2021

存储规划

基础架构存储团队的总结与规划

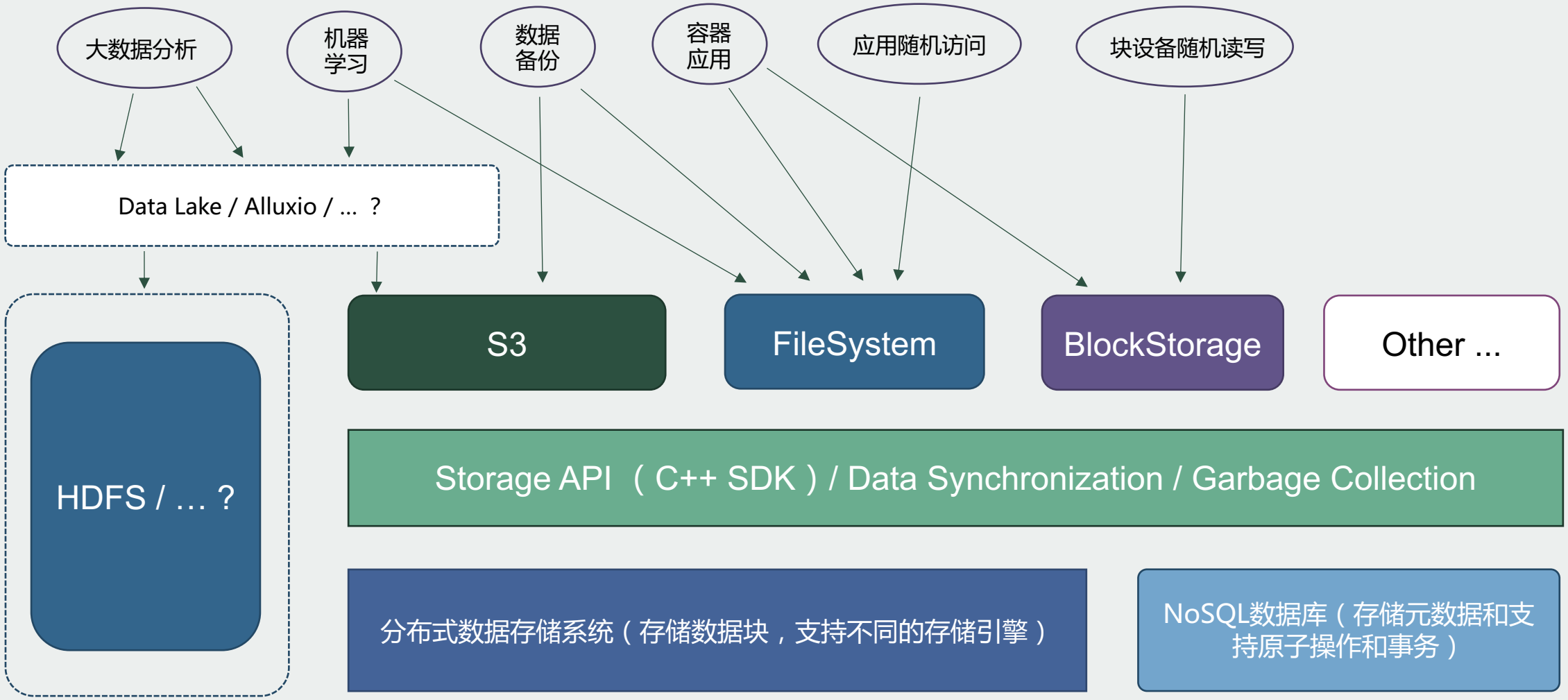
陈磊

2020/11/8

目录

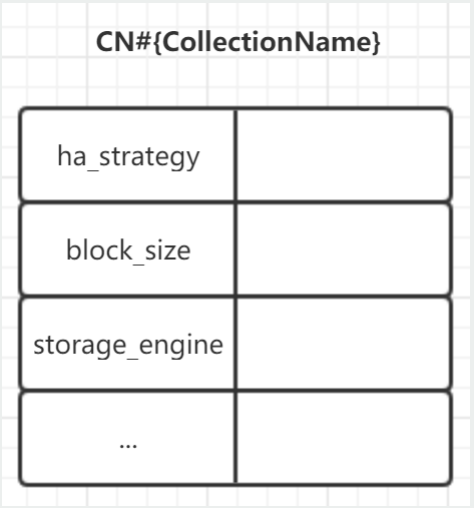
CONTENTS

存储技术栈 - 简介


大数据分析
机器学习
数据备份
容器应用
应用随机访问
块设备随机读写
Data Lake / Alluxio / ... ?
HDFS / ... ?
S3
FileSystem
BlockStorage
Other ...
Storage API （C++ SDK）/ Data Synchronization / Garbage Collection
分布式数据存储系统（存储数据块，支持不同的存储引擎）
NoSQL数据库（存储元数据和支持原子操作和事务）

