



## Ali-HBase的SQL实践与改进

17天前

1368

福利：国际顶级盛会HBaseCon Asia 2018将于8月在北京举行，目前正免费开放申请中，更多详情参考

<https://yq.aliyun.com/promotion/631>

如果你对大数据存储、分布式数据库、HBase等感兴趣，欢迎加入我们，一起做最好的大数据在线存储，职位参考及联系方式：[https://maimai.cn/job?](https://maimai.cn/job?webjid=1heZGlyM4&srcu=1aOrffoj1&src=app&fr=my_jobsrecruit_job)

[webjid=1heZGlyM4&srcu=1aOrffoj1&src=app&fr=my\\_jobsrecruit\\_job](https://maimai.cn/job?webjid=1heZGlyM4&srcu=1aOrffoj1&src=app&fr=my_jobsrecruit_job)

摘要：

2017云栖大会Hbase专场，阿里巴巴的天穆带来Ali-HBase的SQL实践与改进的演讲。本文主要从为什么需要SQL开始谈起，进而讲

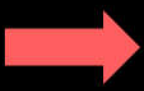


解了SQL on Hbase，接着着重分享了Ali-Hbase SQL的优化与改进，最后对未来进行了展望。

[PPT下载请点击](#)

以下是精彩内容整理：

为什么需要**SQL**？



time (desc)	event	message
100	40011	aaa
80	10020	bbb
80	10010	ccc
70	10040	ddd
60	20000	eee
...	...	...

time (desc)	event	message
120	10010	xxx
110	10020	yyy
100	40011	aaa
80	10020	bbb
80	10010	ccc
...	...	...


云栖社区 yq.aliyun.com

时间序列数据的存取：按照时间顺序追加新记录，按照时间范围查询数据，查询结果按时间倒排。我们数据是按照时间产生的，最新写的数据库一定写在表头，在分布式情况下



所有操作都落在表头，则表头所在的服务器必然会成为写热点。

## Hash散列



time (desc)	event	message
120	10010	xxx
110	10020	yyy
100	40011	aaa
80	10020	bbb
80	10010	ccc
...	...	...

hash	time	event	message
AAAA	100	40011	aaa
BBBB	80	10020	bbb
CCCC	120	10010	xxx
DDDD	80	10010	ccc
EEEE	110	10010	yyy
FFFF			

解决写热点问题就是打散、随机分布，使得任何一行数据都能分布在表的一个随机位置。这带来一个新的问题，数据不再有序，无法按时间进行范围查询。

## 分桶



bucket_id	time(desc)	event	message
1	100	40011	aaa
	70	10040	ddd
	50	30000	fff
2	80	10010	ccc
	60	20000	eee
3	80	10020	bbb
	40	10050	hhh
...	...	...	...

可见，解决写热点和按时间范围查询是一对矛盾的需求。为了同时满足这两个需求，我们需要做一些折中，也就是分桶，通过对原始主键取模，则任何一行数据都可以落在一个随机的“桶”里面，而数据在桶内是有序的，可以按照时间范围来查询。这样，就兼顾了写请求的打散和数据的范围查询需求。其代价是范围查询时必须并发查所有桶，并对结果进行合并。



bucket_id	time (desc)	event	message
1	100	40011	aaa
	70	10040	ddd
	50	30000	fff
2	80	10010	ccc
	60	20000	eee
3	90	10020	bbb
	40	10050	hhh
...	...	...	...

Select \* from eventLog  
where time > 40 and time <= 70;

bucket\_1 : 70,50  
bucket\_2 : 60  
bucket\_3 : NA

merge sort

70, 60, 50

云栖社区 yq.aliyun.com

例如，先查第一个分桶，再查第二个和第三个分桶，得到了70、60、50的结果。因为多机并发查询，分桶方案其实在一定程度上提升了读的性能。

## 基于HBase Native api的实现



- 分桶：
  - 写：打散
  - 读：并发scan, client merge sort
- desc主键：ts = Long.MAX\_VALUE - ts
- rowkey：3列主键的拼接与拆分
- 数据类型转换：Hbase只支持byte[]

