JTangSync 分布式异构数据同步系统的设计与实现

张渊源[1],张琴燕[2,3],李 峰[4],姚祥龙[3]
(1. 浙江中医药大学 信息技术学院,浙江 杭州 310053;
2. 浙江大学 计算中心,浙江 杭州 310058;
3. 浙江大学 计算机科学与技术学院,浙江 杭州 315100;
4. 浙江省科技信息研究院,浙江 杭州 310006)

摘 要:集团企业数据往往分布在各异构环境中,但业务需求又要求企业进行数据交换和同步。由于应用平台和数据模式各不相同,传统的数据同步方式缺乏统一的同步框架。针对这一问题,设计并实现了一种分布式异构数据同步系统 JTangSync。该系统采用分布式架构,每个节点由数据源模块、数据传输模块和处理器模块等组成。每个模块都设计为可替换插件形式以便于二次扩展,各个节点依赖 Zookeeper 组成集群,具有集中管理、故障转移以及断点续传等多项功能。通过对比实验分析,证明该方法在异构数据同步方面具有较好的效果。

关键词:分布式异构数据集成;数据同步方式;JTangSync;数据传输;Zookeeper 集群
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2016)12-0169-07
doi:10.3969/j.issn.1673-629X.2016.12.037

Design and Implementation of Distributed Heterogeneous Data Synchronization Platform JTangSync

ZHANG Yuan-yuan[1], ZHANG Qin-yan[2,3], LI Feng[4], YAO Xiang-long[3]
(1. College of Information Technology, Zhejiang Chinese Medical University, Hangzhou 310053, China;
2. Computer Center, Zhejiang University, Hangzhou 310058, China;
3. College of Computer Science, Zhejiang University, Hangzhou 315100, China;
4. Institute of Scientific and Technical Information of Zhejiang Province, Hangzhou 310006, China)

Abstract: The group enterprise data is usually stored in different heterogeneous environment, but the business requirement needs data exchange and synchronization for enterprise. Due to different application platform and data model, the traditional data synchronization method lacks of an unified framework for data synchronization. To this point, a distributed heterogeneous system JTangSync is designed and implemented for data synchronization. In this distributed architecture, each node contains data source module, the data transmission module and processor module. Each module is a replaceable plug-in which is convenient for secondary extension, and each node uses Zookeeper cluster which has capacity of centralized management, failure shifting and breakpoint continuing. It is proved in experiment that the method has good effect in heterogeneous data synchronization.

Key words: distributed heterogeneous data integration; data synchronization; JTangSync; data transmission; Zookeeper clusters

0 引 言

处于起步期的企业,由于业务数量有限,数据很可能进行集中式存储。但随着业务的不断扩展,数据模式将演变成为分布式数据[1]存储于不同地区,并且容易导致数据的不一致性问题[2]。因此,需要研究一套同步系统将不同地区的数据进行有效整合,保证所有地区数据的一致性。

分布在不同地域的数据传输依赖于网络,要求整个数据同步系统必须是分布式的,并且由于业务的复杂性、地区之间业务的差异性以及人为原因等,每个地区的数据很有可能是存储在异构系统中[3]。如 A 地区数据存储在 MySQL 数据库中,而 B 地区数据存储

收稿日期:2015-10-16 **修回日期**:2016-02-24 **网络出版时间**:2016-11-21
基金项目:浙江省省级公益性技术应用研究计划(2014C33081);浙江省科技计划项目(2013F20007)
作者简介:张渊源(1980-),女,讲师,硕士,研究方向为信息系统和医疗软件中间件。

在 Oracle 数据库中,这两种方式虽有相似之处,但是许多方面是不兼容的,在实现数据同步时需要屏蔽这种差异性。然而,即使数据存储系统是一致的,也有可能出现数据库名称、存储格式等差异,使得数据同步系统设计变得非常复杂[4]。

