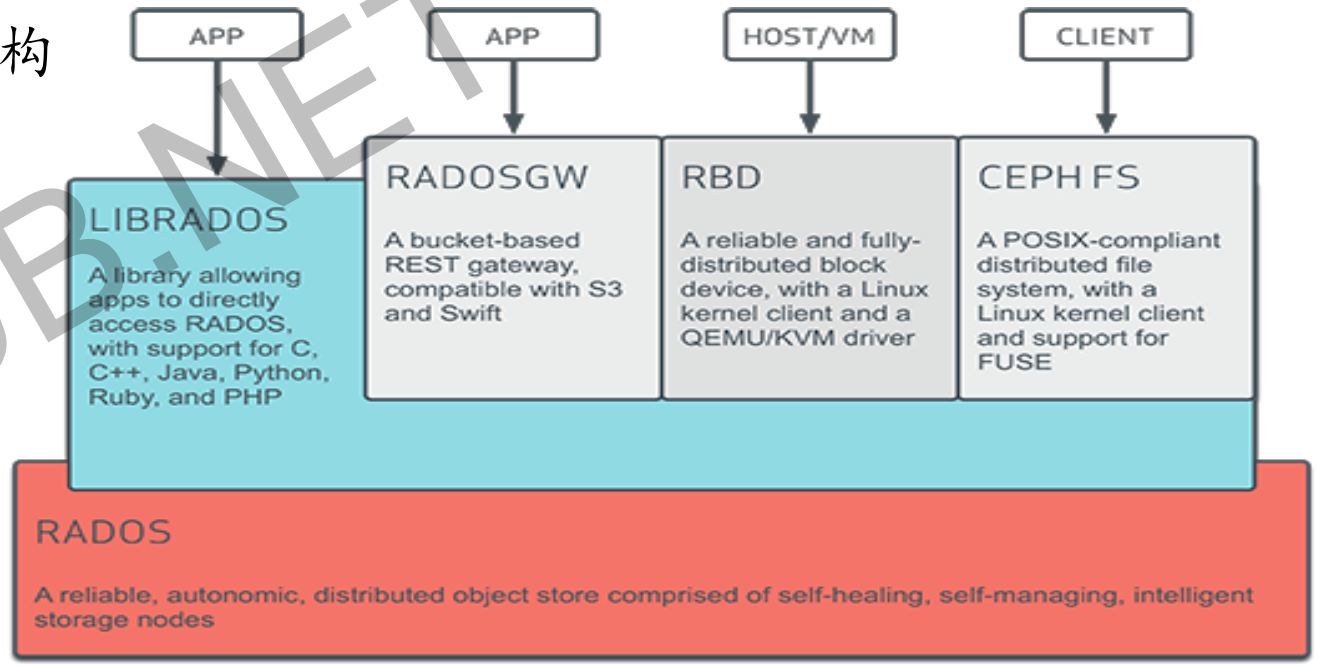
RADOS是国际上广泛部署使用的大规模分布式存储系统Ceph的底层存储组件，提供可伸缩、高可靠的对象存储

- 高数据可靠性，数据支持多副本或纠删码，支持scrub发现静默数据错误
- 无元数据服务、去中心化的设计，良好的横向扩展能力，支持PB级存储
- 运维简单，数据自恢复，集群自伸缩，数据自平衡
- 所有组件集群化设计，无单点故障
- RADOS在生产环境的应用广泛

Ceph架构

- **Client**
  - 提供标准块，文件接口的访问能力
- **Monitor**
  - 监视和维护整个存储系统状态和拓扑结构
- **OSD**
  - 存储数据和元数据

