HBase在新能源汽车监控系统中的应用

重庆博尼施科技有限公司是一家商用车全周期方案服务商，利用车联网、云计算、移动互联网技术，在物流领域 为商用车的生产、销售、使用、售后、回收各个环节提供一站式解决方案，其中的新能源车辆监控系统就是由该公司提供的，本文是阿里云客户重庆博尼施科技有限公司介绍如何使用阿里云 HBase 来实现新能源车辆监控系统。该系统主要用于东风轻卡等新能源商用车监控服务，目前该系统正在阿里云线上稳定运行。

新能源车辆监控系统是一个车辆网服务平台，具有高并发、数据量大、实时性要求高等特点。对车辆监控系统来说最重要的问题就是如何处理车辆产生的海量数据，我们估计，当车辆数量增长到10万时，每天会产生大约2TB的数据，这些数据不仅需要存储，还需要做到实时可查。本文将介绍项目的背景和系统的基本架构，随后介绍我们在开发过程中遇到的各种问题以及解决方案。

项目背景

本项目为车联网监控系统，系统由车载硬件设备、云服务端构成。车载硬件设备会定时采集车辆的各种状态信息，并通过移动网络上传到服务器端。服务器端接收到硬件设备发送的数据首先需要将数据进行解析，校验，随后会将该消息转发到国家汽车监测平台和地方汽车监测平台，最后将解析后的明文数据和原始报文数据存储到系统中。车辆的数据和其他数据需要通过web页面或rest API接口进行查询访问。要求半年内的数据查询响应时间在毫秒级别内，超过半年的数据需要放到更加低成本的介质上，查询延迟在3s以内，这些数据的查询频次比较低。系统的主要参数有以下几项:

- 10万台车辆同时在线；
- 车辆正常情况下平均每分钟发送两个数据报文到监控平台；
- 若车辆处于报警状态，则平均一秒钟发送一次数据报文；
- 数据情况：（1）、车辆数据报文平均大小为1KB;（2）、解析后的数据包大小为7KB；（3）、平均一台车每天会产生20MB的数据；（4）、系统每天会产生2TB的数据；（5）同时系统有2.9亿行的数据需要写入到数据库中；
- 系统并发量：（1）、3300的持续并发量；（2）、10万个长连接；（3）、每秒3.3MB的原始数据需要被解析；（4）、每秒23.1MB的解析数据需要被存储。

系统设计

系统的技术选型对初创公司来说至关重要，所以在设计系统的时候尤为小心。经过仔细分析，我们要求新系统必须满足以下几个条件：

- 必须能够支持海量数据的不间断写入，而且能够存储PB级别的数据，具有高扩展性、高可靠性等；
- 能够支持简单的关键字查询，响应时间在秒级别内；
- 能够兼容大数据生态产品(如Spark、Hive、Hadoop等)，同时支持离线和准实时OLAP；
- 优先选择有雄厚实力的商业公司支持的云平台，最大限度减少运维成本。

为何选择HBase以及阿里云

因为车辆的监控数据非常大，传统关系型数据库（如Mysql、pg等）已经无法胜任存储工作，所以我们需要选用一种分布式数据库用于存储车辆实时数据。
我们在市场上能够找到分布式数据库有MongoDB和 HBase。

- MongoDB：MongoDB是一种我们曾经使用使用过的数据库，该数据库是一种基于文档的数据库。支持分片副本集扩展，但是由于MongoDB的分片集群维护成本过高，另外查询性能严重依赖索引，扩容时分片的迁移速度过慢。所以在这一版本的监控系统中我们并未采用。
- HBase：HBase底层存储基于HDFS的面向列数据库，其核心思想来源于谷歌三大论文内的bigtable。在谷歌和开源界均拥有丰富的应用实践经验。除此之外，HBase还有以下特点：（1）、支持PB级别的数据量；（2）、压缩效率非常高；（3）、支持亿级别的QPS；（4）、在国内外很多大型互联网公司使用；（5）、HBase添加节点及扩容比较方便，无需DBA任何干预。