数据同步系统应具有自动化、实时性、异构性、高可用性、高可靠性。自动化是指整个系统在运行时,基本不需要用户干预,能够自动完成数据的同步。实时性是指处理数据的及时性,从数据的产生到处理耗费的时间应该在业务可承受范围之内,分钟甚至毫秒。异构性是指同步系统不仅支持多种多样的数据源,而且还应该支持数据源的抽象和扩展。数据源包括文件系统、Web 网络、FTP 网络、数据库、ERP 系统等。高可用性是指系统应该能够 7 * 24 运行,在可用性上至少要达到 4 个 9 的目标。因为数据同步系统一般承担企业级的核心数据和业务流程,对可用性上有很高的要求。高可靠性是指在运行时可能会遇到各种突发情形(比如数据库连接断开、网络断开、IP 地址改变等环境因素)导致同步出错或者不能继续进行,数据同步系统应该能够检测同步出错,让系统在环境恢复时重新进行同步[5-6]。

工业界对同步数据问题的解决方案大致有两种,第一种是专门开发一种系统用来满足自己的业务逻辑,可以支持自身要求的数据格式,最多支持同一种数据格式系统的扩展。这一类系统以数据库增量实时同步系统为代表。第二种是基于流处理框架开发,在大数据时代传统的计算模式已经不能适应现在的大数据量和数据多样性,多种计算模式使得流计算又重新焕发了生机,并且随着云计算的普及,数据的流计算增加了许多新的特征,比如高扩展性、分布式支持等等[7]。由于数据同步实际上是流处理的一种,用户在流处理之上开发数据同步需要自己解决断点续传、增量数据获取、分布式等难题,所以这种方法虽然扩展性更高,但是开发成本也更高[7-10]。

基于此,文中设计了一种异构数据同步系统 JTangSync。与传统方法相比,用户基于 JTangSync 提供的接口可以定义自己的同步逻辑,然后快速开发期望数据格式的系统,并且不需要考虑断点续传、故障恢复等问题[11],此外监控管理平台让用户省去开发自己的监控管理平台。JTangSync 专门解决数据同步问题,尤其是数据库同步的问题。设计了大量的异构数据源和处理程序,用户开发一般的数据同步只需按照指导在监控管理平台进行配置然后启动即可。

1 相关技术

数据同步系统设计和实现的相关技术,包括 Zookeeper,Protocol Buffer,Snappy,Netty,Jetty,Fast JSON 等等。

1.1 Zookeeper

JTangSync 使用 Zookeeper 来管理配置信息,包括所有的数据源配置、数据处理器配置、监控实例配置、节点信息和 Manager 的选举[12]。

ZooKeeper 源于 Hadoop 的一个子项目,是一个应用于大型分布式系统的协调系统,功能包括分布式同步、配置维护、名字服务等。ZooKeeper 封装复杂易出错的服务,提供可靠易用的接口和高效稳定的系统。一个分布式的环境需要 Master 实例来存储一些配置信息,使用 Zookeeper 来维护信息和配置信息等,确保写入文件的一致性。

1.2 Netty

JTangSync 使 Netty 作为其核心组件,实现分布式的网络程序。Netty 是一个著名的开源 NIO 框架,主要功能是提供异步的、事件驱动的网络应用程序接口,使用 Netty 能够快速开发高性能、高可靠性的网络服务器和客户端程序[13]。

使用 Netty 可以确保网络应用的开发过程更加简单和快速,例如实现了 HTTP 协议的客户和服务端应用。Netty 大大简化和流线化了网络应用的编程开发过程,例如 TCP 和 UDP 的 Socket 服务开发。在设计上 Netty 借鉴各协议的经验,包括 FTP、SMTP、HTTP、二进制、文本协议等。

1.3 Protocol Buffer

JTangSync 设计了网络数据的交换过程,如数据传输、节点间通信、性能监控等等。由于 Protocol Buffer 在 Netty 中存在相关的编码器和解码器,使用 Protocol Buffer 作为 JTangSync 中的数据标准的交换格式[14]。

Protocol Buffer 是 Google 的一种数据交换格式,它独立于语言和平台,提供了三种实现语言:Java、C++和 Python。每一种实现过程都包含了相应语言的编译器以及库文件。由于它是二进制的格式,比使用 XML 进行数据传输更快,因此可以把它用于分布式应用之间的数据通信或者异构环境下的数据交换。作为一种效率和兼容性都很优秀的二进制数据传输格式,可以广泛运用于诸如网络传输、配置文件、数据存储等诸多领域之中。

