计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(9):65−71 [doi: 10.15888/j.cnki.csa.007040]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Hive 的高可用双引擎数据仓库①

李　翀[1], 张彤彤[1,2], 杜伟静[1,2], 刘学敏[1]

[1](中国科学院 计算机网络信息中心, 北京 100190)
[2](中国科学院大学, 北京 100190)
通讯作者: 张彤彤, E-mail: zhangtongtong@cnic.cn

摘　要: 打破信息孤岛, 整合异构数据, 汇聚共享交换, 深度分析挖掘, 提供行业领域辅助决策和态势分析具有深远的理论和应用价值. 本文以中国科学院教育科研态势感知服务的实际需求为牵引, 设计并实现了一套基于 Hive 的 Hadoop/Spark 双计算引擎大数据仓库, 支持多种方式 OLAP 分析, 进行了可用性、负载均衡、资源管理的优化设计, 为后续进行全院数据汇聚挖掘、知识图谱构建、学科态势分析提供了平台支撑. 实验表明, 系统灵活高效, 高可用可扩展, 资源调度科学, 负载均衡效果明显.

关键词: 数据仓库; Hive; 高可用; OLAP; Hadoop

引用格式: 李翀,张彤彤,杜伟静,刘学敏.基于 Hive 的高可用双引擎数据仓库.计算机系统应用,2019,28(9):65–71. http://www.c-s-a.org.cn/1003-3254/7040.html

High Availability Dual Engine Data Warehouse Based on Hive

LI Chong[1], ZHANG Tong-Tong[1,2], DU Wei-Jing[1,2], LIU Xue-Min[1]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Breaking isolated information island, integrating heterogeneous data, gathering and sharing exchanges, conducting in-depth analysis and mining, and providing industry-side decision-making and situation analysis have far-reaching theoretical and applied value. Based on the actual demand of the situational awareness service of the Chinese Academy of Sciences, this study designs and implements a Hive-based Hadoop/Spark dual computing engine big data warehouse supporting OLAP analysis in multiple ways, and carries out an optimization design of usability, load balancing, and resource management, which provides platform support for the subsequent data aggregation and mining, knowledge map construction and discipline situation analysis. Experimental results show that the system is flexible, efficient, available, and scalable, the resource scheduling is scientific, and the load balancing effect is obvious.

Key words: data warehouse; Hive; high availability; OLAP; Hadoop

随着大数据分析挖掘技术不断发展, 数据所蕴含的价值被不断发掘, 数据已成为各行各业社会团体最重要的资源之一.如何高效存储这种来自不同系统, 具有不同格式的多源异构数据, 怎样将这些科研管理数据整合利用, 打破信息孤岛, 进行数据挖掘, 辅助决策分析, 实现数据互通, 支持在线流处理、离线批处理及 OLAP、OLTP 不同场景的数据分析处理需求, 优化查询效率, 提供经济高效分析计算平台是当下研究热点.

在数字化高速发展的信息时代, 科研管理过程的实质是信息化管理的过程, 是对科研信息资源进行收集、整理、统计、分析并加以利用的过程[1,2]. 科研管理信息系统广泛应用于高校和科研院所, 已积累形式

① 基金项目: 中国科学院"十三五"信息化专项 (XXH13504-03)
Foundation item: CAS Special Fund for Informatization Construction in 13th Five-Year Plan (XXH13504-03)
收稿时间: 2019-02-28; 修改时间: 2019-03-14; 采用时间: 2019-03-18; csa 在线出版时间: 2019-09-05

© 中国科学院软件研究所 http://www.c-s-a.org.cn

多样、相互隔离、分布广泛, 标准各异的海量数据, 由于缺乏全局管理、治理维护、统一平台, 无法对科研成果科学评估及辅助决策制定提供有效支持.