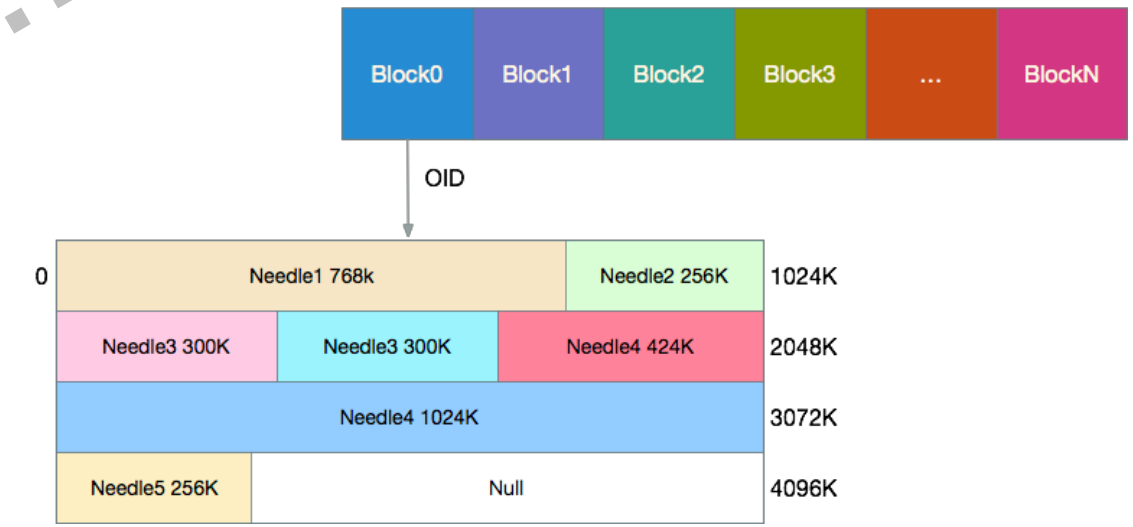




Gift V3.5

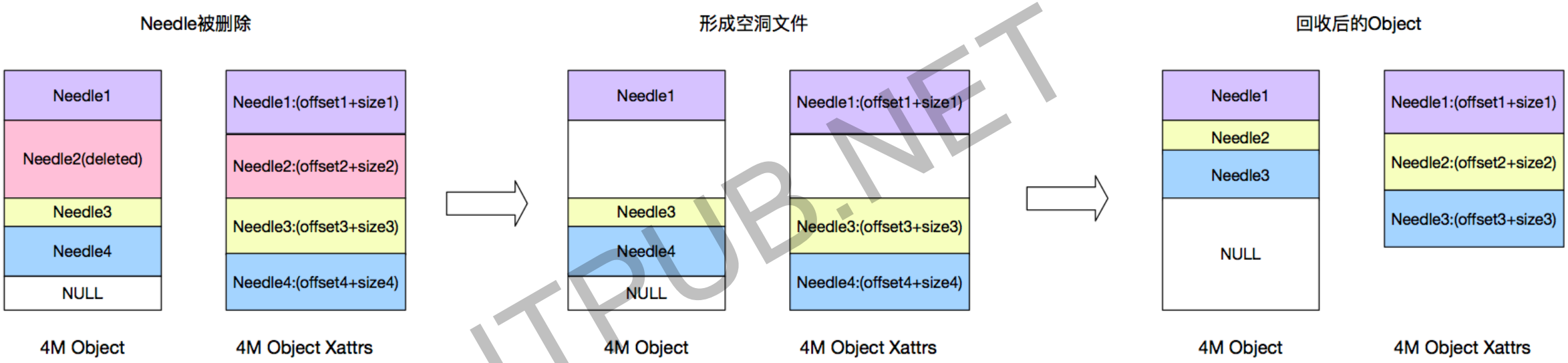
- 大文件分片存储
  - 对于大于4M（可配置）以上大文件，存储到多个rados对象
  - 提高大文件访问性能

- 小文件合并存储
  - 小文件合并存储到一个rados对象
  - rados对象元数据记录小文件在对象内的位置和偏移
  - 小文件异步回收
  - 顺序写提高性能
  - 二级元数据方式减少元数据开销
  - 提高空间利用率