1.4 Snappy

为提高效率,JTangSync 中网络传输的数据需要压缩,但是传统的压缩算法耗时较长。该系统使用 Snappy 算法作为网络传输的标准压缩算法。Snappy 是 Zippy 的开源实现。它作为一个压缩库被 Google 用于许多内部项目中,包括 MapReduce,RPC 和 BigTable。Snappy 优化了 64 位 x86 处理器,经测试,Snappy 在单

个 Intel Core i7 处理器内核上压缩速度为每秒 250 MB,解压速率为每秒 500 MB。

1.5 Fast JSON

JTangSync 使用 JSON 描述数据交换格式。JSON 是基于 JavaScript 的一个子集,采用完全独立于语言的文本格式,使得 JSON 成为理想的数据交换语言,既易于人类阅读和编写,又易于机器解析和生成。

Fast JSON 是阿里巴巴公司开发的一个 JSON 开发库[15]。与传统的 JSON 解析相比,Fast JSON 的解析速度更快,并且 Fast JSON 的 API 设计比较合理,包含多种 Java 对象与 JSON 转化的方法。其中,最大的优点在于可以与 Java Bean 进行相互转化。在 JTangSync 中,Fast JSON 主要用于 Web 程序的数据交互、集群间节点的性能数据交换等方面。

2 数据同步系统设计

同步系统是将数据从一个系统、软件或者文件等存储中提取出来并且通过网络连接、邮件等方式传输到另一个不同或者相同的系统中,最终将数据导入到目标系统的存储中的自动化系统。

如图 1 所示,节点中运行的功能单位是同步实例,代表一个从数据源到数据处理器的完整逻辑。一个同步实例包含一个数据源、一个由 0 到多个数据处理器组成的数据源处理器链、一个由 0 到多个数据处理器组成的本地处理器链。

本节将分别从数据源模块、网络传输模块、数据处理模块和集群模块来介绍 JTangSync 异构数据同步系统。

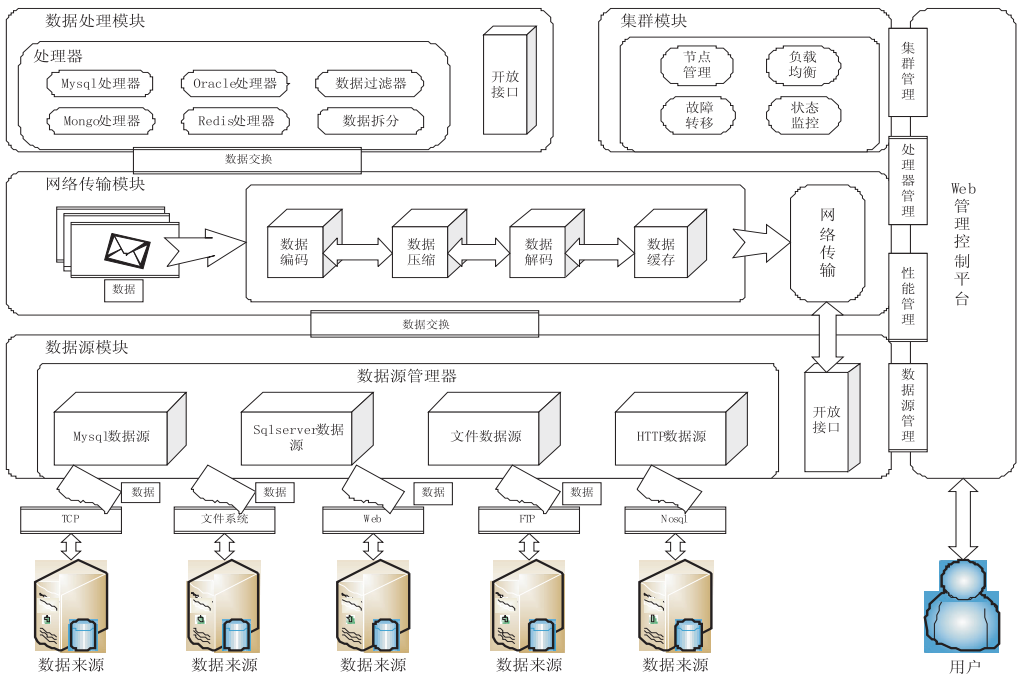

图 1　系统拓扑结构

2.1 数据源模块

数据源是指 JTangSync 获取数据源的一种代理器。由于一般的同步系统中数据源与处理逻辑并不在同一个域中,安全性要求数据源从外界进行访问时必须通过代理来获取数据。此外,数据来源多种多样,如 MySQL 数据源、SQL Server 数据源、Oracle 数据源、文件数据源等。数据源的主要职责就是获取数据并且通过网络转发出去。数据源模块就是管理数据的功能模块。

