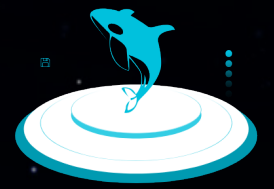
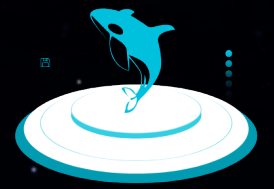


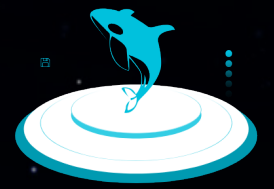
CHTC
中国HBase技术社区

MOB HBase2.0重新定义小对象实时存取

孟庆义 阿里巴巴



目录



CHTC
中国HBase技术社区

01

背景介绍

MOB的使用场景与价值

HBase2.0加持MOB技术，支持小对象实时存取，具有读写强一致，低延迟，高并发等特点，并兼容所有企业级特性如Snapshot，Replication

Medium Objects 中等大小对象

MOB问题背景

IO放大

多副本
WAL
Flush
Compaction

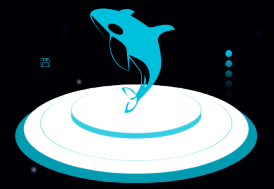
资源限制

ECS 16核64G 5Gbps
高效云盘140MBps
ECS15个挂载点

写入瓶颈

Compaction落后导致**文件数上升**，进而导致flush**延迟**，进而导致内存瓶颈，最终阻塞写入；同时文件数过多导致查询慢

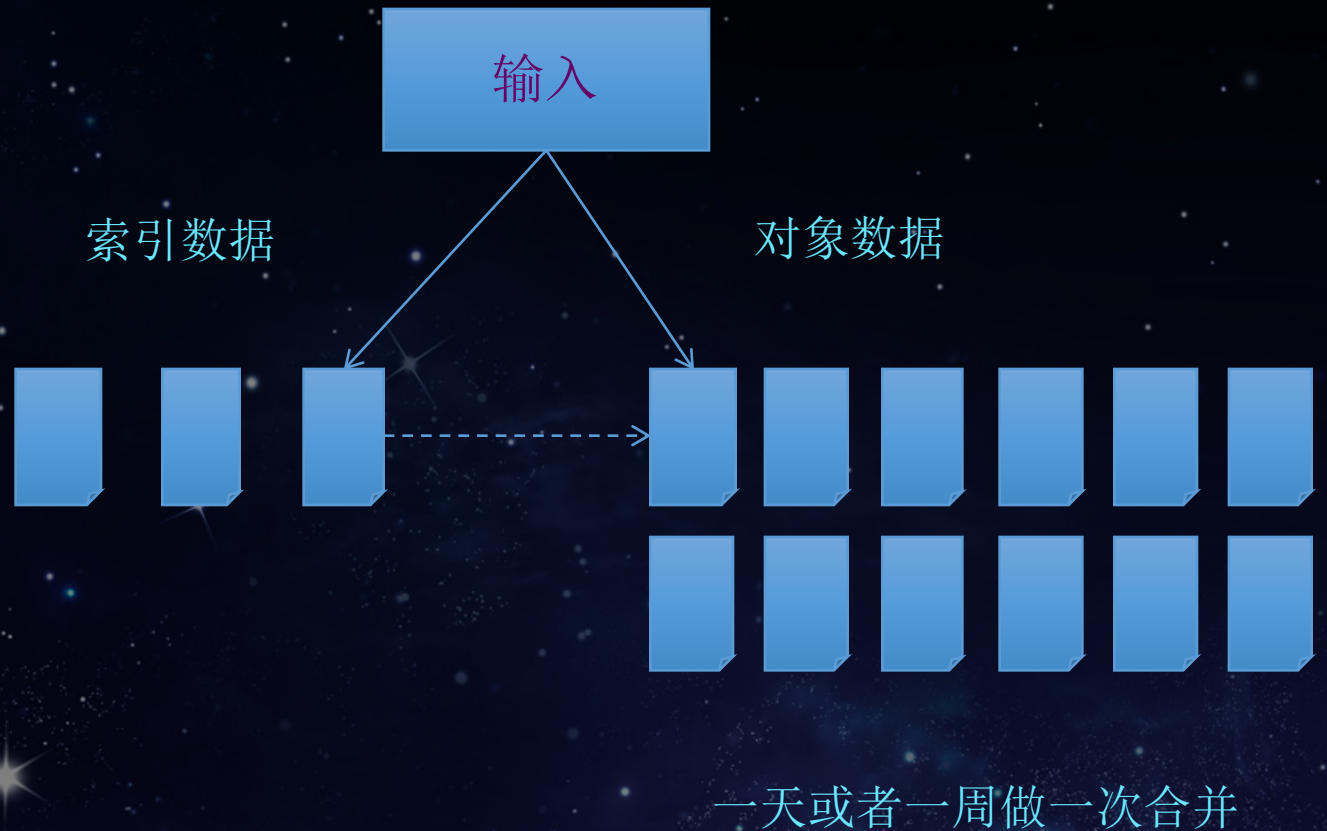
假设网络IO放大系数=5，则实际可用带宽为5Gbps / 8 / 5 = 128MB/s，对于1MB对象单机能提供的TPS=128