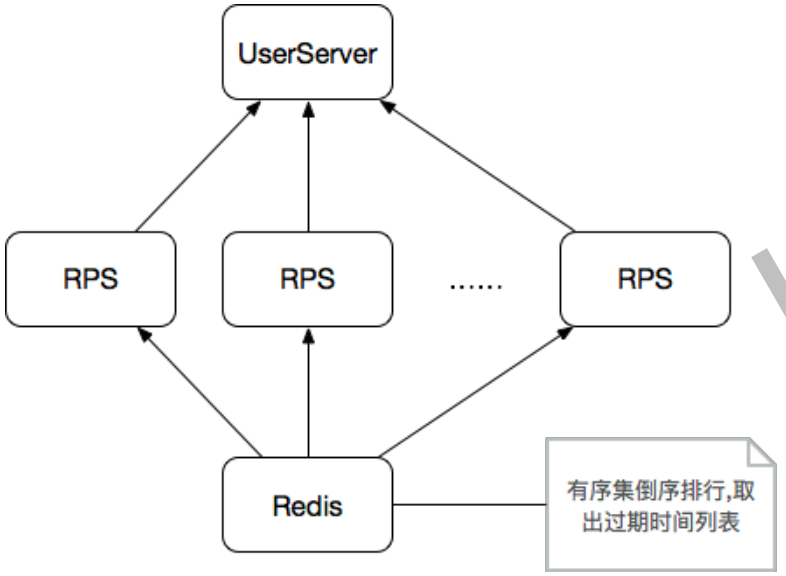


GIFT V3.5

- 异步GC
  - 对小文件的删除，只标记，当小文件所在对象空余空间较多时，再回收



- **上报服务**
  - RPS并行任务通过抢占方式获取有序表中需上报列表信息。加锁抢占任务，谁抢占成功谁来处理任务
  - 通过一个例子来说明整个方案流程，redis创建一组数据，模拟两个用户同时操作
  - 如果记录已超时了，那么进入抢占模式，两用户在同时抢占“100004”这个用户资源，如何解决冲突



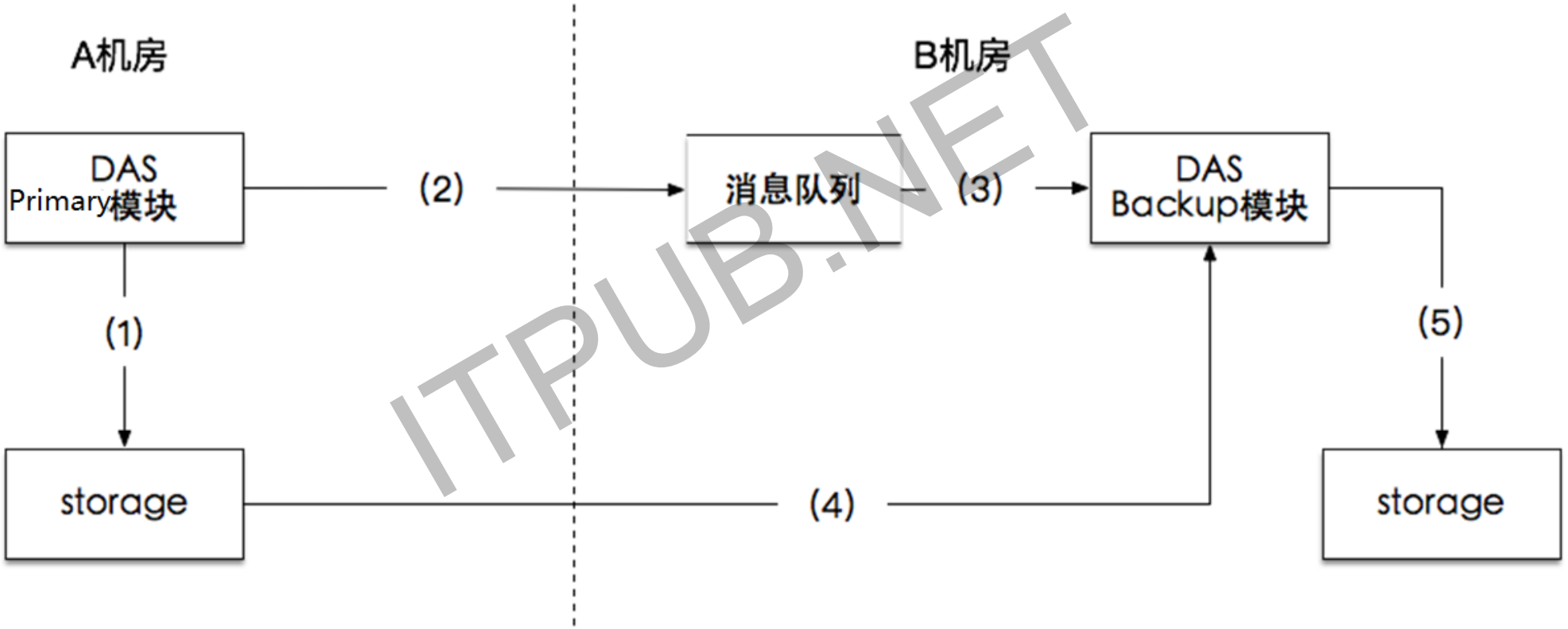

```
[root@kvm10596 ~]# redis-cli
127.0.0.1:6379> ZADD rps_all_time 1513151690 100000
(integer) 1
127.0.0.1:6379> ZADD rps_all_time 1513151680 100001
(integer) 1
127.0.0.1:6379> ZADD rps_all_time 1513151670 100002
(integer) 1
127.0.0.1:6379> ZADD rps_all_time 1513151660 100003
(integer) 1
127.0.0.1:6379> ZADD rps_all_time 1513151100 100004
(integer) 1
127.0.0.1:6379> ZADD rps_all_time 1513151120 100005
(integer) 1
```

```
127.0.0.1:6379> ZRANGE rps_all_time 0 6 WITHSCORES
1) "100004"
2) "1513151100"
3) "100005"
4) "1513151120"
5) "100003"
6) "1513151660"
7) "100002"
8) "1513151670"
9) "100001"
10) "1513151680"
11) "100000"
12) "1513151690"
```