本文以中国科学院科研管理态势感知和竞争力分析为研究背景, 依托中国科学院科研与教育态势感知服务项目, 以汇聚全院科研与教育投入、产出、成果、发展等结构化、半结构化和非结构化数据, 构建可扩展高可用大数据仓库、高效 OLAP 查询分析实际需求为切入点, 聚焦科研管理大数据汇集、存储、分析的需求, 研究、设计和构建面向全院科研管理大数据的数据仓库, 为项目后续在线分析、数据挖掘、搭建知识图谱、学科态势和竞争力分析等需求提供平台支持.

1 研究现状和相关工作

数据仓库 (data warehouse) 是一个面向主题的 (subject oriented)、集成的 (integrated)、相对稳定的 (non-volatile)、反映历史变化 (time variant) 的数据集合, 是在现有数据库的基础上, 对其中的数据再次进行抽取、加工和使用, 并最终用于管理决策的集合, 并不是简单的数据复制或数据累加[3,4]. 数据仓库当前主要的应用场景包括报表展示、实时查询、BI (Business Intelligence) 展示、数据分析、数据挖掘、模型训练等方面.它提供了一种有效的访问这些数据的方法, 可以帮助科研机构快速而正确的做出决策.

传统数据仓库大都是基于 Oracle、MySQL 这样的关系型数据库, 扩展成本高, 面对 PB 级别的数据量以及各种关系数据库、NoSQL 数据库、XML 文件等数据源, 其处理速度和处理效率不能够满足数据存储、查询以及融合多维度数据进行分析的需要[5].

广义上来说, Hadoop 大数据平台也可以看做是新一代的数据仓库系统, 它具有很多现代数据仓库的特征, 且具有低成本、高性能、高容错和可扩展等特性, 被企业所广泛使用. IBM 的研究人员将基于 Hadoop 平台的 SQL 查询系统分为两大类: Database-Hadoop hybrids 和 Native Hadoop-based systems. 第一类中只是使用了 Hadoop 的调度和容错机制, 使用关系数据库进行查询[6]. 第二类则充分利用了 Hadoop 平台的可扩展性, 主要分为 3 个小类: 1) 基于 MapReduce 的 Hive; 2) 基于内存计算框架 Spark 的 Spark SQL; 3) 基于 shared-nothing 架构的大规模并行处理 (Massively Parallel Processing, MPP) 引擎, 如 Impala. 在文献[7]和文献[8]对比分析了最具代表性的 Hive、Impala 和 Spark SQL 这 3 种 SQL-on-Hadoop 查询引擎, 实验表明 3 个查询引擎均有各自的优点, 综合来看, Hive 的查询结果准确率更高, 更为稳定, 但查询时延较为严重, 适合批处理; Impala 查询速度最快, 但系统稳定性有待提高; Spark SQL 处理速度处于二者之间, 更适合多并发和流处理场景.

基于 Hadoop 的多种 SQL 查询引擎各有优势, 但从稳定性、易用性、兼容性和性能多个方面对比分析, 目前并不存在各方面均最优的 SQL 引擎. 考虑到项目离线批处理和在线流处理的需求, 而目前较少有兼顾两种需求的数据仓库实施方案, 结合当下较为成熟的开源技术方案, 本文设计并实现了面向科研管理大数据的数据存储系统.

2 大数据仓库设计

传统数据仓库大都只用到结构化数据处理技术, 大数据仓库不仅要处理关系数据库中的结构化数据, 还要处理海量半结构化和非结构化数据, 并为大数据分析提供平台, 需要结合大数据技术设计和构建. 以下分别以相关技术分析选型、集群高可用设计、系统设计阐述大数据仓库设计思路.

2.1 技术选型

Hive 是基于 Hadoop 的数据仓库工具, 可以提供类 SQL 查询功能, 本质是将 SQL 查询转换为 MapReduce 程序. MapReduce 框架主要适用于大批量的集群任务, 批量执行导致时效性偏低, 并不适合在线数据处理的场景, 一般用来做数据的离线处理. 使用 Hive 来做用来做离线数据分析, 比直接用 MapReduce 程序开发效率更高. 因为大多数的数据仓库应用程序是基于关系数据库现实的, 所以 Hive 降低了将这些应用程序移植到 Hadoop 上的障碍[9].