对于复杂的业务场景，用户要做的事情更多

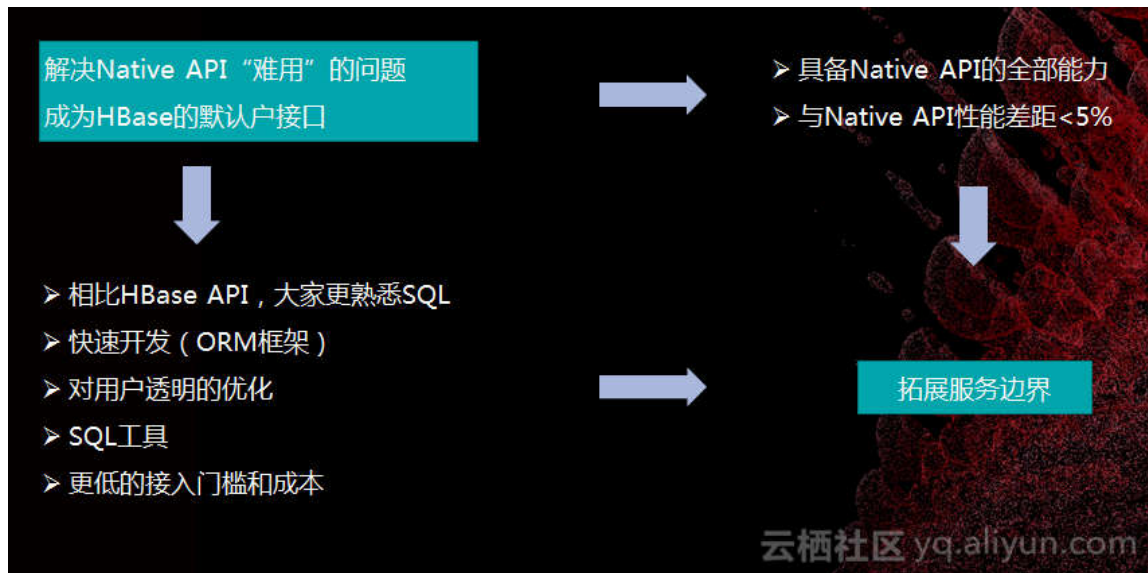
云栖社区 yq.aliyun.com

刚刚说的场景是经过高度抽象的，实际的场景不可能这么简单，即使在简单的场景下就需要做这么多事情。我们HBase API要想用好Hbase就需要很多额外的事情，需要写非常多的代码。学习成本也是很高的，如果想很精准地使用是很需要技巧的，很多东西都要靠经验，用户在利用HBase API的时候要付出很高的学习成本和开发成本。大部分的HBase用户都会遇到这些类似的问题，而且每个用户都需要了解怎么去解决这些问题，



使用了HBase API以后可以对自己的业务做到完全地把控。

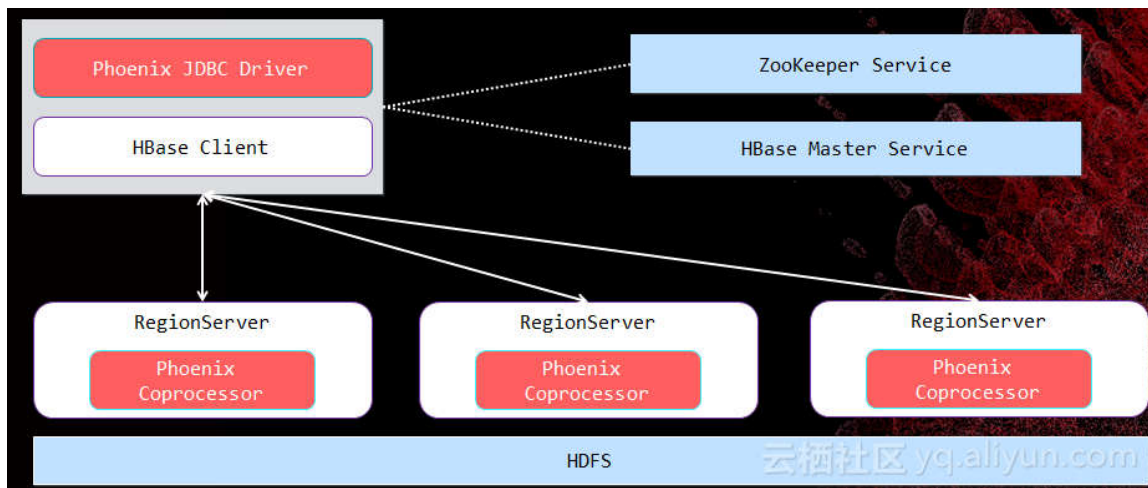
## SQL on HBase



为什么HBase API这么难用，说到底就是太底层了，仅提供了“原语级别的操作”。我们希望能够降低用户的接入门槛，能够低成本低接入Hbase，怎么做这件事情？阿里HBase大部分场景都是相对简单的，并且有共性的，所以我们希望能够引入中间层，来解决这个共性问题。中间层就是SQL，我们希望SQL能够替代API成为HBase的默认户接口。