时间	用户操作	
0		B: set lock_100004 1513151900 ex 60 nx
1	A: set lock_100004 1513151900 ex 60 nx	
结果	A: 抢锁失败，退出	B: 抢锁成功，上报数据
2		B: ZADD rps_all_time 1513159100 100004
3		B: DEL lock_100004 (删除锁)

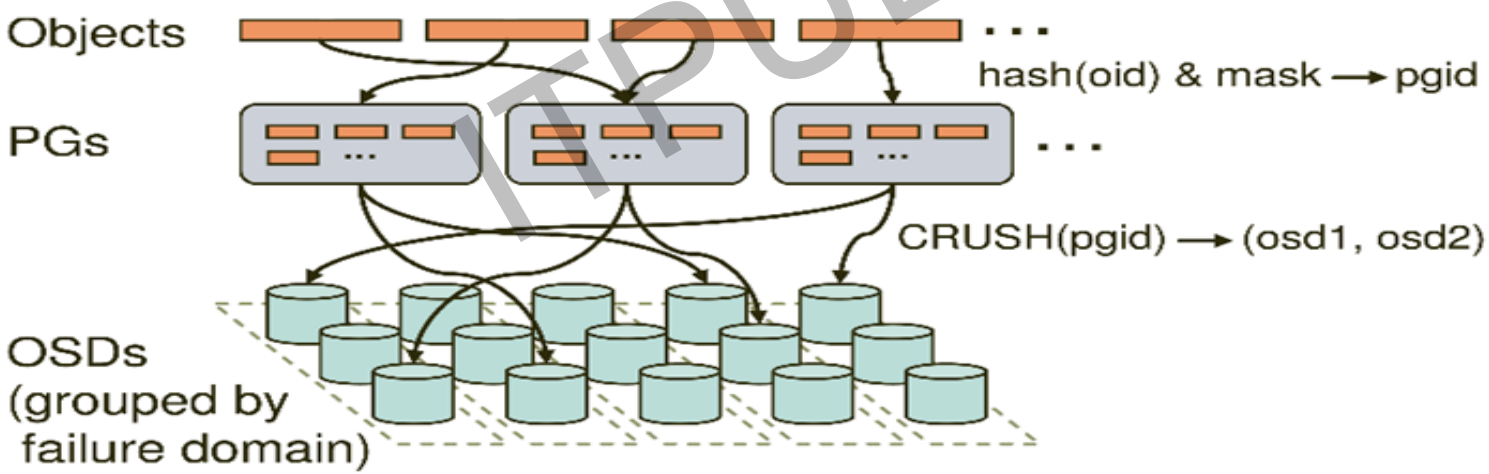
GIFT V3.5

- 异地备份



扩容无数据迁移

- **Ceph对象映射过程 – 两级映射**
  - Object通过Hash映射到PG
  - PG(Placement Group)的引入避免了object与OSD的直接映射，减小了海量对象管理的复杂性，RADOS的许多操作是以PG为单位进行
  - PG通过CRUSH映射到一组OSD， 其中一个为primary，其他为replica



扩容无数据迁移

- **Ceph的去中心化CRUSH算法，当集群扩容时会产生数据迁移，导致集群性能的抖动**
- **CRUSH规则应用粒度为pool，每个pool可指定不同的CRUSH对象映射规则**
- **基于Pool的调度**
  - 增加集群节点时新建pool
  - 为该pool新建CRUSH规则，使得该pool的对象只分布在新增的节点上
  - 上传文件时进行pool调度，选择空间占用较少的pool，将文件写到选择的pool
  - 由于已有的pool的集群节点不变，不会产生数据迁移

GIFT V3.5特点

- 良好的扩展性
  - 接入层无状态，支持通过负载均衡扩展
  - 元数据存储层通过分库分表方式支持横向扩展
  - 数据存储层通过无中心的设计和分集群方式支持横向扩展
- 良好的服务可用性
  - 接入层、元数据存储层、数据存储层都无单点失效问题
- 良好的数据可靠性
  - 数据多副本存储，可指定副本放置规则
  - 数据自恢复

谢谢！