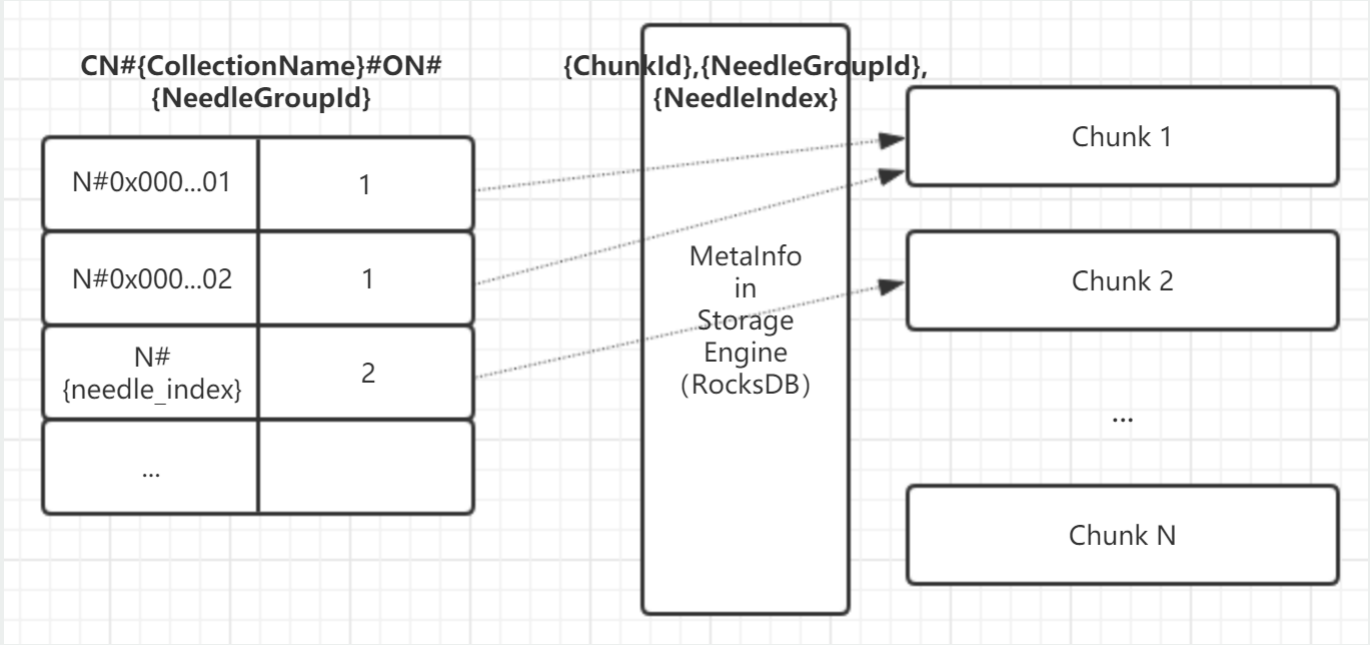
Storage API (Collection)

- 含义：数据集
- 属性
  1. 高可靠方式：Replication or EC
  2. 数据切分的大小
  3. 是否支持随机读写选择不同的存储引擎
- API
  1. Create
  2. Update
  3. Delete
  4. AllocateNeedleGroup
- 对应的元数据见右图



