HBase 在人工智能场景的使用

近几年来，人工智能逐渐火热起来，特别是和大数据一起结合使用。人工智能的主要场景又包括图像能力、语音能力、自然语言处理能力和用户画像能力等等。这些场景我们都需要处理海量的数据，处理完的数据一般都需要存储起来，这些数据的特点主要有如下几点：

- 大：数据量越大，对我们后面建模越会有好处；
- 稀疏：每行数据可能拥有不同的属性，比如用户画像数据，每个人拥有属性相差很大，可能用户A拥有这个属性，但是用户B没有这个属性；那么我们希望存储的系统能够处理这种情况，没有的属性在底层不占用空间，这样可以节约大量的空间使用；
- 列动态变化：每行数据拥有的列数是不一样的。

为了更好的介绍 HBase
在人工智能场景下的使用，下面以某人工智能行业的客户案例进行分析如何利用 HBase
设计出一个快速查找人脸特征的系统。

目前该公司的业务场景里面有很多人脸相关的特征数据，总共3400多万张，每张人脸数据大概 3.2k。这些人脸数据又被分成很多组，每个人脸特征属于某个组。目前总共有近62W个人脸组，每个组的人脸张数范围为 1 ~
1W不等，每个组里面会包含同一个人不同形式的人脸数据。组和人脸的分布如下：

- 43%左右的组含有1张人脸数据；
- 47%左右的组含有 2 ~ 9张人脸数据；
- 其余的组人脸数范围为 10 ~ 10000。

现在的业务需求主要有以下两类：

- 根据人脸组 id 查找该组下面的所有人脸；
- 根据人脸组 id +人脸 id 查找某个人脸的具体数据。

MySQL + OSS 方案

之前业务数据量比较小的情况使用的存储主要为 MySQL 以及
OSS（对象存储）。相关表主要有人脸组表group和人脸表face。表的格式如下：
group表：

group_id	size
1	2

face表：

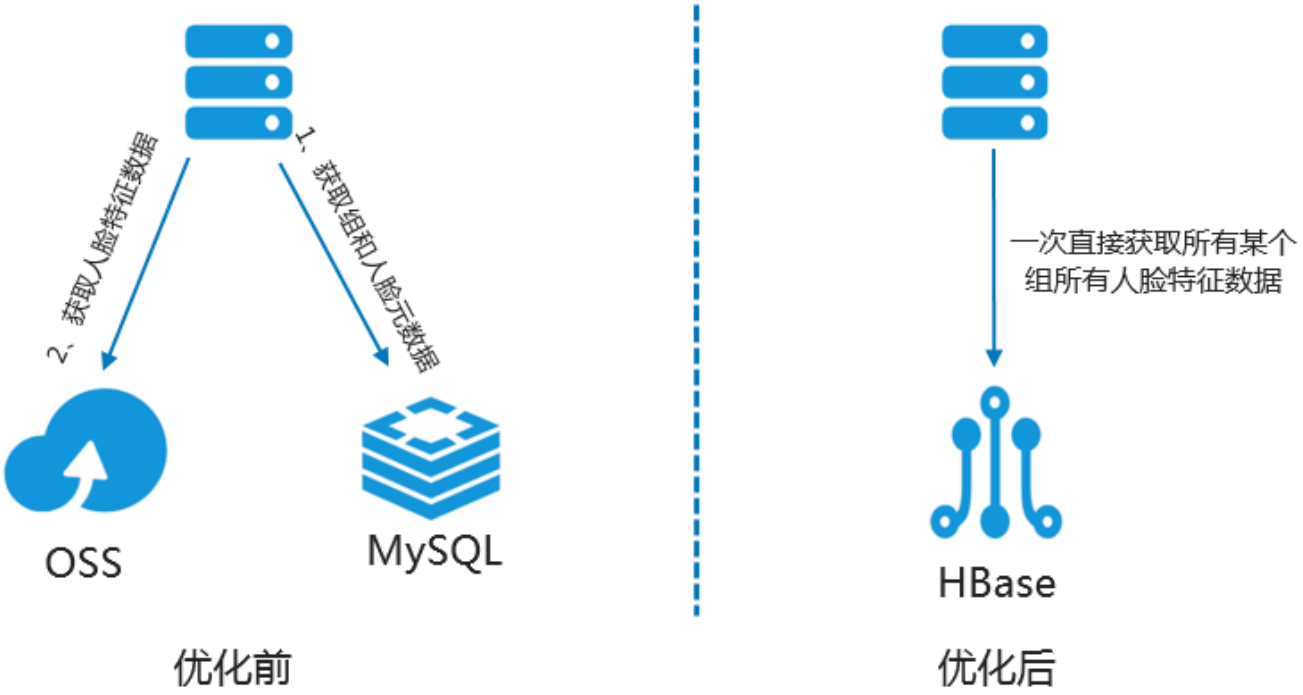
face_id	group_id	feature
"c5085f1ef4b3496d8b4da050cab0efd2"	1	"cwI4S/HO/nm6H....."

