



基于 Kudu 的大数据平台实时业务处理能力提升方案

顾飞杨, 孔莹

(中国电信股份有限公司上海分公司, 上海 200042)

摘 要: 针对目前 Hadoop 大数据平台实时业务处理能力较差的难点, 研究了国际最先进的 Kudu 列存储作为 HDFS 块存储的有效补充的理论, 阐述了利用 Kudu 和 Spark 提供的主键索引和内存加速, 有效解决大数据平台无法支持实时入库、增量更新和 SQL 关联查询等业务痛点的技术实现方法。实验效果证明了方法对提升大数据平台实时业务处理能力的作用。

关键词: Kudu; 大数据; 列存储; 主键索引; 内存加速; 实时入库; 增量更新; SQL 关联查询

中图分类号: TP311.13

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2019158

Scheme of enhancing real-time business processing capabilities based on Kudu for the big data platform

GU Feiyang, KONG Ying

Shanghai Branch of China Telecom Group Co., Ltd., Shanghai 200042, China

Abstract: Aiming at the difficulty of real-time business processing capability of Hadoop big data platform, the theory of the most advanced Kudu column storage was studied as an effective complement of HDFS block storage, and the use of primary key index and memory acceleration provided by Kudu and Spark was illustrated to effectively solve the big data platform cannot support the technical implementation methods of business pain points such as real-time warehousing, incremental update and SQL-related query. The experimental results prove the effect of the method on improving the real-time business processing capability of the big data platform.

Key words: Kudu, big data, column storage, primary key index, memory acceleration, real-time access, incremental update, SQL join query

1 引言

中国电信股份有限公司上海分公司(简称上海电信)的大数据平台经过几年的建设和升级扩容,数据规模不断增加,业务应用不断扩展,满足了上海电信大数据应用的多种业务需求。随着

上海电信业务处理的实时性要求不断提高,大数据平台存在不支持流式数据实时写入、不支持流式数据增量更新、不支持实时数据复杂查询等实时业务应用难点,极大地影响了大数据平台的业务支撑能力。

Kudu 存储引擎作为 Hadoop 平台的增强组



件，可以与现有的 HDFS 组件优势互补，有效解决 Hadoop 平台实时业务应用难点，满足上海电信业务数据的实时处理要求，大大提升大数据平台实时业务处理能力。

2 实时业务应用难点

上海电信大数据平台采用的核心技术是 HDFS 分散存储和 MapReduce 并行计算，可以对传统技术无法处理的大规模数据，进行存储、加工和分析，填补了大规模数据处理的空白^[1]。

随着非日志型的业务数据不断汇聚到大数据平台，Hadoop 大数据平台面临着以下实时业务应用难点。

- HDFS 文件系统支持文件存储、数据块存储等多种存储方式，存储方式灵活，但不支持数据增量存储。
- Hive 数据库写入速度快，支持海量数据存储。但数据查询效率较低，不支持秒级查询。
- HBase 数据库采用 K/V 查询模式，查询效率高，支持秒级查询。但不支持复杂 SQL 操作^[2]。

大数据平台急需对流式数据实时处理的相关技术进行研究，提升大数据平台实时业务处理能力。

3 业界解决方案

针对大数据平台实时业务应用难点，业界目前主要有 Phoenix+HBase 和 Spark+Elasticsearch 两种解决方案^[3]。

3.1 方案 1: Phoenix+HBase

Phoenix 是构建在 HBase 上的一个 SQL 层，能让人们用标准的 JDBC API 而不是 HBase 客户端 API 来创建表、插入数据和对 HBase 数据进行查询。Phoenix 查询引擎会将 SQL 查询转换为一个或多个 HBase scan，并编排执行以生成标准的 JDBC 结果集。直接使用 HBase API、协同处理器