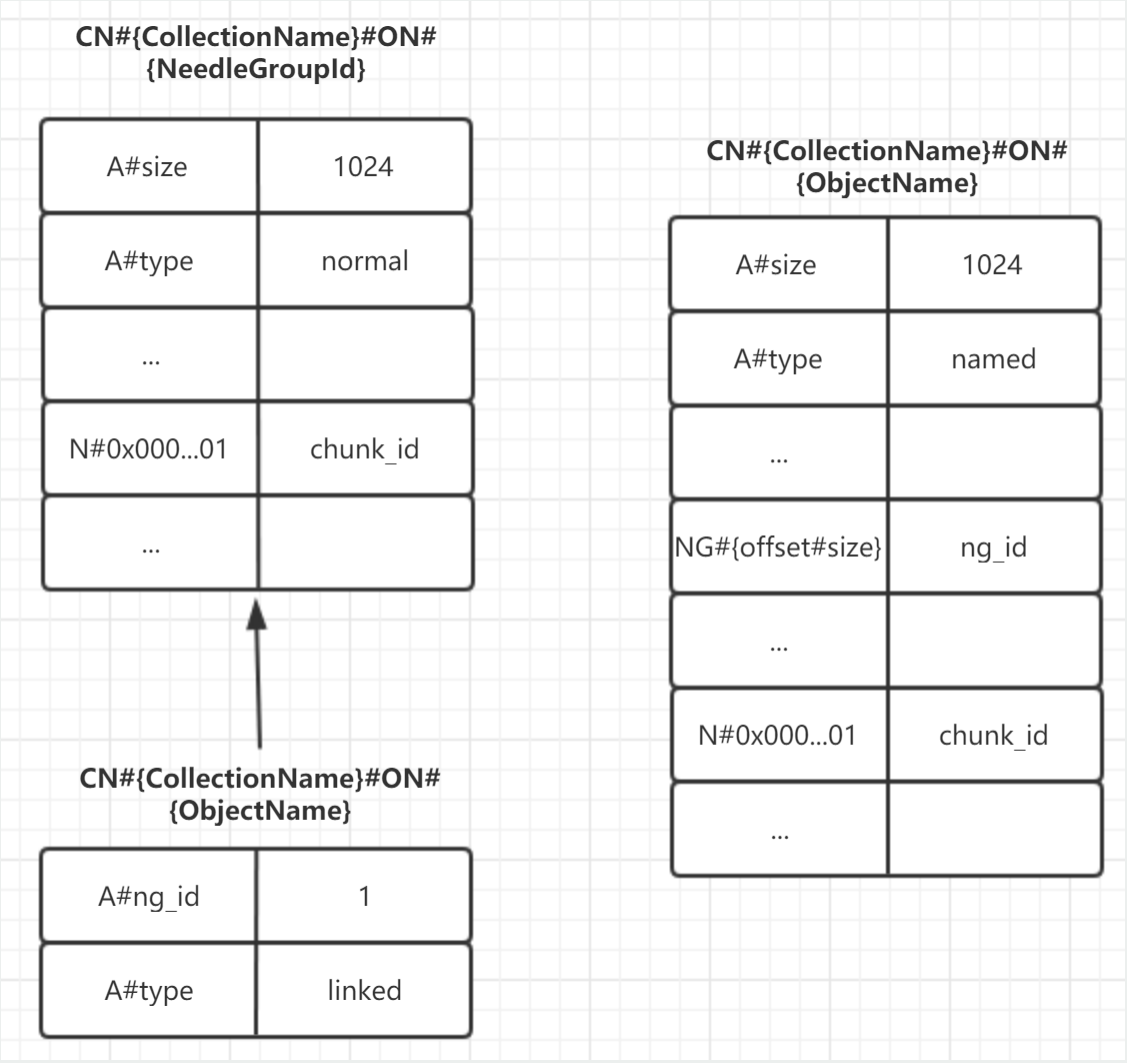
Storage API（NeedleGroup)

- 含义：数据片组，具有唯一的Id
- API
  1. GetId：获取Collection内唯一Id
  2. Read
  3. Write
  4. Truncate
- 对应的元数据和数据见右图



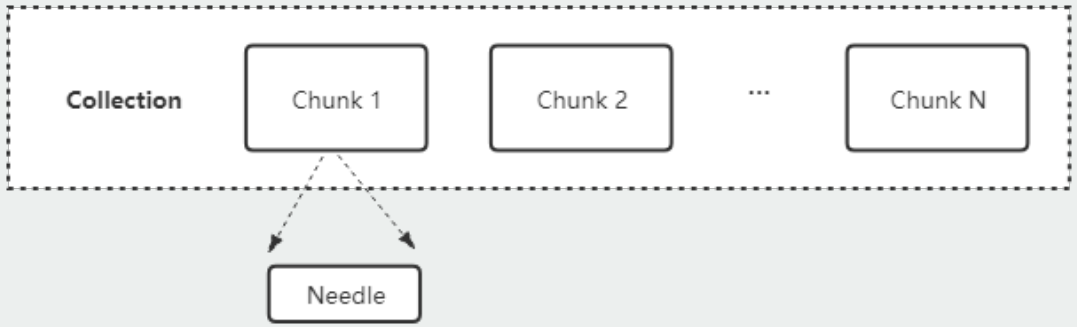
Storage API - Object

- 含义：对象 = Name + Attributes + NeedleGroup
- API
  1. Create/Open/Delete
  2. Open
  3. Read/Write/Truncate
- Object 类型：
  1. normal
  2. named
  3. linked
- 元数据存储为NoSQL，结构见右图