其中 feature 大小为3.2k，是二进制数据 base64 后存入的，这个就是真实的人脸特征数据。

现在人脸组 id 和人脸 id 对应关系存储在 MySQL 中，对应上面的 group 表；人脸 id 和人脸相关的特征数据存储在 OSS 里面，对应上面的 face 表。

因为每个人脸组包含的人类特征数相差很大（1 ~ 1W），所以基于上面的表设计，我们需要将人脸组以及每张人脸特征id存储在每一行，那么属于同一个人脸组的数据在MySQL 里面上实际上存储了很多行。比如某个人脸组id对应的人脸特征数为1W，那么需要在 MySQL 里面存储 1W 行。

我们如果需要根据人脸组 id 查找该组下面的所有人脸，那么需要从 MySQL 中读取很多行的数据，从中获取到人脸组和人脸对应的关系，然后到 OSS 里面根据人脸id获取所有人脸相关的特征数据，如下图的左部分所示。



如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

我们从上图的查询路径可以看出，这样的查询导致链路非常长。从上面的设计可看出，如果查询的组包含的人脸张数比较多的情况下，那么我们需要从 MySQL 里面扫描很多行，然后再从 OSS 里面拿到这些人脸的特征数据，整个查询时间在10s左右，远远不能满足现有业务快速发展的需求。

HBase 方案

上面的设计方案有两个问题：

- 原本属于同一条数据的内容由于数据本身大小的原因无法存储到一行里面，导致后续查下需要访问两个存储系统；
- 由于MySQL不支持动态列的特性，所以属于同一个人脸组的数据被拆成多行存储。

针对上面两个问题，我们进行了分析，得出这个是 HBase 的典型场景，原因如下：

- HBase 拥有动态列的特性，支持万亿行，百万列；
- HBase 支持多版本，所有的修改都会记录在 HBase 中；
- HBase 2.0 引入了 MOB（Medium-Sized Object）特性，支持小文件存储。HBase 的 MOB 特性针对文件大小在 1k~10MB 范围的，比如图片，短视频，文档等，具有低延迟，读写强一致，检索能力强，水平易扩展等关键能力。

我们可以使用这三个功能重新设计上面 MySQL + OSS
方案。结合上面应用场景的两大查询需求，我们可以将人脸组 id 作为 HBase 的
Rowkey，系统的设计如上图的右部分显示，在创建表的时候打开 MOB 功能，如下：

```
create 'face', {NAME => 'c', IS_MOB => true, MOB_THRESHOLD => 2048}
```

上面我们创建了名为 face 的表，IS_MOB 属性说明列簇 c 将启用 MOB 特性，MOB_THRESHOLD 是 MOB 文件大小的阈值，单位是字节，这里的设置说明文件大于 2k
的列都当做小文件存储。大家可能注意到上面原始方案中采用了 OSS
对象存储，那我们为什么不直接使用 OSS
存储人脸特征数据呢，如果有这个疑问，可以看看下面表的性能测试：

对比属性	对象存储	云 HBase
建模能力	KV	KV、表格、稀疏表、SQL、全文索引、时空、时序、图查询
查询能力	前缀查找	前缀查找、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

对比属性	对象存储	云 HBase
扩展性	优	优
适用对象范围	通用	<10MB

根据上面的对比，使用 HBase
MOB特性来存储小于10MB的对象相比直接使用对象存储有一些优势。
我们现在来看看具体的表设计，如下图：

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

上面 HBase 表的列簇名为c，我们使用人脸id作为列名。我们只使用了 HBase
的一张表就替换了之前方面的三张表！虽然我们启用了
MOB，但是具体插入的方法和正常使用一样，代码片段如下：

```
String CF_DEFAULT = "c";
Put put = new Put(groupId.getBytes());
put.addColumn(CF_DEFAULT.getBytes(),faceId1.getBytes(), feature1.getBytes());
put.addColumn(CF_DEFAULT.getBytes(),faceId2.getBytes(), feature2.getBytes());
.....
put.addColumn(CF_DEFAULT.getBytes(),faceIdn.getBytes(), featuren.getBytes());
table.put(put);
```

用户如果需要根据人脸组id获取所有人脸的数据，可以使用下面方法：

```
Get get = new Get(groupId.getBytes());
Result re=table.get(get);
```

这样我们可以拿到某个人脸组id对应的所有人脸数据。如果需要根据人脸组id+人脸id查找某个人脸的具体数据，看可以使用下面方法：

```
Get get = new Get(groupId.getBytes());
```

```
get.addColumn(CF_DEFAULT.getBytes(), faceId1.getBytes())
Result re=table.get(get);
```

经过上面的改造，在2台 HBase worker 节点内存为32GB，核数为8，每个节点挂载四块大小为250GB 的 SSD 磁盘，并写入 100W 行，每行有1W列，读取一行的时间在100ms-500ms左右。在每行有1000个face的情况下，读取一行的时间基本在20-50ms左右，相比之前的10s提升200~500倍。