与自定义过滤器，适合简单的 SQL 查询操作，可以达到毫秒级响应。

该方案存在以下缺点：Phoenix 和 HBase 组合，开发难度高，业务移植困难；Phoenix 多表关联 join 语句，需要全表遍历，复杂 SQL 查询效率低；Phoenix 采用单节点运行作业，不是分布式运行作业，存在单点负荷不均衡的问题。

3.2 方案 2: Spark Streaming + Elasticsearch

Elasticsearch 是一个实时的分布式搜索和分析引擎。Elasticsearch 作为数据库，执行 SQL 脚本，包括执行创建临时表的动作。Spark Streaming 通过 RDD 将流式数据写入 Elasticsearch。

该方案存在以下缺点：Spark Streaming 和 Elasticsearch 组合，开发难度高，业务移植困难；Elasticsearch 数据增量更新需要更新索引，容易出现数据不同步；Elasticsearch 对于复杂 SQL 操作（包括多表关联等执行效率不高）。

4 Kudu 应用创新

4.1 Kudu 应用方案

Kudu 是 Cloudera 开源的新型列式存储系统，专门为了对快速变化的数据进行快速的分析，填补了以往 Hadoop 存储层的空缺。Kudu 的扫描速度几乎和原生 HDFS 文件系统一样快，同时随机访问速度和 HBase 数据库几乎一样快^[4]。

Kudu 有 Master server 和 Tablet server 两种类型的组件。Kudu 使用单个的 Master server 节点来管理集群的元数据，并且使用任意数量的 Tablet server 节点用来存储实际数据。可以部署多个 Master server 节点来提高容错性。Master server 节点负责整个集群的元数据管理和协调，它承担着以下功能。

- 作为元数据管理者，Master server 节点管理着集群中所有 Table 和 Tablet 的 schema 及一些其他的元数据。
- 作为集群协调者，Master server 节点追踪着

所有 server 节点是否存活, 并且当 server 节点挂掉后协调数据的重新分布。

- 作为 Tablet 管理者, Master server 节点跟踪每个 Tablet 的位置。

Master server 的所有信息都在内存中, 因此数据访问速度非常快, 每次查询都在毫秒级别。基于 Kudu 存储引擎技术, 大数据平台可以在现有业务应用不需要配合改造的前提下, 实现大数据平台实时业务处理能力提升, 总体架构如图 1 所示。

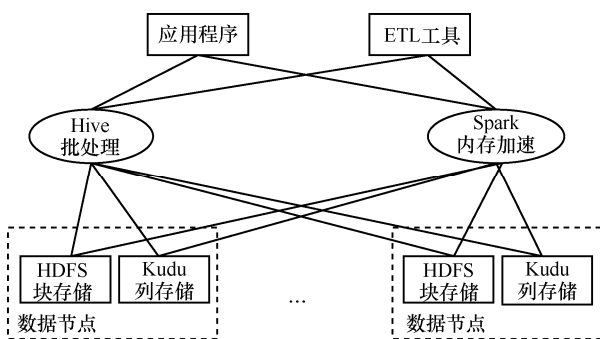


图1 Kudu 大数据应用总体架构

总体架构说明如下。

- Kudu 组件作为 HDFS 存储方式的升级和补充, 可以和 HDFS 文件系统共存, 一起部署在数据节点上; Kudu 组件提供增量更新、复杂逻辑以及交互式查询等数据处理能力, 满足实时业务数据处理需求, 大大提升大数据平台业务支撑能力。
- Kudu 组件可以和大数据平台现有的 MR 并行计算、Hive 数据库、Spark 内存并行计算、Impala 交互式查询等应用组件无缝对接, 极大拓展了大数据平台的业务应用场景。
- 应用程序和 ETL 工具, 可以根据需求调用不同的组件, 完成底层数据的访问和更新。

Kudu 应用方案优点为: 一体化原生方案, 开发难度低, 业务移植方便; Kudu 可以保证数据增量更新一致性; Kudu 可以实现高效率的复杂 SQL 操作。