02

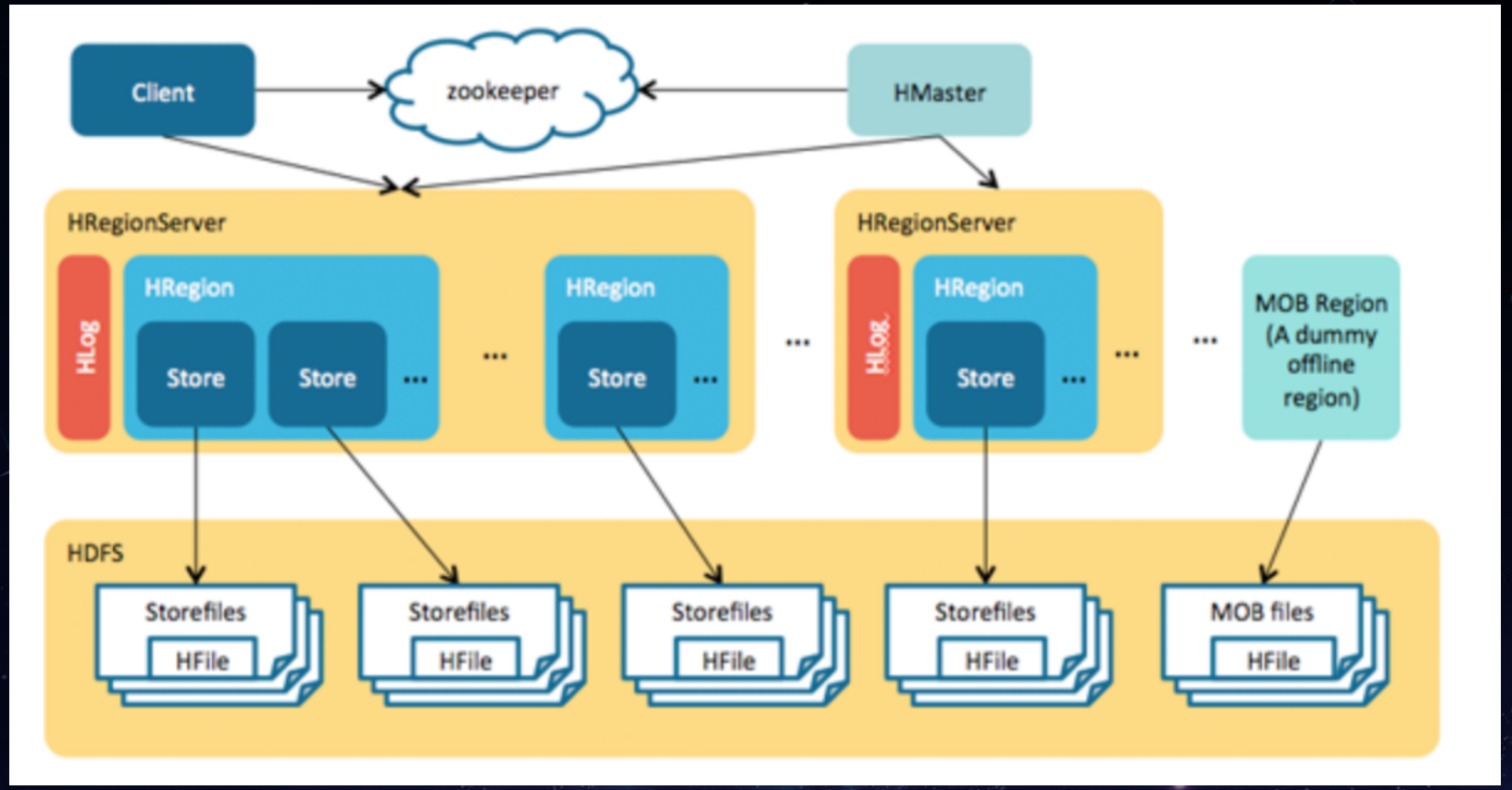
MOB原理与背景

解决思路：降低Compaction的频率

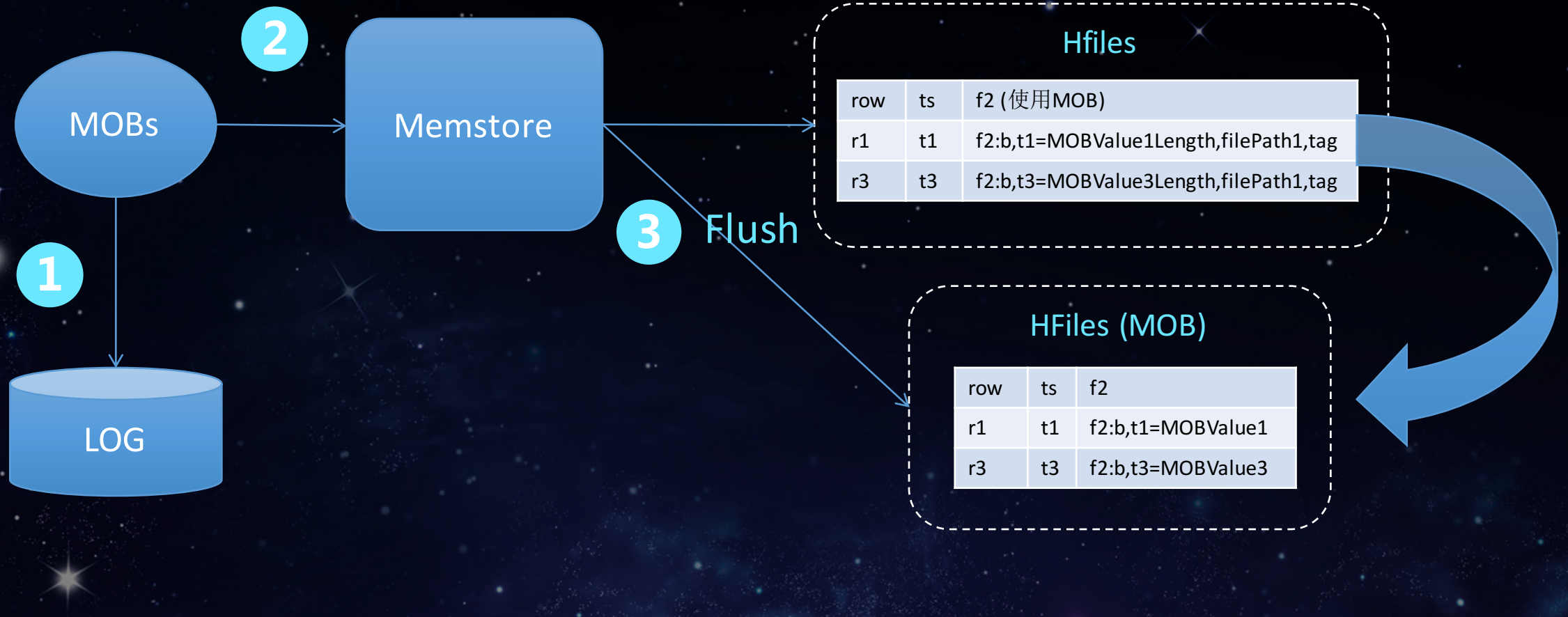


一天或者一周做一次合并

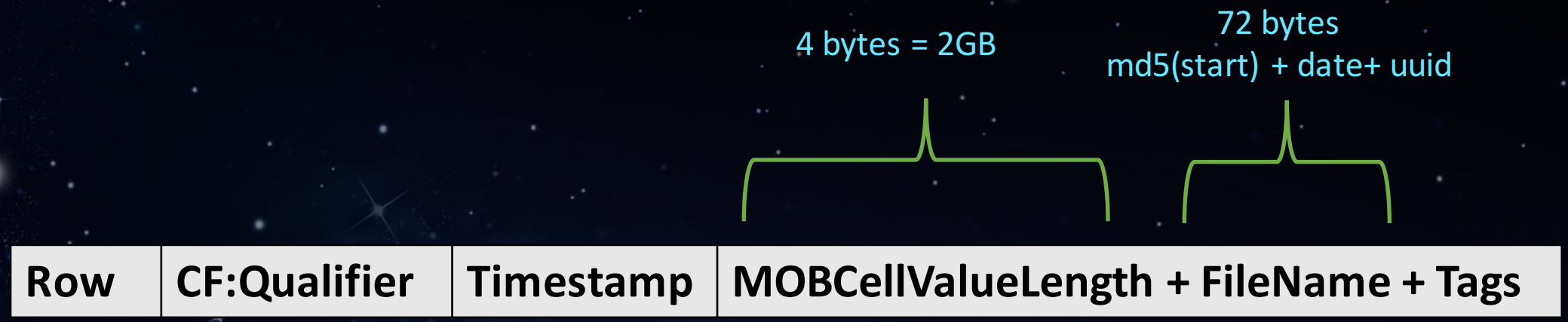
系统架构


Client
zookeeper
HMaster
HRegionServer
HLog
HRegion
Store
Store
HRegion
Store
HRegionServer
HLog
HRegion
Store
MOB Region (A dummy offline region)
HDFS
Storefiles
HFile
Storefiles
HFile
Storefiles
HFile
Storefiles
HFile
MOB files
HFile