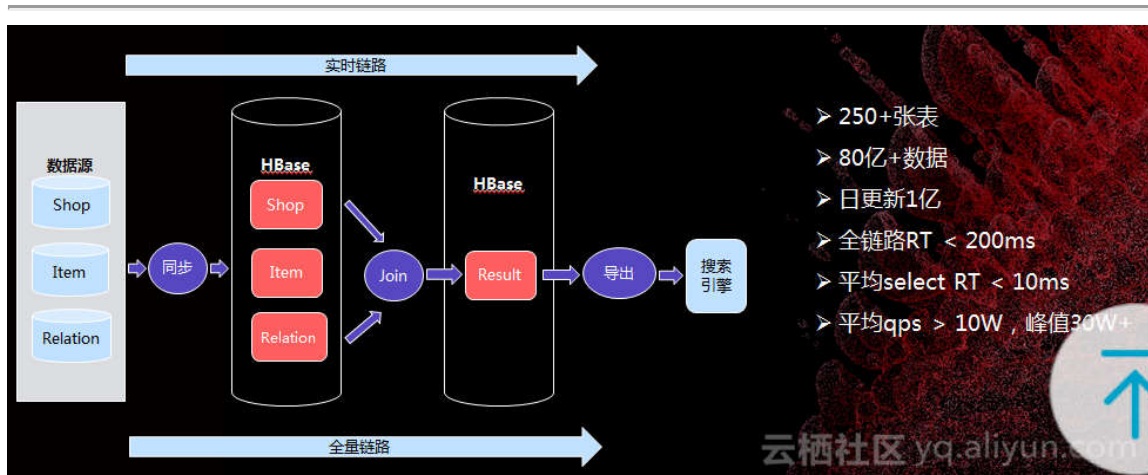






基于Phoenix的SQL ON HBase解决方案，Phoenix就是针对HBase来设计的，而且Phoenix在HBase之间也可以结合得非常好，这也是我们选择Phoenix的一个主要原因。下面看几个具体的场景示例：