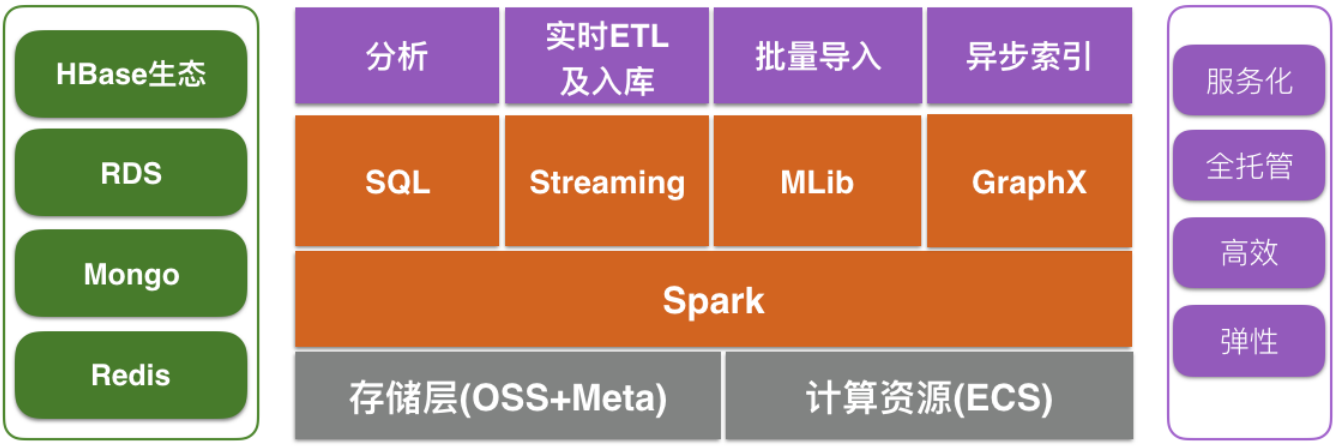
下面是各个方案的对比性能对比情况。

对比属性	对象存储	MySQL+对象存储	HBase MOB
读写强一致	Y	N	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

使用 Spark 加速数据分析

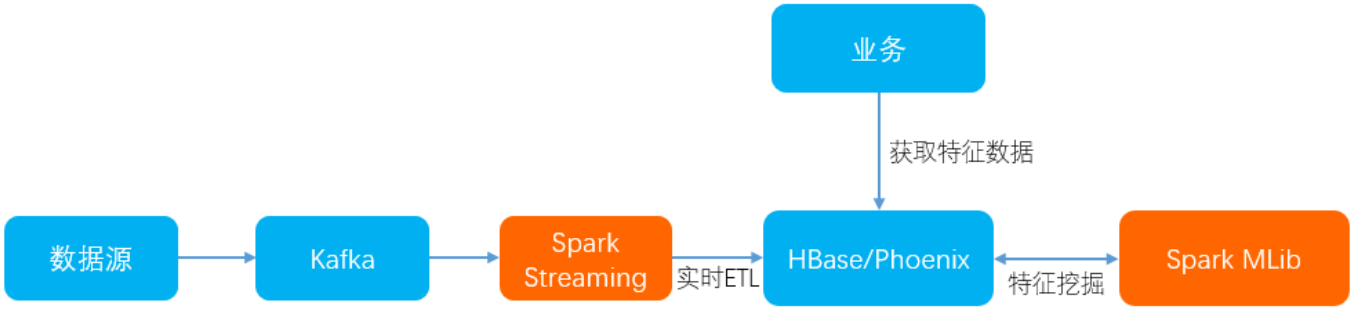
我们已经将人脸特征数据存储在阿里云 HBase 之中，这个只是数据应用的第一步，如何将隐藏在这些数据背后的价值发挥出来？这就得借助于数据分析，在这个场景就需要采用机器学习的方法进行聚类之类的操作。我们可以借助 Spark 对存储于 HBase 之中的数据进行分析，而且 Spark 本身支持机器学习的。但是如果直接采用开源的 Spark 读取 HBase 中的数据，会对 HBase 本身的读写有影响的。

针对这些问题，阿里云 HBase 团队对 Spark 进行了相关优化，比如直接读取 HFile、算子下沉等；并且提供全托管的 Spark 产品，通过SQL服务ThriftServer、作业服务LivyServer简化Spark的使用等。目前这套 Spark 的技术栈如下图所示。



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

通过 Spark 服务，我们可以和 HBase
进行很好的整合，将实时流和人脸特征挖掘整合起来，整个架构图如下：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

我们可以收集各种人脸数据源的实时数据，经过 Spark Streaming 进行简单的 ETL
操作；其次，我们通过 Spark MLib
类库对刚刚试试收集到的数据进行人脸特征挖掘，最后挖掘出来的结果存储到 HBase
之中。最后，用户可以通过访问 HBase 里面已经挖掘好的人脸特征数据进行其他的应用。