写入路径

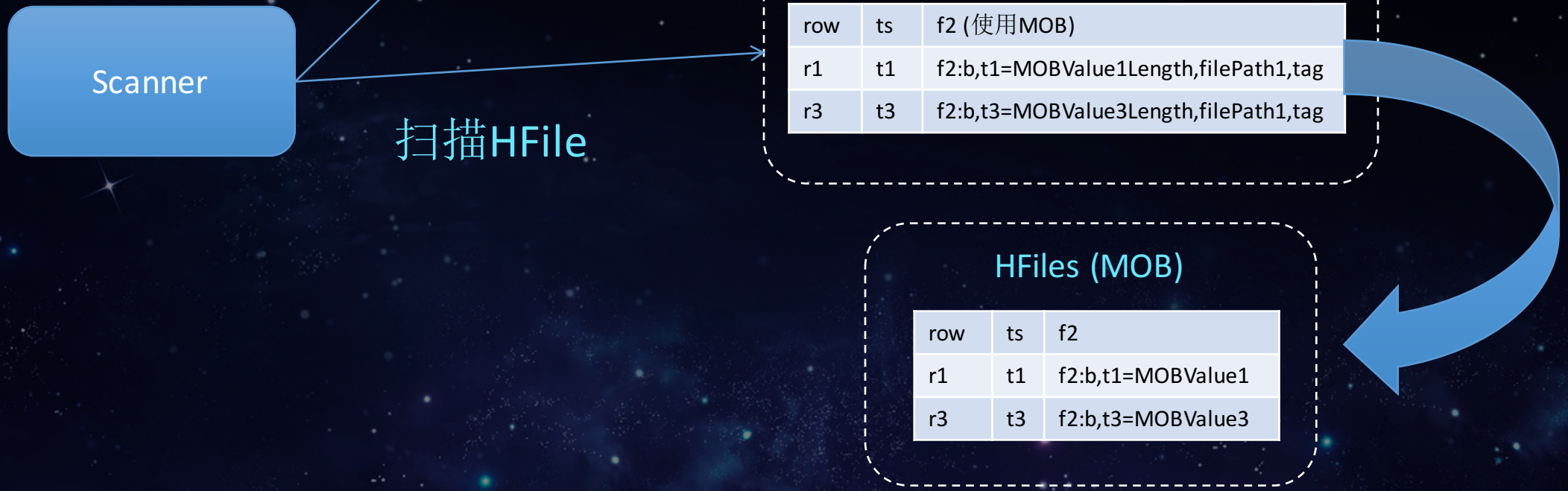

2
MOBs
Memstore
3
Flush
1
LOG
Hfiles
row
ts
f2 (使用MOB)
r1
t1
f2:b,t1=MOBValue1Length,filePath1,tag
r3
t3
f2:b,t3=MOBValue3Length,filePath1,tag
HFiles (MOB)
row
ts
f2
r1
t1
f2:b,t1=MOBValue1
r3
t3
f2:b,t3=MOBValue3

引用结构



默认MOB阈值=100KB

读取路径


Memstore
row ts f2
r1 t1 f2:b,t1=MOBValue1
r3 t3 f2:b,t3=MOBValue3
扫描Memstore
Scanner
扫描HFile
HFiles
row ts f2 (使用MOB)
r1 t1 f2:b,t1=MOBValue1Length,filePath1,tag
r3 t3 f2:b,t3=MOBValue3Length,filePath1,tag
HFiles (MOB)
row ts f2
r1 t1 f2:b,t1=MOBValue1
r3 t3 f2:b,t3=MOBValue3

合并索引文件

row	ts	f2 (使用MOB)
r1	t1	f2:b,t1=MOBValue1Length,filePath1,tag
r3	t3	f2:b,t3=MOBValue3Length,filePath1,tag

row	ts	f2 (使用MOB)
r2	t2	f2:b,t2=MOBValue2Length,filePath2,tag
r4	t4	f2:b,t5=MOBValue4Length,filePath2,tag

Compaction Index

row	ts	f2 (使用MOB)
r1	t1	f2:b,t1=MOBValue1Length,filePath1,tag