经过对这几种数据库的分析，我们最终选用HBase，其满足我们前面提到的四个要求，而且还提供Phoenix插件用于SQL语句的查询。

作为初创公司，我们的运维能力有限，我们需要业务的快速落地。所以自建机房以及运维团队意味着前期较大的投入以及高昂的运维成本，所以我们决定使用云方案。

经过比较国内的各大云厂商，我们最终选用了阿里云平台，因为阿里云提供SaaS化的HBase服务，同时阿里云HBase支持很全面的多模式，支持冷数据存放在OSS之中，节约成本；支持备份恢复等特性，做到了真正的native cloud的数据库服务。另外，HBase 在阿里内部部署超过12000台机器，历经7年天猫双11的考验，这些实际数据以及经验增强了我们对阿里云HBase的技术信心，同时满足了我们的技术和业务需求。

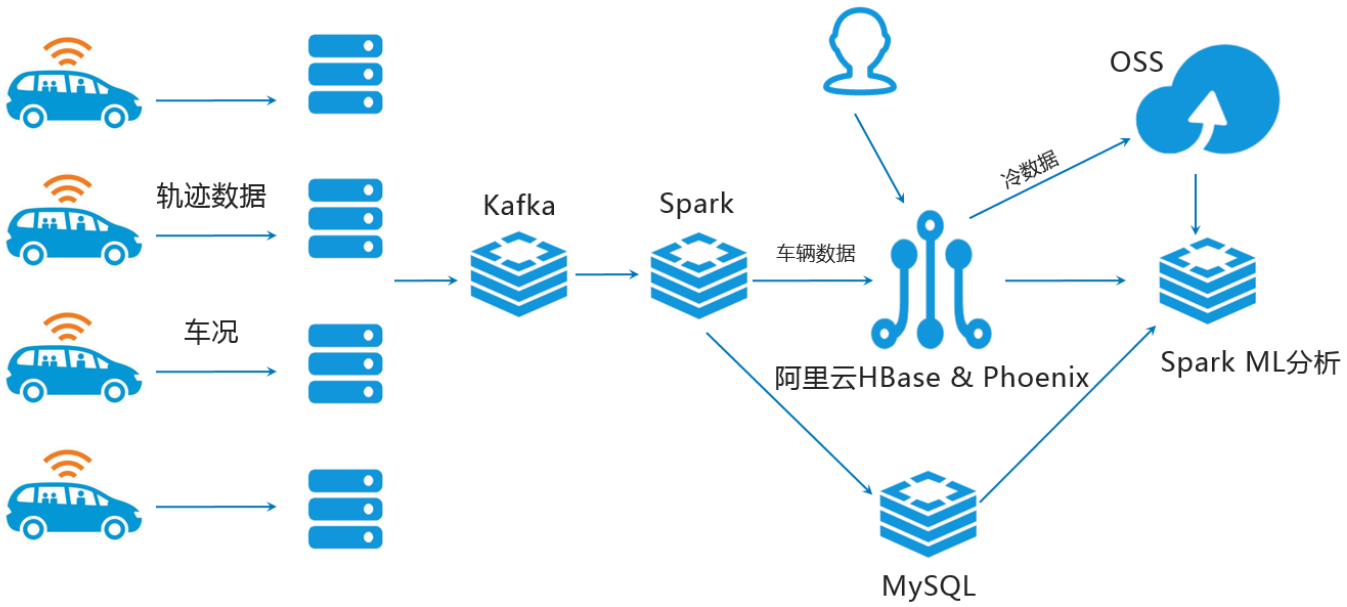
层级系统架构

系统采用层级架构以方便后期扩展和维护，现在主要分为以下几层：

- 接入层
  ：主要负责管理车辆设备的长连接，认证接收车辆发送的数据，下放数据和指令等操作。
- 消息队列层：主要负责缓存接入层收到的车辆实时数据。
- 实时数据处理与解析层
  ：主要负责解析车辆上传的实时数据，并将其存储到数据库中。另外还需要负责数据转发、指令生成等工作。
- 应用层：主要负责处理各种业务逻辑、将解析后的数据进行分析整理提供给用户使用。