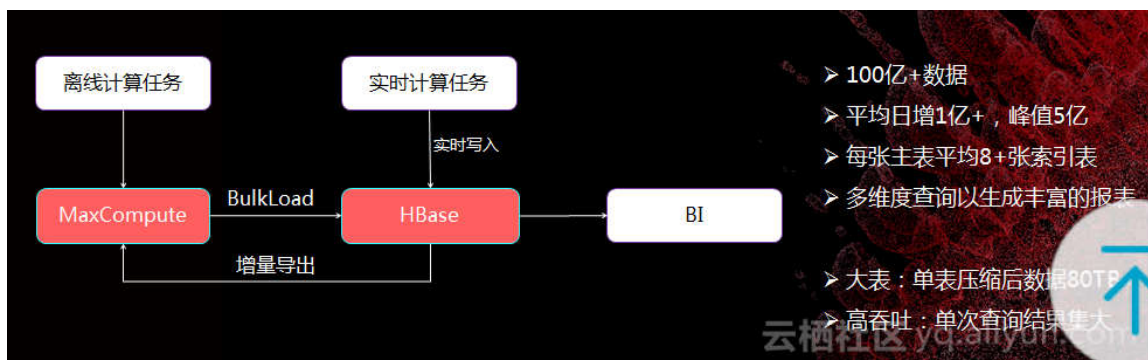
### 支付宝智能搜索dump平台





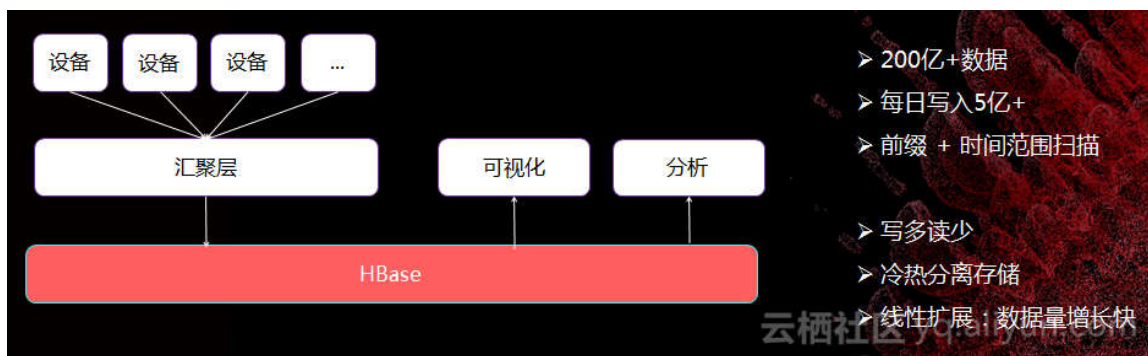
支付宝智能搜索Dump平台，左边的数据源是各种各样的业务数据库，可能是MySQL，可能是HBase的，对业务数据库的变更操作会同步到HBase集群里面很多张维表里面，对维表生成宽表。对于HBase来讲，这个场景除了实时写之外还有很大的全量导入。读是通过很多的全局二级索引，经常变更的索引表。因为搜索的业务，用户的需求经常发生变更，这样对应我们的索引表发生变更，虽然有一些成本，但是相比MySQL来讲，这个变更的成本是可以接受的。因为业务增长比较快，所以线性扩容也是关键点。

## 商品报表



商品报表是另外一套吞吐型的业务，它也有实时的全量写，而且也需要二级索引来生成多维报表。这个报表的场景跟DUMP场景不一样，这个单表比较大，在我们业务里面最大的表在压缩之后有80个TB。

## 物联网设备信息存储



物联网场景也非常典型，读写相对比较简单，但是数据量特别大，对写吞吐要求很高。在这种情况下，存储的成本以及写的吞吐能力、扩容能力，这些是HBase比较擅长。在成本这块，采用冷热分离存储以及压缩算法降低成本。



HBase SQL的场景基本上都是HBase自己的场景，海量的数据、线性扩展等等，但SQL赋予了HBase丰富的查询语义，从而拓展了HBase的业务边界。

## Ali-Hbase SQL

性能领先	功能丰富	稳定可靠
<ul style="list-style-type: none"><li>➢ 前缀BloomFilter</li><li>➢ BucketCache</li><li>➢ CrossRegion写合并</li><li>➢ CompactedConcurrentSkipListMap</li><li>➢ 连接数优化</li><li>➢ 锁优化</li><li>➢ Netty替换RPC</li><li>➢ HLog压缩</li><li>➢ 新的Block Encoding格式</li><li>➢ ...</li></ul>	<ul style="list-style-type: none"><li>➢ 离散式TTL</li><li>➢ 资源隔离</li><li>➢ 异构介质多副本</li><li>➢ 异步API</li></ul>	<ul style="list-style-type: none"><li>➢ 完善的跨集群数据链路<ul style="list-style-type: none"><li>✧ 毫秒级延迟</li><li>✧ 拓扑可视化</li><li>✧ 量化的延迟</li><li>✧ 多地多单元</li><li>✧ 链路隔离</li><li>✧ 支持同步复制</li></ul></li><li>➢ 一键切换</li><li>➢ 自动切换</li><li>➢ 快速故障恢复</li><li>➢ 多年沉淀，久经双十一沙场，坚如磐石</li></ul>

为了支持这些场景，我们在HBase做了很多的优化和改进，在HBase本身我们针对阿里的场景做了很多性能和功能上的变革。在稳定性方面，我们做了很多工作，能够让HBase久经双十一沙场。Ali-HBaseSQL与Phoenix在功能补齐、功能增强、数据导入导



