CoolCeph系统设计

袁冬 2017年11月

# 1 系统概述

CoolCeph是一个参考Ceph RADOS实现的分布式存储系统，旨在提供一种基于Object语义的通用的后端存储类型，以便在之上构建NAS、Block、S3等不同类型的业务存储系统。

CoolCeph主要针对Ceph做以下改进：

* 避免单点故障而导致的超时等待，IO在设计范围内不存在Hang
* 严格分离RADOS和具体存储协议（CephFS、RBD、RGW）的实现
* 支持ROW形式的对象级快照

系统的关键设计选型包括：

* 采用MultiPAXOS作为数据一致性协议
* 数据对象通过一致性哈直接映射到存储服务器

## 1.1 数据模型

### 1.1.1 数据模型

CoolCeph以对象为数据的基本单位，每一个对象包含以下内容

* 对象ID（oid）：采用字符串表示，用于表示一个对象。
* 对象内容（data）：二进制内容，支持随机读写
* 对象属性（attr）：对象的键值对集合，类似文件的扩展属性

每一个对象的ID必须是唯一的，对象和对象之间没有任何关系。该数据模型可以理解为下图：



系统后续将支持存储池，不同存储池，除了共享底层物理存储空间，其数据是完全对立的。

### 1.1.2 访问接口

系统支持对于对象创建、删除、读写、KV等操作，还支持对象级事务，即对于单个对象的多次修改（包括对内容和KV的修改），可以通过一次事务完成。

具体来说，系统对外提供的访问接口包括：

（1）对象操作接口

* 创建对象：touch(char\* oid)
* 删除对象：delete(char\* oid)

（2）对象内容接口

* 对象内容写：write(char\* oid, int64 offset, int64 length, char\* data)
* 对象内容读：read(char\* oid, int64 offset, int64 length, char\* buffer)

（3）对象attr接口

* 添加/更新attr：put\_attr(char\* oid, char\* key, char\* value)
* 读取attr：get\_attr(char\* oid, char\* key, char\* buffer)

（4）对象事务接口

对于单个对象的多个修改，可以通过一个事务完成，代码示例如下：

1. transaction t = begin\_new\_trasaction(oid)
2. t.write(offset, length, data)
3. t.put\_atrr(key, value)
4. submit\_transaction(t);

## 1.2 设计目标

### 1.2.1 性能

系统面向的场景主要是大量随机IO的场景，因此性能采用4K随机写作为主要衡量指标。

系统在预期硬件（SSD+万兆网络）环境下的预期性能为：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 50分位 | 99分位 | 最大延迟 |
| 正常场景 | 1ms | 3ms | 10ms |
| 单点故障场景 | 1ms | 5ms | 50ms |
| 多点故障场景 | 10ms | 100ms | 1s |

表格内为再次读/再次写性能，单位是毫秒，理论上初次读写性能与此相同。通常读性能要优于写性能，但考虑到缓存影响，并不将读性能作为衡量指标，但要求读性能不能低于写性能。

以上指标中，平均延迟和99分位延迟为核心指标。前者表示系统的直观性能，后者则表示系统最差情况下的表现。对于整个系统的性能监控和统计中，采样周期不大于1分钟，理想情况是10s。

系统的核心设计目标是：**所有IO请求不等待任何可用性设计范围内的超时。**

即假设数据采用三副本的容灾设计，可用性为允许一个副本宕机，则对于任何单点故障，请求的延迟（Latency）不受影响。

对于吞吐量来说，通过理解为对于大于1MB以上的请求，可以达到硬件（主要是网络带宽）的极限。

### 1.2.2 一致性

系统对于对象的读写，提供强一致性保证。

即对于对象的读写操作，假设数据x的当前版本是v0，则系统保证以下读写一致性：

1. 对于修改请求v1，如果系统已经给出ack，则系统确保后续所有对x的读操作返回的至少是v1之后的版本。
2. 对于修改请求v1，如果系统没有给出ack，则此时有可能读到v0版本，也可能读到v1版本。但是系统保证，如果返回了v1版本，则后续所有请求都返回至少v1版本，即不会存在脏读。

### 1.2.3 可靠性

提供9个9的可靠性，基于单个物理机房、三副本、多Tor数据分布。

数据丢失的定义：即一年内存在任意一个对象丢失的概率。

### 1.2.4 可用性

提供5个9（99.999%）的可用性。

可用性采用时间作为单位来定义，即 可用性 = 可用时间/总时间。

通常总时间采用一个月为单位，可用时间的颗粒度理想状态下采用1s为单位。出于统计的考虑，也可以采用10s或者1分钟作为单位。

则单位时间可用的标准为：

* 所有系统收到的请求都成功
* 所有系统收到的请求延迟都满足标准（按照正常状态计算）

以上两条中任意一条不满足，则该时间段视为不可用，即可用时间减少了1s。

则，按照5个9的可用性，1个月内的不可用性时间为25秒，1年的不可用时间为300秒。

## 1.3 系统限制

### 1.3.1 集群规模

单个集群支持1000台物理机，单台物理机支持两种目标型号：

* SSD型：12 \* 8TB，集群物理容量96PB
* SATA型：36 \* 8TB，集群物理容量288PB

支持集群的平滑扩容，初始集群大小为9台。磁盘利用率大于90%。

### 1.3.2 环境要求

目标软硬件环境为

|  |  |  |
| --- | --- | --- |
|  | 配置 | 备注 |
| CPU | Intel Xeon 2630V3+ | SSD机型需要较好的CPU |
| 内存 | 64GB+ | 更大的内存将有利用缓存 |
| 存储 | SSD支持 12 \* 8TB  SATA支持 36 \* 8TB | 视存储引擎的情况支持NVME和SDPK |
| 网络 | 10Gb \* 2 或 25Gb \*2 | 支持RDMA网络 |
| 其它 | Intel Optane | 视存储引擎支持情况而定 |
| OS | CentOS 7 x64 | 系统实现为用户态 |