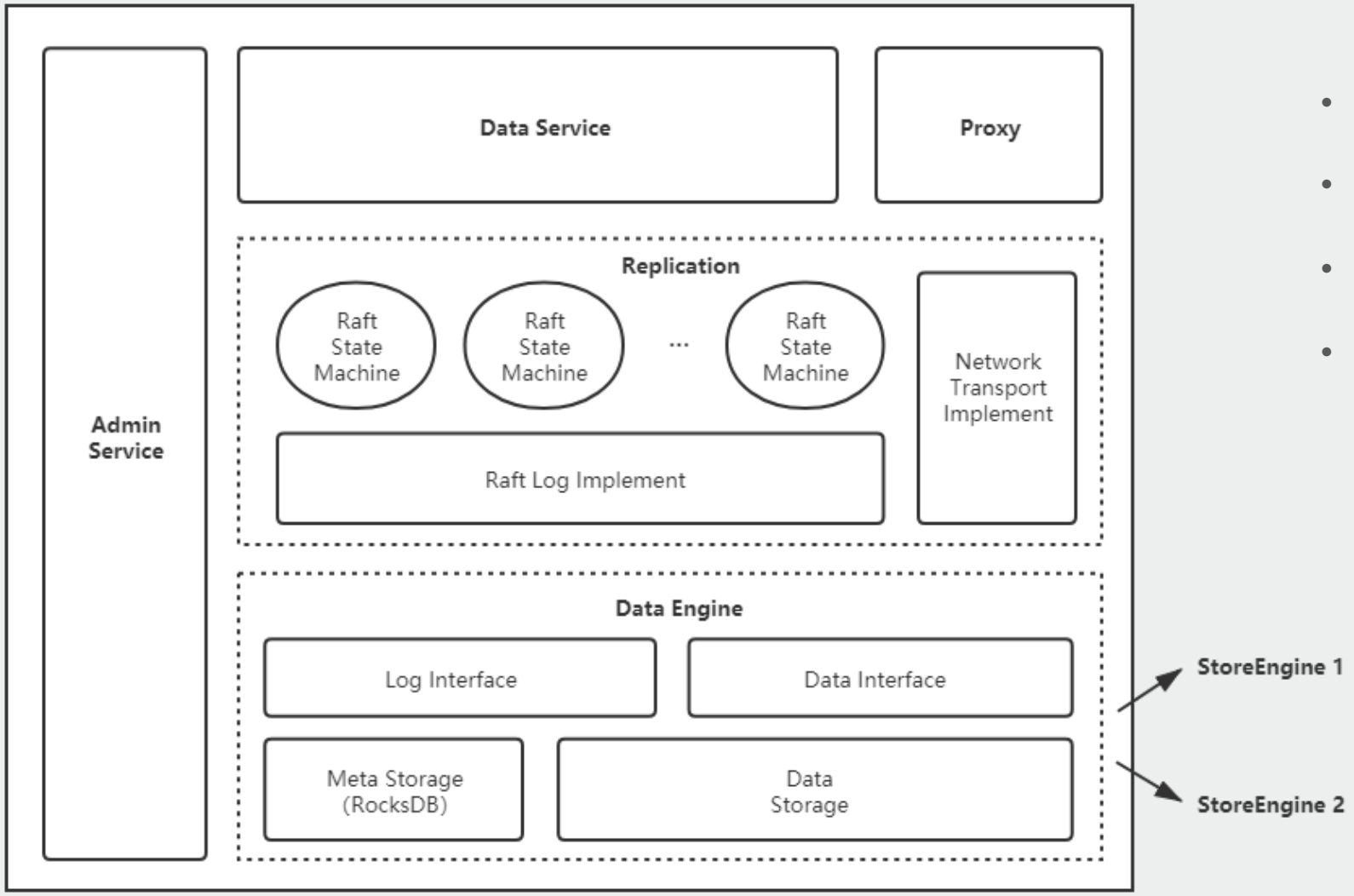


数据存储 - 逻辑概念

- Collection：数据池，一个集群可创建多个Collections
- Chunk：一个Collection可以包含多个Chunks，数目根据使用情况可增加；作为集群管理的基本单位，分布于不同的集群数据节点上，是副本的基本单位
- Needle：用户读写数据的基本单位，存在key和value；key有两部分做成 {chunk id},{logical key}；通过key在存储引擎的索引中找到needle的offset和length，然后访问