2.2 数据源工作流程

数据源的主要职责就是获取数据并且转发到数据传输模块。数据源管理器管理所有的数据源,数据源管理器接收到初始化数据,然后查找是否存在此类型的数据源。如果找到,则初始化数据源,数据源进入工作状态。在此过程中,如果发生任何异常或错误,数据源退出。最后重新初始化数据源是同步实例的职责,而不是数据源的职责。

2.3 网络传输模块

网络传输模块的主要职责是将数据尽量及时转发出去。实现难点包括延迟低、数据速率高。延迟低是指数据到达网络传输模块之后,需要尽快地传输出去,以及时将数据提交到对端。

数据速率高是指网络传输的速度要尽量快。当数据包比较小需要合并时,这两者是冲突的,因为由于 IO 接口只能支持有限的调用次数(现今大约为数百次/s),所以必须在每次调用 IO 接口时传输尽量多的数据。由于数据包较小,那么每秒发送的数据量会很小。因此,必须进行缓存。

2.4 数据传输流程

数据传输模块处理数据源模块产生的数据。由于 IO 设备的限制,每秒的 IO 次数是一定的,数据要先进入缓存。如果数据量达到或超过限制,或者超时(超时限制设置为 200 ms),将数据序列化方便网络传输[16]。序列化方法要求序列化和反序列化的速度要快,不能影响网络传输的效率。文中以 Protocol Buffer 作为序列化技术。

接着分析当前的数据是否需要压缩,判断策略主要通过预测算法,通过过去的数据预测当前时间点的情况,如果过去 1 小时之内使用压缩传输数据的平均速度为 m kB/s,不使用压缩传输数据的平均速度为 M kB/s,那么有以下策略:

$$f(x)=\begin{cases}\text{压缩}, m > M\\ \text{不压缩}, m \leqslant M\end{cases}$$

上式的意义为,当 $m > M$,即压缩的数据传输速度大于不压缩,则将当前的压缩模式切换为非压缩模式。由于 Snappy 算法在压缩和解压缩方面速度远胜于 GZIP 库,所以 JTangSync 在网络传输模块中采用 Snappy 算法。

2.5 数据处理模块

数据模块是管理和配置、执行数据处理器的模块。用户的数据处理需求比较复杂,单一的数据处理器不能满足用户的全部需求,采用责任链模式来定义用户要求的数据处理逻辑。此外,如果 JTangSync 中提供的数据处理器不满足用户的特殊需求,用户可以通过 JTangSync 数据处理开发接口开发自己的数据处理器。

2.6 JTangSync 数据处理流程

数据处理模块是处理数据的主要模块,JTangSync 中处理数据的一个功能单元模块,又称 Handler。每一种数据处理逻辑在 JTangSync 中都对应一个数据处理器,多个数据处理器可以配置为一个数据处理器链,数据处理器链是单向并且不可循环的。其中数据处理器是数据处理模块的组成单元。当配置一个同步实例时,用户需要配置若干数据处理器,这些数据处理器按照配置的先后顺序组成一个责任链来处理这一个同步实例的数据。

数据处理器在使用之前需要初始化,这一过程可能需要配置信息,然后数据处理器开始处理数据。初始化所需配置信息统一为 Key-Value 格式,配置信息数据类型为 String。

数据源处理器链由 0 到多个数据处理器组成,数据源会将数据提交给处理器链然后进行网络传输。本地处理器链由 0 到多个数据处理器组成,可以从网络接收数据并且进行处理。为与数据源处理器链相区别,称为本地处理器链。一个同步实例中数据处理器需要在两个阶段运行,第一个阶段在网络传输之前进行,主要进行数据过滤、合并、统计等,称为数据源处理器。第二个阶段是在网络传输之后进行,主要责任是处理数据,称为本地处理器。两个阶段的处理器是共用的,接口设计完全相同,体现了 JTangSync 的高度灵活性。