r2	t2	f2:b,t2=MOBValue2Length,filePath2,tag
r3	t3	f2:b,t3=MOBValue3Length,filePath1,tag
r4	t4	f2:b,t2=MOBValue4Length,filePath2,tag

HFiles (MOB)

row	ts	f2
r1	t1	f2:b,t1=MOBValue1
r3	t3	f2:b,t3=MOBValue3

row	ts	f2
r2	t2	f2:b,t2=MOBValue2
r4	t4	f2:b,t4=MOBValue4

CHTC
中国HBase技术社区

合并MOB文件

row	ts	f2 (使用MOB)
r1	t1	…filePath1
r2	t2	…filePath2
r3	t3	…filePath1
r4	t4	… filePath2

引用

HFiles (MOB)

row	ts	f2
r1	t1	f2:b,t1=MOBValue1
r3	t3	f2:b,t3=MOBValue3

row	ts	f2
r2	t2	f2:b,t2=MOBValue2
r4	t4	f2:b,t4=MOBValue4

HDFS对文件数量的支持存在上限
需要周期性合并小文件

Bulkload

Compaction MOB

row	ts	f2 (使用MOB)
r1	t1	f2:b,t1=MOBValue1Length,filePath3,tag
r2	t2	f2:b,t2=MOBValue2Length,filePath3,tag
r3	t3	f2:b,t3=MOBValue3Length,filePath3,tag
r4	t4	f2:b,t2=MOBValue4Length,filePath3,tag

引用

row	ts	f2
r1	t1	f2:b,t1=MOBValue1
r2	t2	f2:b,t2=MOBValue2
r3	t3	f2:b,t3=MOBValue3
r4	t4	f2:b,t2=MOBValue4