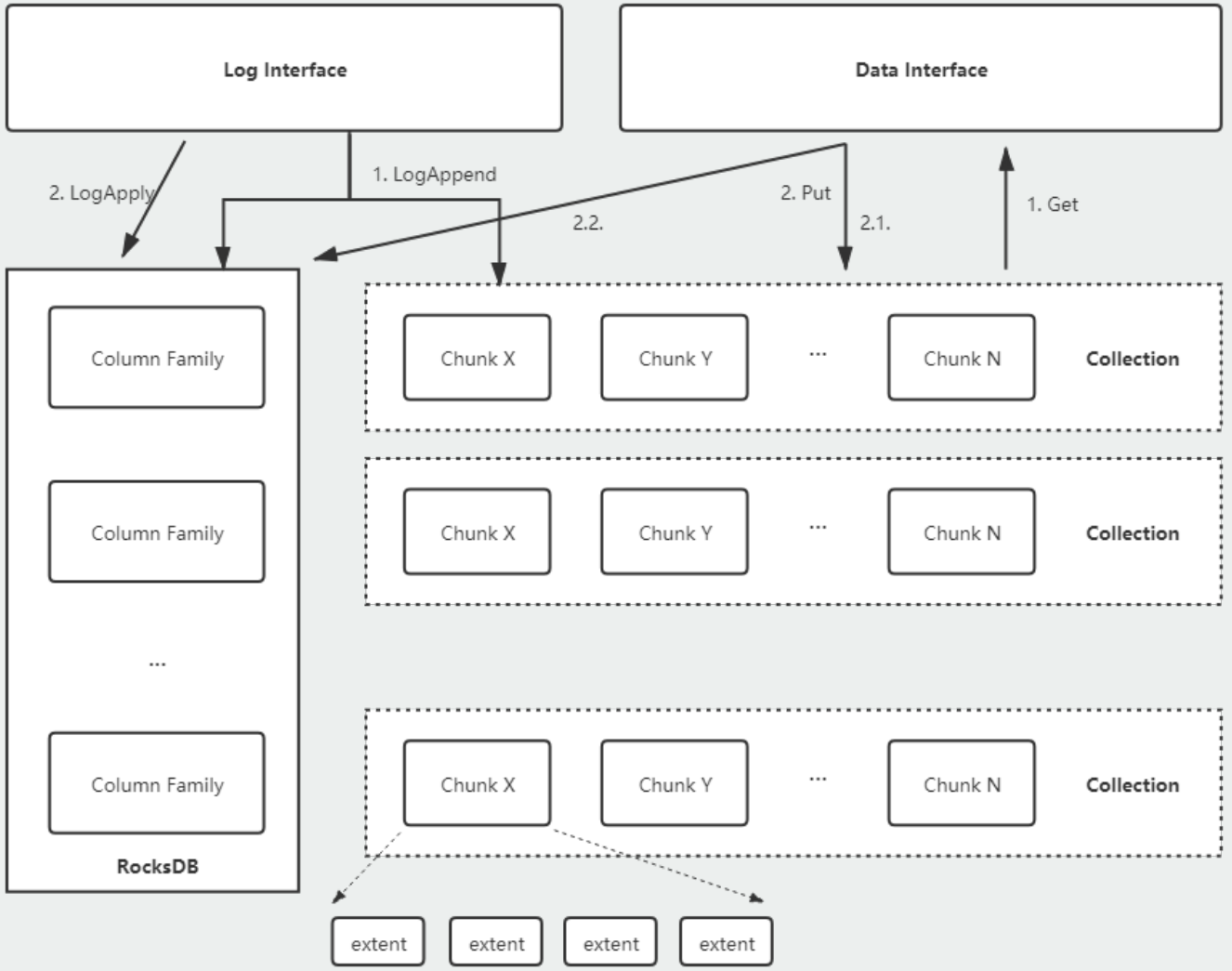
数据存储 – 数据节点功能模块



- Replication采用Raft（Parallel Raft）
- 日志接口：LogAppend/LogApply/...
- 数据接口：Get/Put/...
- 支持多种存储引擎，面向不同的存储场景