2.7 JTangSync 集群模块

集群模块是管理和控制 JTangSync 集群的模块,主要功能有节点管理、负载均衡、故障转移和状态监控、Manager 选举、集群间通信、本地同步实例管理等。在管理控制平台,用户可以实现 JTangSync 集群管理、数据源管理、性能监控、处理器管理等所有目标。

2.8 JTangSync 集群流程

JTangSync 设计为集群模式,多个 JTangSync 节点通过 JTangSync 集群模块可以组成一个 JTangSync 集群。节点指一个 JTangSync 进程,是指 JTangSync 集群中的一个资源单位,一个 JTangSync 集群中可能包含 1 到多个 JTangSync 节点,同步实例可以运行在其中任何一个节点上。Node 的定义与节点定义相同。JTangSync 集群模块包含 JTangSync 集群运行的必需功能。

文中使用 Zookeeper 实现 JTangSync 集群模块,提供一致的分布式事件回调接口,使用方便。树形目录结构与当前系统保持一致,并且树形结构目录的每个节点都提供服务,这满足了分布式系统中的配置信息保存的需求[17-18]。

3 JTangSync 节点实现

3.1 数据源的实现

数据源是 JTangSync 中的核心组件,数据源的质量和数量决定了 JTangSync 是否强大。数据库增量数据源是 JTangSync 核心中的核心,是指当前提取的数据都是数据库在一段时间之内的增量数据,与数据库数据源是不一样的概念。数据库数据源只是简单的连接数据库,然后获取其中的全量数据,实现的难度较数据库增量数据源低。数据库增量数据源的实现难度大,对效率、延迟的要求都会比其他的数据源高。

MySQL 数据源实现是基于 MySQL 本身的 Replication 协议,Replication 是专门用于 MySQL 之间复制的协议。Replication 协议中需要同步数据的 MySQL 称为 Slave,提供增量数据的称为 Master,数据传输基于 Slave 和 Master 之间的 TCP 连接,Master 中增量数据存储的位置是 binary log(二进制日志),存储格式有三种:statement,row,mixed。在 JTangSync 中选择 Row 格式作为 MySQL 增量数据源的二进制日志格式。

Sqlserver 的增量数据源使用 Sqlserver 的变更数据捕获功能(又称为 Change Data Capture,CDC),Sqlserv-

er 的变更数据捕获功能有一些限制。首先 CDC 有版本限制,只有 SQL server2008 以及以上版本才支持。其次 CDC 功能需要为每个表开启一个临时表。尽管存在这些限制,比起触发器方法还是有很多优势。

3.2　数据传输的实现

数据传输基于 Netty 实现。Netty 不仅是优秀的 NIO 和多线程框架,而且提供了许多协议库,其中包括 snappy 和 protocol buffer 的实现。数据传输代码如下:

```
ChannelPipeline pipeline=ch.pipeline();
ch.pipeline().addLast(new MySnappyDecoder());
ch.pipeline().addLast(new MySnappyEncoder());
ch.pipeline().addLast(new ProtobufVarint32FrameDecoder());
ch.pipeline().addLast(new ProtobufDecoder(SyncEvent.Event.getDefaultInstance()));
ch.pipeline().addLast(new ProtobufVarint32LengthFieldPrerpender());
ch.pipeline().addLast(new ProtobufEncoder());
pipeline.addLast("handler", new SnappyProtobufServerHandler(ch));
```

自适应压缩算法的主要思想是在 Netty 中实现一个专门用于统计网络传输字节数的的 handler,并且向上层提供接口,上层逻辑可以使用这个接口实现自适应压缩算法。例如,此 handler 统计到已经发送了 n_1 字节,经过 t s 之后变成 n_2 字节,t s 内的发送速度为 $(n_2-n_1)/t/1\ 024$,单位为 kB/s,相比之前的速度即可决定是否需要进行压缩算法。

3.3　数据处理器的实现

过滤处理器是指根据一定规则过滤数据的处理器,这里的规则主要有相等、不等、大于、小于、大于等于、小于等于等规则。过滤处理器使用的范围是在数据库同步领域,比如当传输数据库数据时,用户可能需要过滤掉多余数据。