MOB合并策略

分组合并

按照文件所属分区以及日期两个维度进行分区，分区内的文件进行合并

称为：PartitionedMobCompactor

默认每日合并

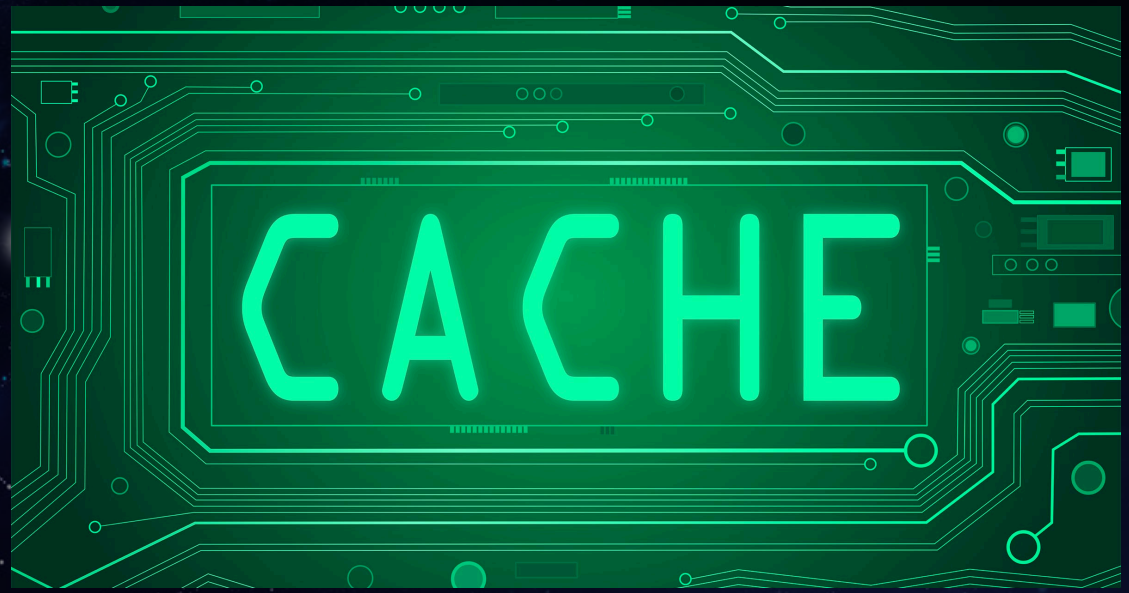
按天分区，小于阈值的文件参与合并，阈值默认为1028MB

周月合并

周合并，按周分区，当前周文件小于阈值的参与，其它周的文件小于7*阈值的参与

月合并，按月分区，当前周不参与，本月周小于7*阈值参与，其它月小于28*阈值的参与

MOB缓存



文件句柄缓存

hbase.mob.file.cache.size = 1000

LRU cache

MOB对象缓存

复用BlockCache

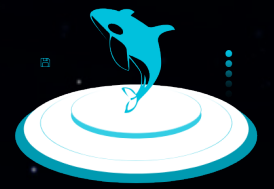
使用方式

Shell访问

```
hbase> create 't1', {NAME => 'f1', IS_MOB => true,
MOB_THRESHOLD => 102400}
hbase> alter ‘t1’, {NAME => ‘f1’, IS_MOB => true,
MOB_THRESHOLD => 102400}
```

Java API

```
HColumnDescriptor hcd = new HColumnDescriptor(“f”);
hcd.setMobEnabled(true);
hcd.setMobThreshold(102400L);
```



CHTC
中国HBase技术社区

03

MOB VS. 传统对象存储

回顾对象存储

模型

KV 结构

一组KV组成的集合成为Bucket

缺少灵活性

API

支持前缀扫描，获取符合条件的Keys

通过Key获取对象

检索能力弱

访问协议

REST

简洁，语言无关

消耗更多的链接和网络带宽

计费模式

按请求次数计费

请求频率越高成本越高

Bug可能引发计费灾难

云HBase与对象存储对比

	对象存储	云HBase
建模能力	KV	KV，表格，稀疏表
查询能力	前缀查询	前缀查询 过滤器 索引
性能	优	优，特别对小对象有更低的延迟；在复杂查询场景下，比对象存储有10倍以上的性能提升
成本	按流量，请求次数计费，适合访问频率低的场景	托管式，在高并发，高吞吐场景有更低的成本
扩展性	优	优
适用对象范围	通用	<10MB