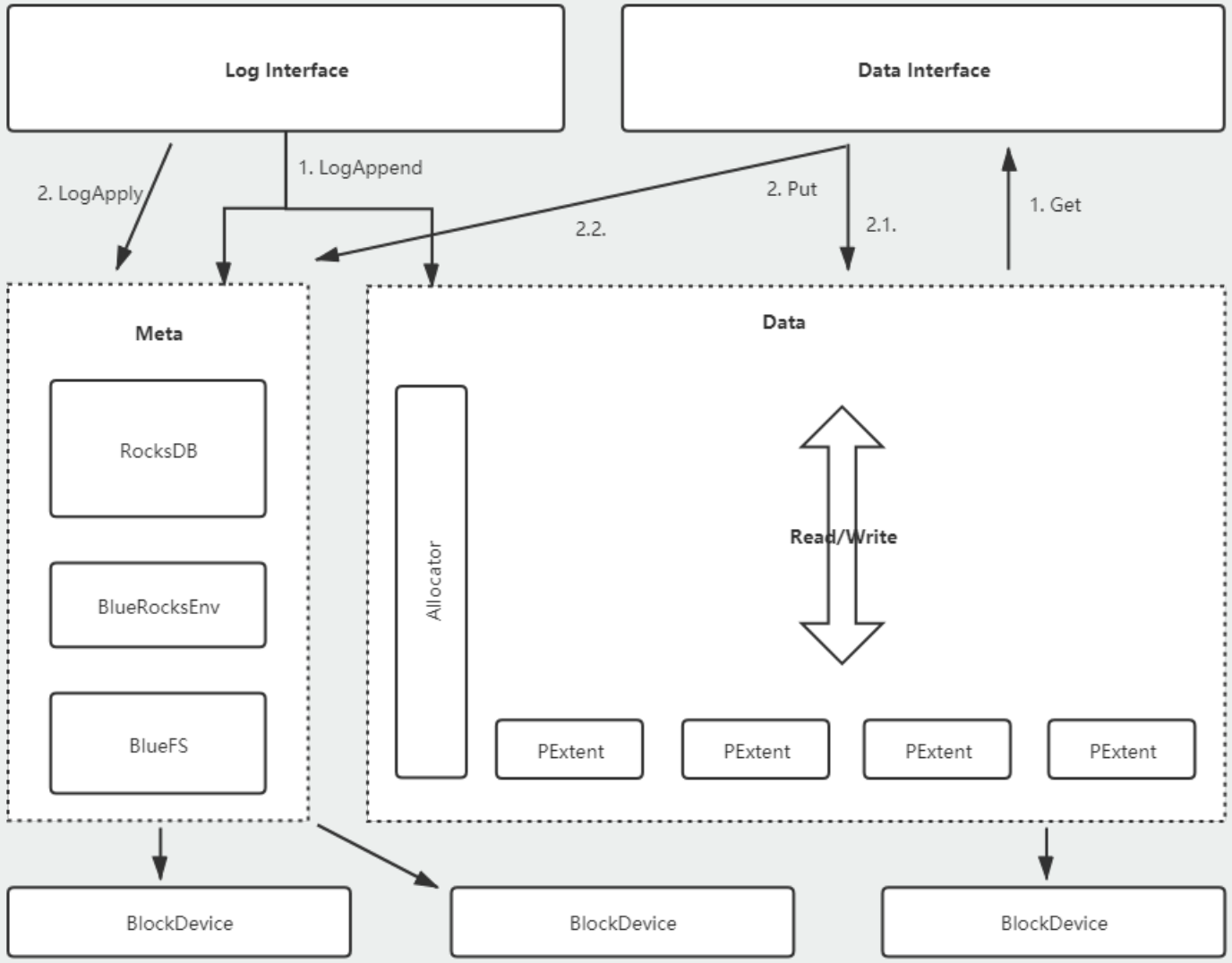
数据存储 - 日志型

- 日志即数据
- 每个节点下的每个Collection的所有Chunk组成一个集合，共享同一个Column Family
- 小于某一个阀值的Needle数据可直接存在RocksDB中
- Chunk内部可优化为多个extents，extent集群不可见

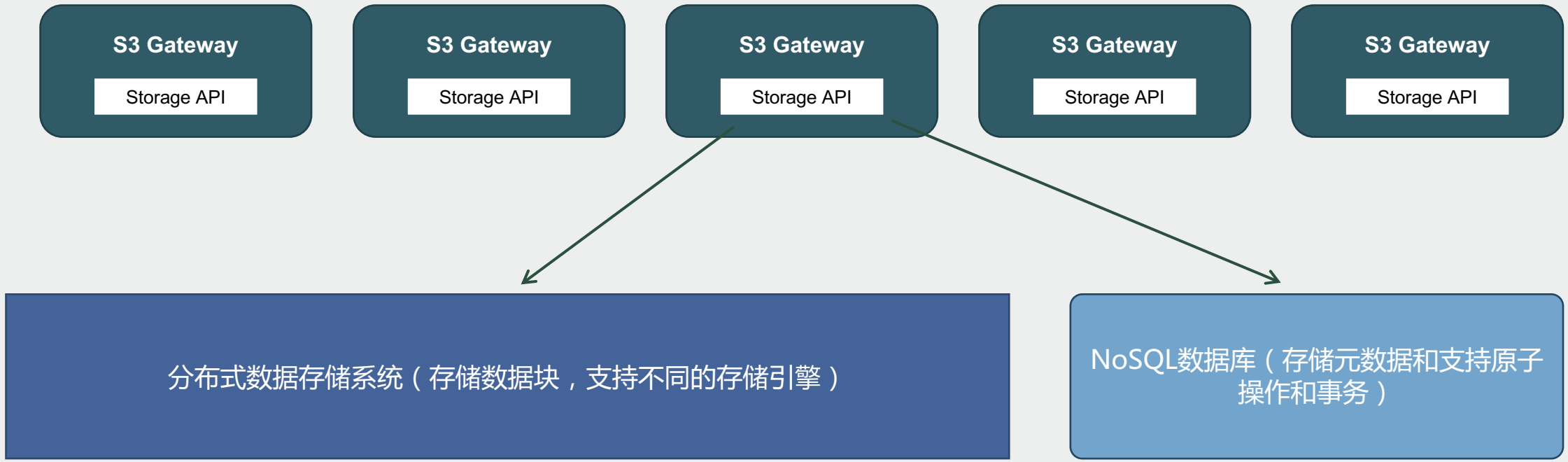


数据存储 - 随机读写型

- Meta是在块设备上构建BlueFS用来支持RocksDB sst文件的读写
- Data是基于Bitmap构建的Allocator，按照物理的PExtent来管理整个块设备
- WAL/Meta/Data可以存不同的设备
- 对用户来说，提供跟日志型存储引擎相同的概念和API