MapReduce 框架及其生态相对较为简单, 对计算机性能的要求也相对较弱, 运行更稳定, 方便搭建及扩充集群, 适合长期后台运行. 但其执行速度慢, 不适合实时性要求较高的查询场景, 在保证系统稳定、减少运维难度的前提下, 融合同样基于 Hadoop 平台且系统相对稳定的 Spark 框架是更好的选择, 并且能为在线分析、数据挖掘等提供支持.

Spark 是借鉴了 MapReduce 框架并在其基础上发

© 中国科学院软件研究所 http://www.c-s-a.org.cn

展起来的,继承了其分布式计算的优点并改进了 MapReduce 明显的缺陷. Spark SQL 作为 Spark 生态主要组件之一, 与 Hive 基于 MapReduce 进行查询类似, Spark SQL 使用 Spark 作为计算引擎, 在使用时需要处于 Spark 环境. Spark SQL 几乎完全兼容 HiveQL 语法, 只是 Hive 特有的一些优化参数及极少用语法不支持.

Hive on Spark 是由 Cloudera 发起, 由 Intel、MapR 等公司共同参与的开源项目. 它把 Spark 作为 Hive 的一个计算引擎, 将 Hive 查询作为 Spark 任务提交到 Spark 集群进行计算. Hive On Spark 和 Spark SQL 只是 SQL 引擎不同, 并无本质的区别, 都是把 SQL 查询翻译成分布式可执行的 Spark 程序. 而 Hive on Spark 与 Hive on MapReduce 一样可以使用 HiveQL 语法. 如果要在数据仓库中使用 Spark 作为计算引擎, 融入 Hive on Spark 是更好的选择.

综上所述, Hive on Spark 与 Hive on MapReduce 结合, 可以高效切换计算引擎, 同时提高资源利用率, 降低运维成本.

2.2 高可用性

高可用性, 即 HA (High Availability), 指的是通过尽量缩短因日常维护和突发系统崩溃导致的停机时间, 以提高系统和应用的可用性.

分布式系统通常采用主从结构, 即一个主节点, 连接 N 个从节点. 主节点负责分发任务, 从节点负责执行任务, 当主节点发生故障时, 整个集群都会失效, 这种故障称为单点故障.

HDFS 集群的不可用主要包括以下两种情况: 一是主节点主机宕机, 导致集群不可用; 二是计划内的主节点软件或硬件升级, 导致集群在短时间内不可用.

在 Hadoop2.0 之前, 也有若干技术试图解决单点故障的问题. 如元数据备份、Secondary NameNode、Backup NameNode、Facebook AvatarNode 方案[10]等, 还有若干解决方案, 基本都是依赖外部的 HA 机制, 譬如 DRBD[11], Linux HA, VMware 的 FT 等等. 但以上方案存在需要手动切换、恢复时间过长、需要引入另一个单点等问题.

为了解决上述问题, Hadoop 社区在 Hadoop2.X 版本中给出了真正意义上的高可用 HA 方案: Hadoop 集群由两个 NameNode 组成, 一个处于 Active 状态, 另一个处于 Standby 状态. Active NameNode 对外提供服务, 而 Standby NameNode 仅同步 Active NameNode 的状态, 以便能够在它失败时快速进行切换. 其原理如图 1 所示. 集群通过 ZooKeeper 进行心跳检测, 通过 JournalNode 独立进程进行相互通信, 同步 NameNode 状态.