从一条SQL开始

用户表T：包含三个属性S1、S2、S3
属性的大小为50bytes左右，包含一个对象数据Object从100KB~10MB

S1	S2	S3	Object

查询 select Object from T where S1 = xxx and S2 > yyy and S3 < zzz

对象存储方案

设计逻辑表为：S1+S2+S3 => Object，将S1、S2、S3组合成一个key

```
// 对象存储支持前缀检索，按“S1 = xxx and S2 > yyy”查找元数据
List keys = GetBucket(key >= xxx+yyy)

// 用户需要自己写代码实现对S3属性列的条件过滤
filterByS3(keys, zzz)

// 依次读取出Object
for(key : keys) {
  GetObject(key);
  // do something
}
```

优势：

- 读写强一致
- 支持水平扩展

劣势：

- 实时性差，一次请求要查询N次服务器
- 检索能力不足，仅支持key的前缀检索
- 当属性列增多，特别是动态增加的情况下，对象存储很难支持（key会变得非常复杂）

MySQL+对象存储


```
List references = select reference from T where S1 = xxx
and S2 > yyy and S3 < zzz

// 依次读取出Object
for(Key : references){
    GetObject(key);
  //do something
}
```

优势：

- 检索能力强
- 支持存储结构化数据

劣势：

- 实时性差，一次请求要查询N次服务器
- 读写存在不一致问题
- 不支持动态列
- 运维复杂

HBase MOB

CHTC
中国HBase技术社区

Rowkey	普通列	普通列
S1+S2	S3	Object

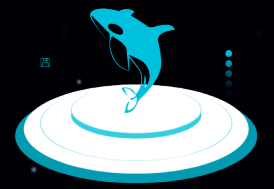
```
Scan scan = new Scan();
scan.setStartKey(S1+S2); // 按S1+S2前缀查询
scan.setFilter(S3); // 利用Filter过滤S3
scan.addColumn(Object);  // 返回Object列
List objects = T.scan(scan); // 一次查询即返回所有结果
```

优势：

- 实时查询，延迟低
- 读写强一致
- 检索能力强
- 支持存储结构化数据
- 支持动态列
- 支持水平扩展
- 易于维护

各方案对比

	对象存储	MySQL+对象存储	HBase MOB
读写强一致	Y	N0	Y
查询能力	弱	强	强
查询响应时间	高	高	低
运维成本	低	高	低
水平扩展	Y	Y	Y

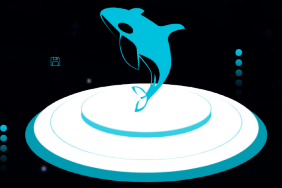


CHTC
中国HBase技术社区

04

总结与展望

HBase2.0重新定义小对象实时存取



HBase2.0重新定义了小对象实时存取的业务访问方式，不再是索引+对象的多次查询，提供简洁的一体化解决方案。具有访问延迟低，读写强一致，检索能力强，水平易扩展等关键能力；并且具备动态列，多版本等灵活的功能；最后一体化的解决方案简化了用户端的代码，也减少了服务端的运维成本。

后续发展

整合OSS

将冷数据归档到OSS，优化成本

提升MOB合并能力

目前由Master执行，考虑改为分布式

独立配置

MOB可以采用不同于主体表的压缩，编码，块大小等配置

CHTC
中国HBase技术社区

THANK YOU

When a cigarette falls in love with a match,it is destined to be hurt.When a cigarette falls in love with a match,it is destined to be hurt.

CHTC
中国HBase技术社区

加入我们；另外，阿里云提供云 HBase 技术支持，欢迎扫描下面二维码。

社区管理员

HBase 技术社区公众号

阿里云 HBase 技术支持

中国 HBase 技术社区网站：http://hbase.group