S3


S3 Gateway
Storage API
S3 Gateway
Storage API
S3 Gateway
Storage API
S3 Gateway
Storage API
S3 Gateway
Storage API
分布式数据存储系统（存储数据块，支持不同的存储引擎）
NoSQL数据库（存储元数据和支持原子操作和事务）

S3

- Bucket -> Collection（Storage API） -> Collection（数据存储）
- 普通对象存储使用NamedObject，获取元信息交互次数少，可采取hscanrange获取object全部属性及部分或者全部block索引信息
- NamedObject不支持追加和修改，需要追加和修改的对象采用NormalObject+LinkedObject
- 需要去重的对象采用NormalObject+LinkedObject，RefCount存储在NormalObject的属性中
- 需要与FileSystem打通的对象采用NormalObject+LinkedObject，ObjectName可任意修改

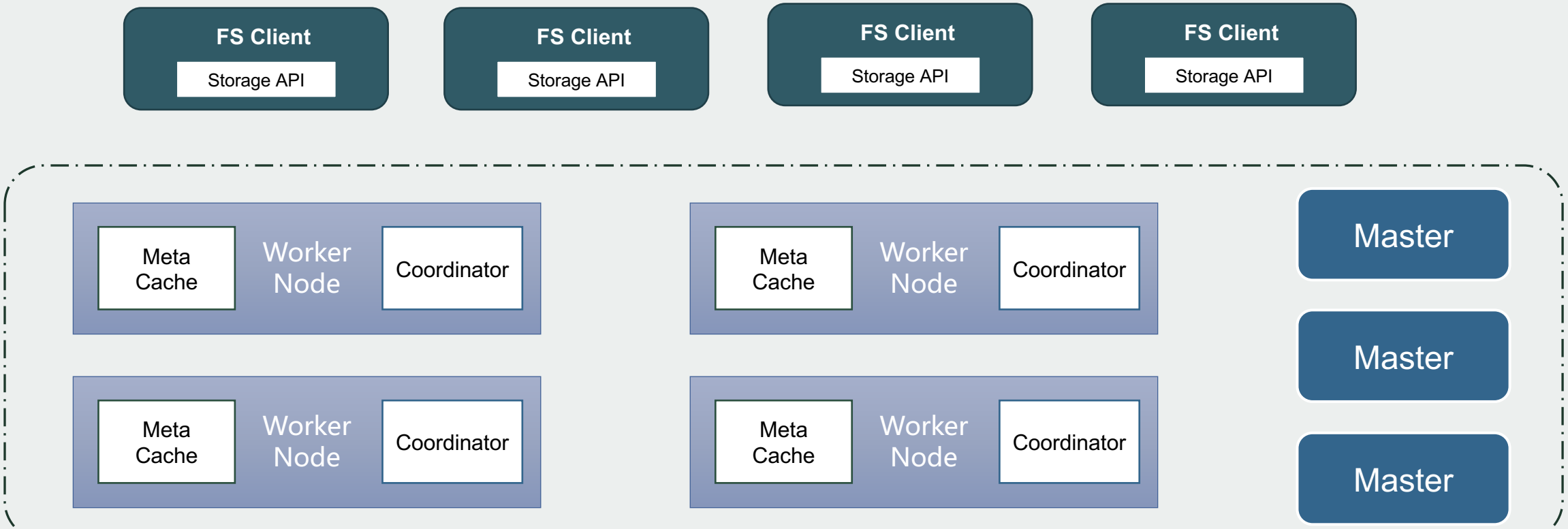
对象下载：

```
Status s = Object::Open(obj_name, &obj);
if (s.Ok()) {
  obj->Read(offset, len, buf);
}
obj->Close();
```

对象上传：

```
NeedleGroup* ng = Collection::AllocateNeedleGroup();
ng->Write(0/*offset*/, data, data_len);
ngs.push(ng);
obj = Object::CreateNamedObj(obj_name, attributes, ngs);
// obj = Object::CreateLinkedObj(obj_name, attributes, ng->GetId();
if (obj->NeedGc()) {
  obj->DoGc();
}
obj->Close();
```