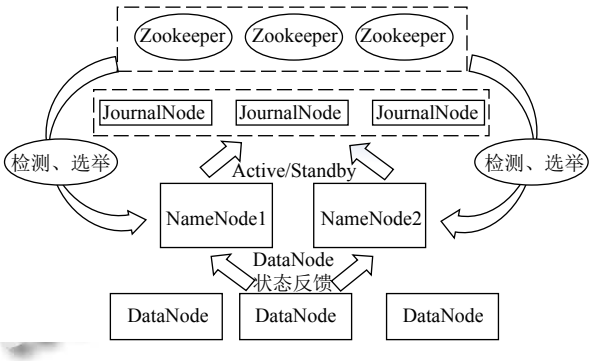

图 1 高可用原理

在生产环境中, 必然要考虑到集群的高可用性, 因此集群需要设置一个主节点备用节点, 在主节点出现故障后能够及时切换到备用节点, 保证集群可用性.

2.3 系统设计

基于以上分析,本文采用基于 Hive 的 MapReduce+Spark 双计算引擎混合架构进行大数据仓库系统设计, 满足了项目对于数据仓库高效、高可用和可扩展性的需求. 为更好的管理 Hadoop 和 Spark 两个计算集群, 提高集群资源的利用率及集群的计算效率, 采用 YARN (Yet Another Resource Negotiator) 进行资源管理, 保证了整个系统的稳定性和可靠性, 系统架构如图 2 所示.

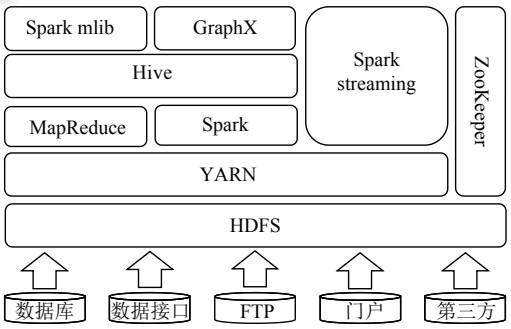

图 2 系统技术架构

系统将来自不同数据库、互联网、第三方的多源异构数据汇聚到 HDFS 文件系统, 采用 Hive 进行管理和索引, 再通过上层计算引擎对数据进行查询分析和计算. 通过 YARN 进行 Hadoop 集群和 Spark 集群的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

资源分配和管理, 并通过 ZooKeeper 实现系统中 Hadoop、Spark、YARN 组件的高可用性, 可按需扩展集群节点进行扩容.

依据计算需求不同, 通过配置或简单命令可以随时切换 Hive 计算引擎. 在对实时性要求不高或对稳定性要求较高的场景下使用 MapReduce 引擎; 对实时性有一定要求时使用 Spark 引擎. 两种引擎均使用 HiveQL 对数据进行操作, 无需切换开发环境, 可以高效利用集群资源对数据进行抽取、转换, 为机器学习和图计算提供数据源, 系统还可以通过 Spark Streaming 基于 HDFS 对数据进行流处理, 为实时流处理提供平台.

3 实现与优化

Hadoop 和 Spark 均依赖于 Java, 集群需要安装 JDK, 同时使用 MapReduce 和 Spark 作为数据仓库的计算引擎, 需要安装配置 Hive.

以 Spark 作为计算引擎时, 需要注意 Hive 版本对 Spark 版本的兼容性, 具体对应版本在可以在下载 Hive 源码时查看 pom.xml 文件中的 spark.version, 或者参考 Hive 官网[12]. 默认 Spark 预发布的版本中有 Hive 的 jar 包, 要使用 Hive on Spark 需要去掉这些 Spark 访问 Hive 的 jar 包, 所以需要重新编译 Spark 源码. 不同的 Spark 版本编译命令有所区别, 同样参考 Hive 官网. 编译 Spark 源码需要用到 Maven, 使用 Spark 框架还需要用到 Scala, Spark 从 2.X 版本开始使用 Scala 的 2.11.X 版本. 我们使用 MySQL 数据库存储 Hive 元数据. 各资源版本如表 1 所示.