数据库处理器是将数据导入数据库的处理器。由于不同的数据库的 SQL 标准是不一样的,所以每个数据库对应一个数据库处理器。数据库处理器需要考虑:首先,如果是关系型数据库,那么要将数据的提交格式设置为 false 并且执行一定数量的 SQL 之后提交;如果是非关系型数据库,那么要尽量减少 IO 的次数。其次,数据传输时要尽量打包,减少网络 IO 的次数。最后,如果通过 JDBC 执行 SQL,需要使用 prepared statement,因为数据只有 insert、delete、update 三种,可以将三个 prepared statement 缓存起来,等待重复使用。实验证明,性能至少提高了 10% ~20%。

4　JTangSync 集群实现

JTangSync 集群中存在多个节点,需要一个协调者担任 Manager 的角色。这个 Manager 节点的主要任务是启动 Web 监控管理平台和协调集群运行。

4.1　Manager 选举算法

JTangSync 中选举算法的流程如图 2 所示。

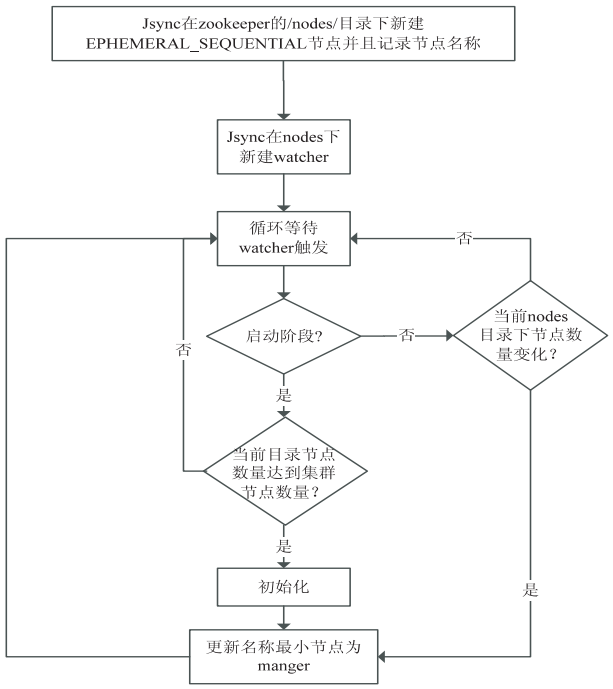

图 2　JTangSync 中选举算法的流程图

1)JTangSync 在 zookeeper 目录中的//nodes/目录下新建一个类型为 EPHEMERAL_SEQUENTIAL 的 node,它是 zookeeper 中的顺序内存 node。SEQUENTIAL 保证了创建此 node 的目录下所有的 SEQUENTIAL 类型的 node 名称都是顺序的,名称由 zookeeper 决定;EPHEMERAL 类型 node 保证了此节点只存在于连接有效的时间段内,如果创建此 node 的连接失效,那么 zookeeper 将自动把此 node 删除。EPHEMERAL_SEQUENTIAL 类型就是以上两种类型的集合。

2)JTangSync 在/nodes 目录下新建一个 Watcher。Watcher 是 zookeeper 中的一种分布式事件通知器,如果 Watcher 所在的 node 的数据发生变化(包括删除、修改)或者 node 下的节点发生变化(包括删除、新建),Watcher 就会得到通知,并且执行 Watcher 中的回调方法。

3)如果当前 Watcher 被调用,那么分两种情况:

如果 nodes 下节点数量达到配置中 JTangSync 集群中节点的数量并且当前还未启动完成,那么将获取当前 nodes 下所有的节点并且排序,名称最小的节点将成为 manager 节点。Manager 节点将会完成以下操作:

(1)启动当前节点的 web 监控管理控制平台。

(2)分配所有标志为"自动启动"的同步实例到 JTangSync 节点上运行。

如果 nodes 下节点数量没有达到 JTangSync 集群中节点的数量,并且当前还未启动完成,那么就返回。

如果当前 JTangSync 集群标志为已启动,并且当前 nodes 下的 node 数量有变化,那么更新 manager,新的 manager 需要完成以下操作:

(1)启动当前节点的 web 监控管理控制平台。

(2)启动故障转移模块。

JTangSync 的选举非常灵活,利用 zookeeper 不仅能够快速准确地选举出 manager 节点,并且利用 zookeeper 实现了在过去 manager 节点失效的情况下选举新的 manager,实现了分布式系统中的高可靠性。

4.2 节点热部署

利用 Zookeeper,还可以实现在运行时新增节点。新增节点必须在整个 JTangSync 集群已经启动的情况下进行(如图 3 所示)。

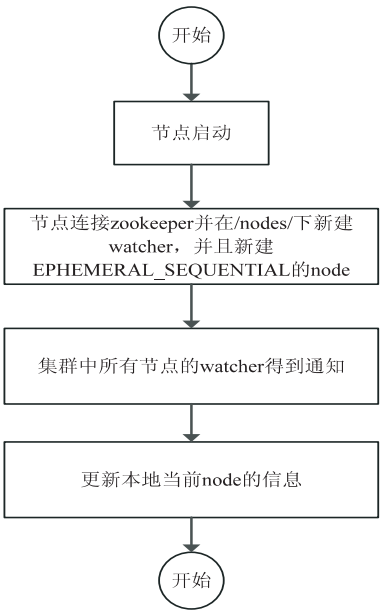

图 3 增节点的流程

从图 3 中看到,节点的热部署[19-20]大致分为 2 步:

(1)启动,并且连接 Zookeeper,与选举算法中的描述相同,在 nodes 下新建 watcher,在 nodes 下新建 EPHEMERALSEQUENTIAL 类型 node。

(2)此时,其他 watcher 都会收到通知,并且更新本地所有的 node 信息。

4.3 节点热移除

借助 Zookeeper 的分布式通知功能,可以实现 JTangSync 节点的热移除功能。JTangSync 集群中节点的移除流程如图 4 所示。

由于节点在 Zookeeper 中建立的是 EPHEMERAL 类型的节点,停止之后,zookeeper 中建立的节点消失,/nodes/节点下 node 数量变化,其他的 JTangSync 节点都会收到通知,最后更新 manager 和本地的 node 信息。

5 系统性能测试

5.1 操作类型的支持

测试 JTangSync 在 MySQL 数据库 insert,update,delete 语句。测试数据共 426 570 条,包括 146 570 条插入数据,140 000 条删除数据,以及 140 000 条更新数据。测试环境为内网环境。测试见表 1。

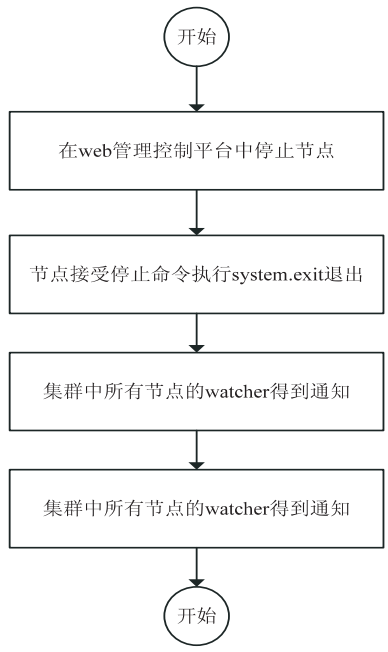

图 4 节点热移除的流程

表 1 操作类型测试

测试类型	花费时间/s	性能/(条/s)
insert	20	7 328
delete	20.3	6 896
update	22	6 363

5.2 模块性能测试

测试 JTangSync 的数据源的数据捕获性能、数据发送模块性能、数据处理器性能和内存占用。测试配置从 Master 到 Slave,同步 426 570 条数据。Slave 自身性能测试先使用独立的 20 万条插入数据,之后与整个系统的性能比较,得出是否已经达到了整个系统的瓶颈。

性能测试结果如表 2 所示。

表 2 性能测试结果

测试项	测试结果
数据源模块/(条/s)	42 530
数据发送模块/(条/s)	16 500
数据处理模块/(条/s)	6 874
内存占用/MB	163
Slave 本身性能/(条/s)	7 104