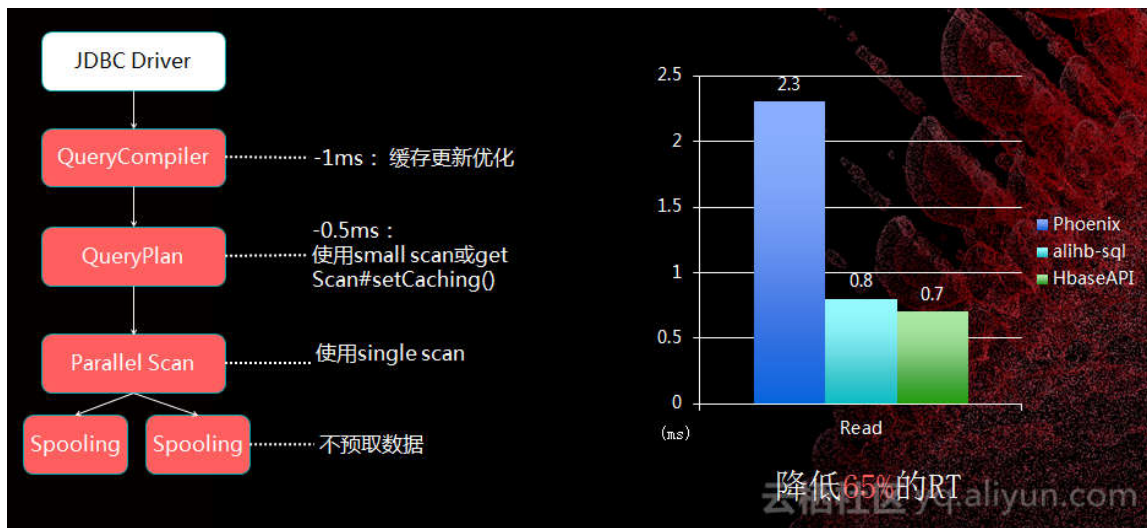
出方面有所改善。**Phoenix**在索引选择方面有缺陷，我们重做了一套基于**RBO**的查询优化器，同时增强了可以访问多张索引表的能力。另外，也配套开发了数据导入导出插件，使得数据可以在各个系统之间自如流动。

## 性能优化

目标是将简单请求的性能优化到极致，与对应的**HBase Native API**性能差距小于**5%**。单行读写的场景下，**SQL**和**HBase API**的差距很明显。

客户端的元数据缓存，元数据：列名、数据类型、表属性、索引信息等等。元数据更新策略：并不是每次都刷新元数据，我们做了周期性的刷新，通过版本号的方式来识别是不是最新的，如果不是最新的就更新一版，这是优化**UPSERT**的缓存更新策略。





**SELECT**优化，我们会根据用户请求的类型来选择使用**scan**或者**get**，这个选择对性能影响非常大，因为我们的目标是优化简单的请求，在极高的**tps**情况下，**RT**上优化一点点在用户那边的体现都是非常明显的。由于我们并不是分析型的场景，并不需要数据的预取，所以，**spooling**这部分可以直接移除。做了这些事情以后，**SQL**的读性能已经跟**HBase** 原生**api**比较接近了。

此外，**HBase + SQL**也在阿里云上提供，支持从其他的**RDBMS**迁移至云**HBase**。

未来的工作



未来的计划是支持列名映射以及 **ImmutableDataEncoding**，我们现在正在调研这两个特性。列名映射在大宽表的情况下能够节省1/3—1/2的存储空间。

**immutableData**编码能进一步节省近50%的存储空间，但其限制是数据不能修改。

另外，重客户端也是需要改变的，目前，我们要优化功能或者修复bug都需要让用户去升级**SQL**客户端，这是非常恶心的事情；所以，支持**query server mode**和瘦客户端，可以有效解决产品不断迭代的问题，用户不需要升级也可以享受到我们的改进；

支持分布式**Sequence**，最终我们也要把**SQL**的能力做到分布式；

可选的索引一致性，异步全局二级索引，有些场景下用户不需要强一致性，比如说日志，最终在1分钟之内一致就**OK**了，所以我





们做一个异步的全局更新，更新成本也进一步降低了。

数据存储与数据库

分布式

性能优化

hbase

性能

SQL

API

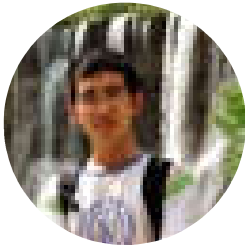
索引

物联网

报表

存储

作者



杨晗  
TA的文章

浅谈HBase的数据分布

相关文章

Ali-HBase的SQL实践与改进