表 1 各资源版本

资源	版本
JDK	1.8
Hadoop	2.9.1
Spark	2.3.0
Scala	2.11.8
ZooKeeper	3.4.12
Hive	3.1.0
Maven	3.3.9
MySql	8.0.12

3.1 集群部署架构

Hadoop 集群和 Spark 集群搭建在 5 台虚拟机上, 虚拟机具体配置信息见表 2. 集群设置有一个主节点, 一个主节点备用节点, 进行任务管理, 三个从节点进行任务执行, 架构如图 3 所示.

表 2 服务器集群环境

服务器名称	IP 地址	操作系统	硬件环境
服务器 1	10.2.4.60	CentOS release7.3 x86_64	4CPU、8 GB 内存、120 GB 硬盘
服务器 2	10.2.4.61	CentOS release7.3 x86_64	4CPU、8 GB 内存、120 GB 硬盘
服务器 3	10.2.4.62	CentOS release7.3 x86_64	4CPU、8 GB 内存、120 GB 硬盘
服务器 4	10.2.4.63	CentOS release7.3 x86_64	4CPU、8 GB 内存、120 Gzb 硬盘
服务器 5	10.2.4.64	CentOS release7.3 x86_64	4CPU、8 GB 内存、120 GB 硬盘

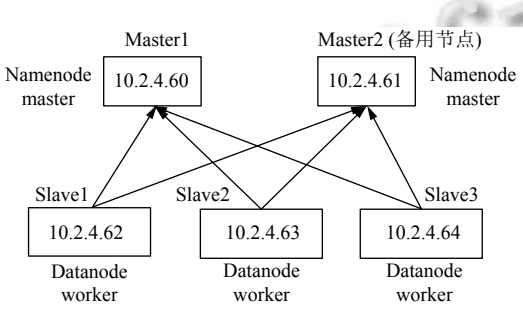

图 3 集群架构

3.2 高可用性实现

集群使用 YARN 进行资源管理, Spark 集群部署为 yarn-cluster 模式, 通过高性能协调服务 Zoo-Keeper 和 ZKFC (ZK Failover Controller process) 组件实现高可用, 具体服务部署如表 3 所示. 目前已实现 Hadoop、YARN、Spark 主备节点故障切换. 在主节点进程 Namenode (Hadoop)、Resourcemanager (YARN)、Master (Spark) 因异常退出后, 备用节点能够及时启用, 继续管理集群.

3.3 Hive 负载均衡实现与优化

使用 Hive 数据仓库有三种连接方式:

(1) Hive 的 CLI 操作方式: bin/hive.

(2) Hive JDBC 服务:

nohup bin/hive --service hiveserver2 &

bin/beeline

!connect jdbc:hive2://10.2.4.60:10000

(3) Hive 命令, bin/hive-e “HQL 语句”或者 bin/hive-

© 中国科学院软件研究所 http://www.c-s-a.org.cn

f SQL 文件.

以上连接方式, CLI 或者 Hive 命令的方式仅允许使用 HiveQL 执行查询、更新等操作, 并且比较笨拙单一. 而 HiveServer2(HS2) 支持多客户端的并发和认证, 并且允许远程客户端使用多种编程语言如 Java、Python 向 Hive 提交请求, 取回结果, 为开放 API 客户端如 JDBC、ODBC 提供了更好的支持. HS2 使得客户端可以在不启动 CLI 的情况下对 Hive 中的数据进行操作, 如可以使用 beeline 连接 HS2 执行查询. 当集群中存在多个 HS2 服务时, 用户可以自行选择具体主机进行连接, 但某台服务器连接数过大时容易造成端口不响应, 服务器故障也会造成无法查询, 使用 HAProxy 可以最大程度避免这种情况. HAProxy 是一款提供高可用性、负载均衡, 基于 TCP 和 HTTP 应用的代理软件, 并且具有代理集群状态监控功能. HAProxy 通过配置前端 (frontend) 监听端口和后端 (backend) 服务端口进行请求转发, 并提供多种负载均衡算法, 适合不同场景下的负载均衡. 当客户端向前端绑定的端口发送请求时, HAProxy 根据指定的算法选择可用的后端服务, 并将请求转发.