4.2 基于 Kudu 的流式数据实时入库

大数据平台通常使用 HDFS 文件系统来存储

Kafka 流式实时消息系统。其中 HDFS 文件系统的基本存储单位是数据块, 默认大小 64 MB, 数据入库时需要每个数据块逐个写入。而且 HDFS 文件系统采用 Hadoop 的 MapReduce 批处理计算框架, 适合处理海量数据的批量处理数据集。因此 Kafka 流式数据批量写入大数据平台 HDFS 数据块时, 数据入库周期为分钟以上级别, 实时性较差。

本文使用 Kudu 存储引擎直接对接 Kafka 实时消息系统。Kudu 是支持实时数据处理的 OLAP 数据库, 采用分布式列式存储技术, 可以支持流式数据逐条写入, 因此 Kudu 数据表能够满足 Kafka 流式数据的实时写入需求, 单条数据入库周期为毫秒级别。

基于 Kudu 的流式数据入库数据处理流程如图 2 所示。

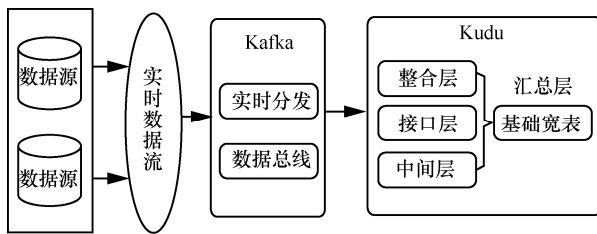


图2 基于 Kudu 的流式数据入库数据处理流程

基于 Kudu 的流式数据实时入库测试结果见表 1。

表 1 基于 Kudu 的流式数据实时入库测试结果

表名	导入数据量/条	时间/s
wls_carrier	11 284	2.43
trs_dwdmwave	28 083	5.92

基于 Kudu 的流式数据入库时间为秒级, 可以保证实时业务数据的入库要求。

4.3 基于 Kudu 的流式数据增量更新

大数据平台通常使用 HDFS 文件系统和 Hive 数据库更新大数据平台的存量数据, HDFS 文件系统的基本存储单位是数据块, 默认大小为 64 MB。

目前 HDFS 数据块仅支持数据追加, 不支持文件



的随机修改，数据更新时必须整个 HDFS 数据块全部替换，因此无法支持单条记录的修改和增量更新。

本文推荐的 Kudu 存储引擎技术，是一种基于列存储的结构化数据的数据仓库，用户可以建立多个 table，每个 table 都有一个预先定义好的 schema 数据库表定义。Kudu 底层存储采用列存储模式，因此数据是可修改的，更新、插入、删除都是临时先缓冲于内存，随后合并进永久性列式存储中，因此 Kudu 是可以支持单条记录修改和增量更新。

Kudu 技术架构如图 3 所示。

Kudu 增量更新的方式是：一个 Table 的数据被分割成多个 Tablet，由它们对应的 Tablet server 提供数据读写服务。Tablet 由 RowSet 组成，一个 RowSet 包含两部分数据：基础数据通常以 DiskRowSet 形式保存在磁盘上；而变更数据先以 MemRowSet 的形式暂存在内存中，后续再异步地刷写到磁盘上。

基于 Kudu 的流式数据增量更新技术是真正实现“单条更新”和“实时更新”的技术方案，不是“批量更新”和“定时更新”，是真正的“实时增量更新”。

基于 Kudu 的流式数据增量更新测试结果见表 2。

表 2 基于 Kudu 的流式数据增量更新测试结果

表名	5 min		15 min	
	更新数据量	时间/s	更新数据量	时间/s
inc_addr_segmem_eqp	33	6	42	8
inc_asn_link_route	5 032	6	35 981	35
inc_cbl_pair	176	7	1 606	13
inc_c_servicecustomer	174	6	2 088	6
cbl_jnt_box	4	6	32	7