系统架构预览

最终新能源监控系统的系统架构设计如下



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

图中最左端为监控的车辆，它会实时采集车辆的各项数据，并把采集到的数据通过移动互联网发送到平台。平台验证完数据会将其写入到Kafka消息队列中。流式计算服务器从Kafka消息队列中取出车辆的原始数据，并对车辆的数据进行解析、存储、转发等操作。HBase集群负责存储车辆实时数据，MySQL负责存储组织关系数据。同时，我们还会将超过一定时间（比如半年前）的数据转存到OSS存储介质中，以便降低存储成本。Spark
ML会对系统中的各项数据进行分析。终端用户会从HBase中查询一些数据。

HBase使用难点

Row key 设计

团队在使用HBase之前一直使用MySQL关系型数据库，在系统设计之初并没有考虑HBase的特性，而是按照关系型数据库的设计原则设计。经过一段时间的了解后才知道HBase主要使用Row key进行数据查询。Row key的设计至关重要。 目前系统中设计的Row key如下

- 设备ID +
  时间戳
  ：此种方式可以快速定位单台车辆。但是由于设备ID前缀为型号，一旦某批次的设备或车辆发送故障，则会造成HBase的某个Region写入压力过大。
- subString(MD5(设备ID),x) + 设备ID +
  时间戳
  ：此种方式可以将所有车辆打散到每个Region中，避免热点数据和数据倾斜问题，式中的x一般取5左右。但对于某个型号的车辆查询就只能够每台车辆单独进行查询了。

复杂查询问题

虽然通过Row key的设计可以解决部分数据查询的需求，但是在面对复杂需求时难以通过Row key直接索引到数据，若索引无法命中，则只能进行大范围或全表扫描才能够定位数据。所以我们在使用HBase时尽量避免复杂的查询需求。但业务方面仍然会有部分较为复杂的查询需求。针对这些需求，我们主要使用两种方式来建立二级索引。

- 手动在事务产生时将索引写入到HBase表中
  ：使用这种方式建立索引虽然可以不借用第三方插件，但是事务的原子性很难得到保障，业务代码也会因为索引的变化而难以维护。另外索引的管理也较为麻烦，后期的数据迁移很难能够。
- 通过Phoenix构建索引
  ：通过Phoenix构建索引可以避免事务原子性问题，另外也可以通过重建索引来进行数据迁移。因为使用的SQL语句，开发人员更能够利用之前关系型数据库的设计经验建立数据索引。

目前新能源监控系统中主要使用Phoenix实现二级索引，大大增加了数据的查询使用场景。

虽然Phoenix能够通过二级索引实现较为复杂的数据查询，但对于更为复杂的查询与分析需求就显得捉襟见肘。所以我们选用了Spark等其他数据分析组件对数据进行离线分析，分析后对结果通过接口提供给用户。

多语言连接问题

团队使用Python语言构建系统，但HBase使用Java语言编写，原生提供了Java API，并未对Python语言提供直接的API。我们使用happybase连接HBase，因为它提供了更为Pythonic的接口服务。另外我们也是用QueryServer 作为Python组件和Phoenix连接的纽带。

HBase冷数据存储

系统中车辆数据分为热数据和冷数据，热数据需要HBase中实时可查，冷数据虽不需要实时可查，但却需要一直保存在磁盘中。阿里云HBase支持将冷数据直接存储在OSS中，而这些数据的转存只需要简单的设置表相关属性，操作非常简单。将冷数据存储在OSS之中大大减少了数据的存储成本。

总结

首先，本文介绍了新能源车辆监控系统的项目背景，随后分析了本项目的项目难点，并介绍了我们团队的各种解决方案。针对项目需求，介绍了我们选择HBase的原因，及在HBase数据库使用过程中的经验和痛点。

展望

未来，我们会在系统接入大量车辆后，使用golang重写高性能组件以满足后期的并发性能需求。由于项目初期考虑到开发时间的问题，并未采用服务拆分的方式进行开发，这限制了系统的可扩展性，后期我们会根据实际业务需求，将系统切分成相对独立的模块，增强扩展性可维护性。
另外，车辆数据积累到一定程度后，我们可以利用这些数据进行大数据分析, 如车辆的故障诊断，

车辆状态预测等，这样就可以在车辆出现问题前提前发出预警，为车主和保险公司避免更大的损失，降低运营成本。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接: 【】（ ）