将 HAProxy 部署在服务器 1 上, 并将服务器 3、4、5 作为 Hive Server, 运行 HiveServer2 服务, 工作原理如图 4 所示.

表 3 集群服务部署

名称	master1	master2	slave1	slave2	slave3
IP	10.2.4.60	10.2.4.61	10.2.4.62	10.2.4.63	10.2.4.64
Hadoop	NameNode	NameNode	DataNode	DataNode	DataNode
YARN	Resource Manager	Resource Manager	NodeManager	NodeManager	NodeManager
Zookeeper	Zookeeper	Zookeeper	Zookeeper	Zookeeper	Zookeeper
Spark	Master	Master	Worker	Worker	Worker
JournalNode			JournalNode	JournalNode	JournalNode
ZKFC	ZKFC	ZKFC			

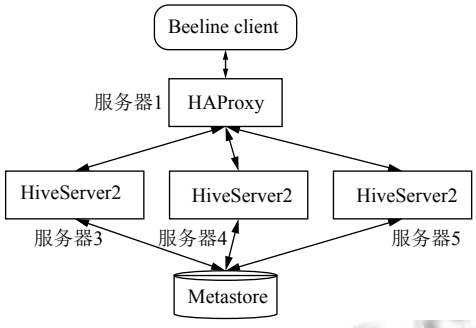

图 4 负载均衡原理

HAProxy 核心配置如下所示:

bind 0.0.0.0:25005 #HAProxy 作为代理绑定的 IP, 端口

mode tcp #在第四层进行代理服务

balance leastconn #调度算法

maxconn 1024 #最大连接数

server hive_1 10.2.4.62:10000 check inter 180000 rise

1 fall 2

server hive_2 10.2.4.63:10000 check inter 180000 rise

1 fall 2

server hive_3 10.2.4.64:10000 check inter 180000 rise

1 fall 2

将服务器 1 上的 25005 作为前端端口, 每当通过 beeline 客户端向服务器 1 上的 25005 端口发起请求时, HAProxy 通过 leastconn 算法 (最少连接数分配)[13]轮询可用的后端服务, 即轮询 hive_1、hive_2、hive_3 的 10000 端口, 10000 端口为 HS2 提供 TCP 层服务的默认端口. 服务器 3–5 上需要配置 Hive 元数据对应信息, 客户端通过 10000 端口获取元数据信息, 进而查询 Hive 数据. 集群只需对外提供服务器 1 的统一前端端口, 最终即可实现通过任意 beeline 客户端访问服务器 1 的 HAProxy 代理, 使用服务器 3–5 上的 Hive Server2 服务执行 Hive 查询. 并充分使用每个 Hive Server, 分散压力.

3.4 其他优化

(1) 资源管理器调度优化:

YARN 主要由 ResourceManager、NodeManager、

© 中国科学院软件研究所 http://www.c-s-a.org.cn

ApplicationMaster 和 Container 等几个组件构成.

ResourceManager 是 Master 上一个独立运行的进程, 负责集群统一的资源管理、调度、分配等等; NodeManager 是 Slave 上一个独立运行的进程, 负责上报节点的状态; App Master 和 Container 是运行在 Slave 上的组件, Container 是 YARN 中分配资源的一个单位, 包涵内存、CPU 等等资源, YARN 以 Container 为单位分配资源.

ResourceManager 内存资源配置, 配置的是资源调度相关:

yarn.scheduler.minimum-allocation-mb 分配给 AM 单个容器可申请的最小内存, 默认 1024 MB;

yarn.scheduler.maximum-allocation-mb 分配给 AM 单个容器可申请的最大内存, 默认 8192 MB, 由于我们所有的虚拟机都是 8 GB 内存, 需要留 2 GB 内存给操作系统, 1 GB 内存给 Hbase, 因此此处将单个 Container 内存分配上限设为 5 GB, 即 5120 MB.