基于 Kudu 的流式数据增量更新时间均为秒级，可以保证增量更新业务的实时性。

4.4 基于 Kudu 的实时数据复杂查询

大数据平台通常使用 Hive 数据库进行复杂 SQL 查询。Hive 是建立在 Hadoop 上的数据仓库，Hive 数据库采用类 SQL 的查询方式，将 SQL 查询提交到在 Hadoop 大数据平台上作为 Job 作业执行。由于 Hive 数据库构建在基于静态批处理的 Hadoop 的 MapReduce 之上，Hadoop 通常都有较高的时延并且在作业提交和调度的时候需要大量的开销，因此 Hive 数据库并不能够在大规模数据

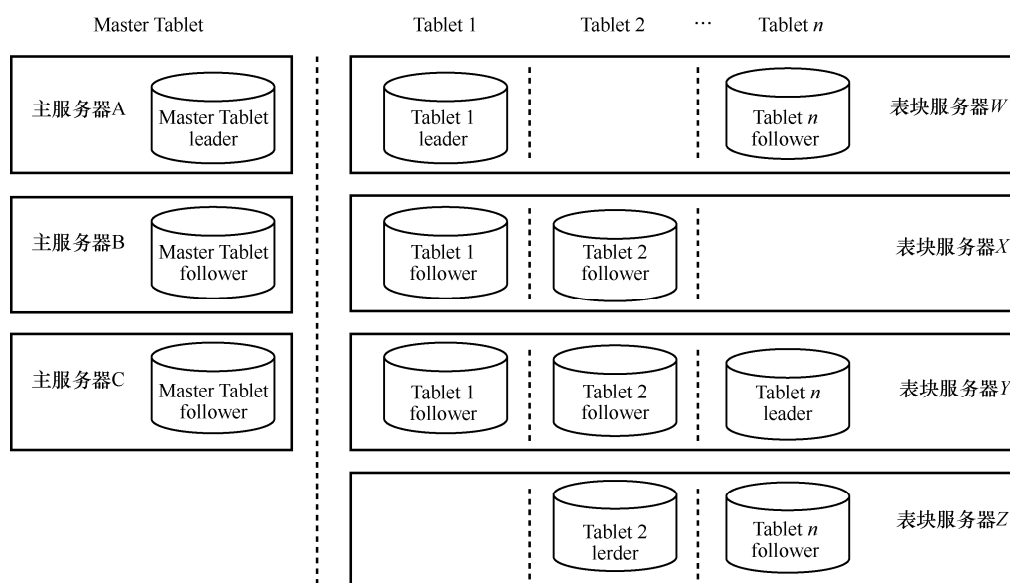


图 3 Kudu 技术架构

集上实现低时延、快速的查询。Hive 数据库的复杂 SQL 关联查询响应时间为分钟至小时级别,查询的实时性较低。

本文通过 Kudu 存储引擎以及 Spark 内存计算框架,实现高效率的复杂 SQL 查询。Spark 是专为大规模数据处理而设计的快速通用的计算引擎,相比 MapReduce 计算引擎,Spark 计算引擎采用内存作为缓存,能够更高效地完成分布式计算任务。Kudu 作为底层存储,可以支持 MPP 并行查询而且可以支持主键索引功能,通过内存缓存和索引功能的加速,因此 Kudu 可以大大提升 SQL 关联查询的实时性。

Kudu 存储引擎和 Spark 内存计算框架集成,对外提供基于 SQL 的实时查询分析服务,主要有以下技术优势:

- 实现业务数据的实时写入和实时更新,对分析应用来说是即时可见的,无时延;
- Spark 对数据的过滤条件可以下推到 Kudu 这一存储层,能提高数据读取/扫描的性能;
- 基于 Kudu 主键索引的 SQL 查询,SQL 查询性能更高。

基于 Kudu 的复杂 SQL 查询测试结果如图 4 所示,具体如下:

- 复杂 SQL 语句操作<将近 100 行 SQL 语句操作>;
- 平均执行时间为 1.79 s。

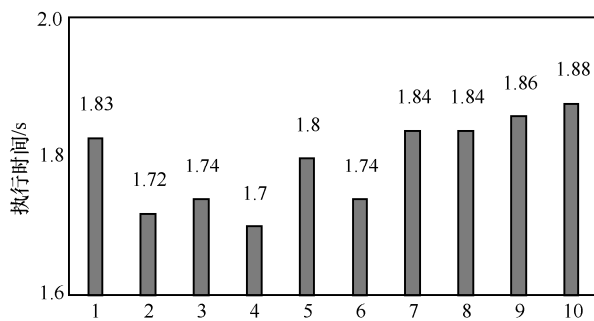


图 4 基于 Kudu 的复杂 SQL 查询测试结果

基于 Kudu 的复杂 SQL 查询时间均为秒级,可以保证复杂查询业务的实时性。

4.5 Kudu 应用效果

Kudu 存储引擎满足流式数据实时入库、流式数据增量更新、实时数据复杂查询的实时业务应用需求,可以提升大数据平台实时业务处理能力。

Kudu 应用效果总结见表 3。

根据上述 Kudu 应用效果,本项目技术先进,各项功能指标、性能指标均业界领先。

5 结束语

本文针对目前 Hadoop 大数据平台实时业务处理能力较差的难点,在现有大数据平台业务应用不进行改造的前提下,通过引入 Kudu 的列存储模式,作为 HDFS 块存储的有效补充,同时利用 Spark 具备内存缓存的 SQL 实时查询,逐步替代传统的 Hive 批处理的 SQL 查询,从而满足运营商级别的实时业务处理的迫切需求,流式数据处

表 3 Kudu 应用效果总结

实时业务应用	Kudu 应用前	Kudu 应用后
流式数据实时入库	块存储 HDFS 整个数据块写入 实时性较差,分钟级别	列存储 Tablet 逐条写入 实时性高、秒级
流式数据增量更新	HDFS 不支持更新 不支持增量更新,只能插入 全量更新时间为分钟至小时级别	列存储支持更新 支持增量更新和插入 增量更新时间为秒级
实时数据复杂 SQL 查询	不支持内存缓存 没有索引功能 查询时间为分钟至小时级别	支持内存缓存 具备主键索引功能 查询时间为秒级



理时间从原来的分钟级别缩短到秒级别，同时支持增量数据更新，并且提高了复杂数据查询的效率，大大提升了 Hadoop 大数据平台实时业务处理能力。

参考文献:

- [1] 汤劲松, 蔡韵, 王晓亮, 等. 电信运营商大数据基础能力建设与实践[J]. 电信科学, 2018, 34(1): 158-165.
TANG J S, CAI Y, WANG X L, et al. Construction and practice of big data fundamental ability for telecom operators[J]. Telecommunications Science, 2018, 34(1): 158-165.
- [2] 王瑞通, 李炜春. 大数据基础存储系统技术研究[J]. 计算机技术与发展, 2017(8): 66-72.
WANG R T, LI W C. Research of big data storage system technology[J]. The Computer Technology and Development, 2017(8): 66-72.
- [3] 陈涛, 鲁萌, 陈彦名. 运营商大数据技术应用研究[J]. 电信科学, 2017, 33(1): 130-134.
CHEN T, LU M, CHEN Y M. Research on operators' big data technologies and applications[J]. Telecommunications Science, 2017, 33(1): 130-134.

- [4] 鲍远松. 基于 Kudu 的结构化数据存储分析方案设计[J]. 信息技术与标准化, 2017(10): 60-63.

BAO Y S. Structured data storage analysis scheme design based on Kudu [J]. Information Technology and Standardization, 2017(10): 60-63.

[作者简介]



顾飞杨 (1978-), 男, 中国电信股份有限公司上海分公司企业信息化部数据中心副主任、工程师, 主要研究方向为大数据技术以及应用等。



孔莹 (1982-), 女, 中国电信股份有限公司上海分公司企业信息化部数据中心副主任、工程师, 主要研究方向为异地跨机房的大数据平台集群、数据快速迁移等。