数据传输模块的瓶颈在 16 000 条/s 左右,与数据源性能相差如此大的原因是因为需要将数据源模块获

取的数据转化为 JTangSync 的统一数据格式,这耗费了很多时间。另一方面是 JTangSync 已经达到了整个系统的性能瓶颈,JTangSync 的性能最高达到 6 874,Salve 的性能瓶颈在 7 104,除去 JTangSync 在数据传输、压缩、转换的消耗,JTangSync 在资源的利用和性能方面做的比较好。

6　结束语

JTangSync 是异构数据同步框架,主要功能是为用户提供一个高可用、高性能、高可扩展的数据同步开发平台。用户使用这个框架无需关心数据同步中常见的数据格式、数据传输、断点续传、故障恢复等问题,只需要考虑基本的业务逻辑和数据源、数据处理器的开发即可。JTangSync 内置了大量的数据源和数据处理器,免去用户重复开发一些常用数据源和数据处理器的烦恼。从测试结果上看,JTangSync 是一个优秀的异构数据同步框架。

未来,由于数据同步的需求多种多样,JTangSync 将整合用户的各种需求,开发新的数据源和数据处理器,并加入新的数据同步逻辑。

参考文献:

[1] Leslie B,Ron B,Charles D L T,et al. Formulary and database synchronization[J]. American Journal of Health-System Pharmacy,2011,68(3):204-206.

[2] 潘群华,吴秋云,陈宏盛. 分布式数据库系统中数据一致性维护方法[J]. 计算机工程,2002,28(9):255-257.

[3] 戴婉荣. 基于分布式数据库的数据同步机制的研究与应用[D]. 武汉:武汉理工大学,2010.

[4] Liao Guoqiong. Data synchronization and resynchronization for heterogeneous databases replication in middleware-based architecture[J]. Journal of Networks,2012,7(1):210-217.

[5] 魏祥麟,陈　鸣,范建华,等. 数据中心网络的体系结构[J]. 软件学报,2013,24(2):295-316.

[6] 章　筠. 计算机网络可靠性分析与设计[D]. 杭州:浙江大学,2013.

[7] 蔡　亮,刘　腾. 基于写操作集的数据库同步复制模型[J]. 计算机工程,2011,37(13):61-62.

[8] 秦　森,杨　艳. 基于 Oracle 日志分析的数据还原操作的设计及实现[J]. 电脑知识与技术:学术交流,2007,1(3):626-628.

[9] 纪　彬,贺　立,白广利,等. 基于 JMS 的数据交换技术研究与实现[J]. 自动化技术与应用,2011,30(2):70-73.

[10] 赵　莉,杜思锋. 数据交换平台中异构数据转换技术的研究[J]. 电子设计工程,2011,19(5):91-93.

[11] Hossain M S,Masud M,Muhammad G,et al. Automated and user involved data synchronization in collaborative e-health environments[J]. Computers in Human Behavior,2014,30(1):485-490.

[12] 孙大为,张广艳,郑纬民. 大数据流式计算:关键技术及系统实例[J]. 软件学报,2014,25(4):839-862.

[13] 金志国,李　炜. 基于 Netty 的 HTTP 客户端的设计与实现[J]. 电信工程技术与标准化,2014,27(1):84-88.

[14] 窦进业. 基于 Protocol Buffers 的企业即时通讯应用的研究与实现[D]. 青岛:中国海洋大学,2014.

[15] 王　凯,赵庚申,刘国良. 基于 Android 的电能管理系统的研究[J]. 南开大学学报:自然科学版,2015,48(2):69-74.

[16] 陈增强,郭嘉琳,刘忠信,等. 具有断点续传功能的文件传输系统的设计与关键技术[J]. 计算机工程,2002,28(12):14-16.

[17] 王　宾. Hadoop 集群的部署与管理系统的设计与实现[D]. 南京:南京大学,2013.

[18] 张　璇. 面向移动计算的分布式文件共享服务平台软件研究[D]. 杭州:浙江大学,2010.

[19] 李海骋,曹　春,吕　军,等. 支持依赖修复的热部署技术[J]. 计算机科学,2014,41(11):141-145.

[20] 金　海,官象山,吴　松,等. 分布式存储系统中文件传输优化的设计与实现[J]. 华中科技大学学报:自然科学版,2005,33(1):4-6.