NodeManager 的内存资源配置, 配置的是硬件资源相关:

yarn.nodemanager.resource.memory-mb 节点最大可用内存, 默认 8192 MB, 参考 RM 设置为 5120 MB.

yarn.nodemanager.vmem-pmem-ratio 虚拟内存率, 默认 2.1.

可以计算节点最大 Container 数量:

max(Container)=yarn.nodemanager.resource.memory-mb/yarn.scheduler.minimum-allocation-mb=5.

(2) MapReduce 调优:

MapReduce 程序优化关系到 Hive 作业的每次提交, 一些特定值的设置会较大影响到 MapReduce 任务执行效率. Map Task 和 Reduce Task 调优的一个原则就是减少数据的传输量、尽量使用内存、减少磁盘 IO 的次数、增大任务并行数, 除此之外还要根据自己集群及网络的实际情况来调优.

3.5 测试与分析

(1) 高可用性测试:

以 YARN 为例验证自动故障切换, 在 Active 节点上 kill 掉 ResourceManager 服务, 备用节点能够自动由 Standby 状态切换为 Active, 过程及结果如表 4 所示.

经过测试, Hadoop、YARN、Spark 均可以进行主备节点故障切换.

表 4 YARN 高可用测试

```
[root@master1~]# jps
31010 DFSZKFailoverController
23972 QuorumPeerMain
30709 NameNode
31160 ResourceManager
32221 Jps
13261 Master
[root@master1~]# yarn rmadmin -getServiceState rm1
active
[root@master1~]# yarn rmadmin -getServiceState rm2
standby
[root@master1~]# kill -9 31160
[root@master1~]# jps
31010 DFSZKFailoverController
23972 QuorumPeerMain
30709 NameNode
32315 Jps
13261 Master
[root@master1~]# yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /opt/workspace/hadoop-
2.9.1/logs/yarn-root-resourcemanager-master1.out
[root@master1~]#yarn rmadmin -getServiceState rm1
standby
[root@master1~]#yarn rmadmin -getServiceState rm2
active
```

(2) 负载均衡测试:

启动 3、4、5 节点的 HiveServer2 服务, 服务器 3、4、5 分别命名为 hive_1、hive_2、hive_3, 集群监控管理界面如图 5 所示, 当前状态为无客户端请求. 通过 beeline 客户端连接 HAProxy 服务器前端服务对应端口, 输入存放元数据的 MySQL 数据库账号密码, 即可成功通过 HiveServer2 服务对 Hive 中的数据进行操作, 如下所示:

```
[root@slave1 bin]# beeline
Beeline version 3.1.0 by Apache Hive
beeline>!connect jdbc:hive2://10.2.4.60:25005
Connecting to jdbc:hive2://10.2.4.60:25005
Enter username for jdbc:hive2://10.2.4.60:25005:root
Enter password for jdbc:hive2://10.2.4.60:25005:
**********
Connected to: Apache Hive (version 3.1.0)
Driver: Hive JDBC (version 3.1.0)
Transaction isolation: TRANSACTION_REPEATABLE_
READ
0: jdbc:hive2://10.2.4.60:25005&
```

当多个 beeline 客户端请求连接时, HAProxy 会自动

© 中国科学院软件研究所 http://www.c-s-a.org.cn

分配可用的 Hive Server, 如图 6 所示, 可以看到 Sessions 一栏中当前连接数 Cur 由全部为 0 变为 hive_1 为 0、hive_2 为 1、hive_3 为 1, 已按照 leastconn 分配算法成功实现请求分配, 可有效利用 Hive Server, 均衡 Hive 请求.