Filesystem



分布式数据存储系统（存储数据块，支持不同的存储引擎）

NoSQL数据库（存储元数据和支持原子操作和事务）

Filesystem

- FileSystem -> Collection（Storage API） -> Collection（数据存储）
- 文件内容索引+目录树索引存储到NoSQL数据库中
- 文件内容索引和文件内容可以通过Storage API实现，每个文件存储为一个NormalObject，其NeedleGroup Id作为Inode ID
- Worker Node可包含两大模块元信息模块+协调模块，元信息节点缓存目录树索引和文件内容索引
- 协调模块用于处理原子操作（如：O_APPEND、write的原子性）和文件锁

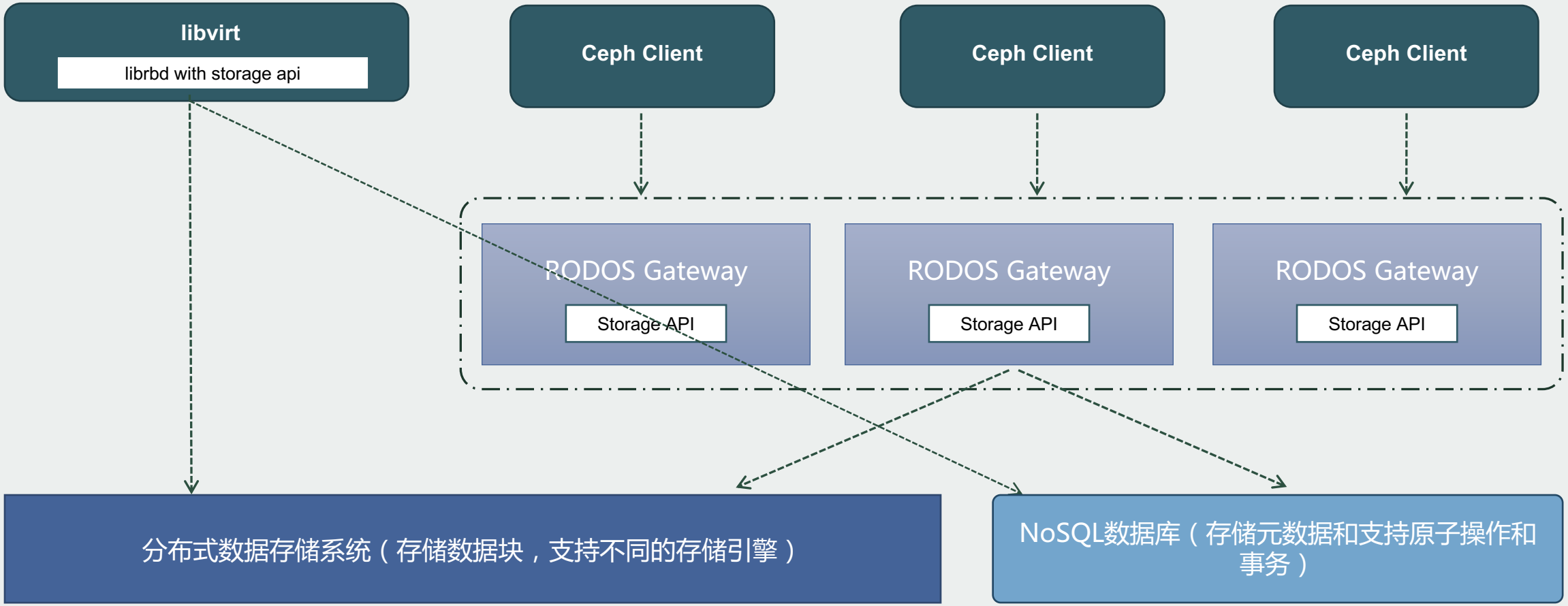
文件读：

```
Status s = Object::Open(inode, &obj);
if (s.Ok()) {
  obj->Read(offset, len, buf);
}
obj->Close();
```

文件更新写：

```
Status s = Object::Open(inode, &obj);
if (s.Ok()) {
  obj->Write(offset, buf, buf_len);
}
obj->Close();
```

BlockStorage



BlockStorage

- Volume -> Collection（Storage API） -> Collection（数据存储）
- 块设备相当于一个可以随机读写的大文件
- 全部功能已经在FS中得到全部支持
- 相比FS只有一个客户端在访问，更简单
- 需要兼容Ceph访问协议，即可做到无缝切换

随机读：

```
Status s = Object::Open(inode, &obj);
if (s.Ok()) {
  obj->Read(offset, len, buf);
}
obj->Close();
```

创建卷：

```
NeedleGroup* ng = Collection::AllocateNeedleGroup();
ng->Truncate(volume_size);
Status s = Object::CreateNormalObj(attributes, ng);
if (!s.Ok()) {
  return s;
}
obj->Close();
```

高级特性

- 对象、文件互通
- 对象、文件、块混合存储
- 冷热分层，动态迁移
- 多集群实时同步（Mirror）
- EC
- QoS
- FS可以对接外部存储服务

谢谢聆听