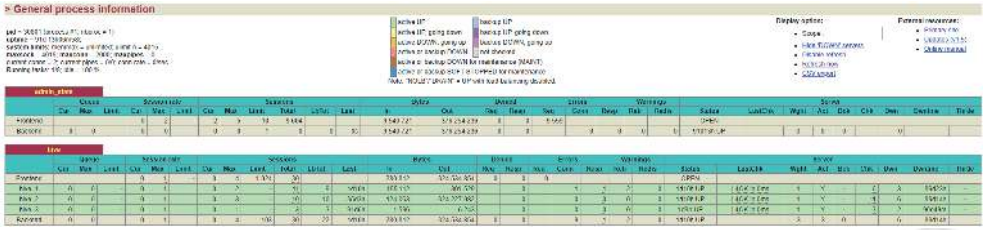

图 5 代理集群监控

Hive

	Queue			Session rate			Sessions					
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last
Frontend				0	1	-	2	4	1024	32		
Hive_1	0	0	-	0	1		0	2	-	11	9	1d10h
Hive_2	0	0	-	0	1		1	3	-	11	11	2m32s
Hive_3	0	0	-	0	1		1	1	-	4	4	12s
Backend	0	0		0	1		2	4	103	32	24	12s

图 6 多客户端请求

4 结论与展望

本文根据中国科学院教育科研态势感知服务项目现阶段实际需求, 以数据仓库的高可用性和 OLAP 为目标, 研究、设计和构建了面向科研管理大数据的 Hadoop+Spark 双引擎数据仓库, 支持对异构数据高效存储, 提供多种查询分析引擎, 为项目后续需求如数据挖掘、搭建知识图谱、学科态势分析等非实时场景提供了数据存储和计算分析平台.

由于 Hive 对事务弱支持, 且事务执行速度很慢, 存在诸多限制和不便, 不适合高并发的场景. 目前架构具有较好扩展性, 未来考虑整合 Hbase 以提升数据仓库查询实时性和对于事务更好的支持, 让平台满足更广泛的应用场景.

参考文献

1 许燕, 曾建勋. 面向科研管理的机构知识库建设政策与机制. 图书情报工作, 2015, 59(6): 22–27.

2 董成立. 谈高校科研管理及其信息管理系统. 科技管理研究, 2009, 29(5): 274–276. [doi: 10.3969/j.issn.1000-7695.2009.05.092]

3 Inmon WH. Building the data warehouse. 3rd ed. New York: Wiley, 2002.

4 于娟. 数据仓库与大数据融合的探讨. 电信科学, 2015, 31(3): 160–164.

5 吴真. 基于 Hadoop 平台构建数据仓库关键技术的研究[硕士学位论文]. 北京: 华北电力大学, 2017.

6 Floratou A, Minhas UF, Özcan F. SQL-on-Hadoop: Full circle back to shared-nothing database architectures. Proceedings of the VLDB Endowment, 2014, 7(12): 1295–1306. [doi: 10.14778/2732977]

7 何明, 常盟盟, 刘郭洋, 等. 基于 SQL-on-Hadoop 查询引擎的日志挖掘及其应用. 智能系统学报, 2017, 12(5): 717–728.

8 吴黎兵, 邱鑫, 叶璐瑶, 等. 基于 Hadoop 的 SQL 查询引擎性能研究. 华中师范大学学报 (自然科学版), 2016, 50(2): 174–182. [doi: 10.3969/j.issn.1000-1190.2016.02.003]

9 Capriolo E, Wampler D, Rutherglen J. Hive 编程指南. 曹坤, 译. 北京: 人民邮电出版社, 2013.

10 吕艳峰. Hadoop 分布式文件系统存储机制的研究与优化[硕士学位论文]. 西安: 西北大学, 2018.

11 LINBIT | DRBD HA, Disaster Recovery, Software-Defined Storage. LINBIT HA | LINBIT - High Availability with LINBIT HA. https://www.linbit.com/en/high-availability/, 2019.

12 https://cwiki.apache.org/confluence/display/Hive/Hive+on+Spark%3A+Getting+Started.

13 Cbonte. github. io. HAProxy version 1.5. 19 - Configuration Manual. https://cbonte.github.io/haproxy-dconv/1.5/configuration.html#balance. [2016-12-25, 2019-03-15].

© 中国科学院软件研究所 http://www.c-s-a.org.cn