基于 SpringBatch 框架的数据批量处理在决策分析系统中的应用

徐 杨 周文彬
(河海大学计算机与信息学院 南京 210098)

摘 要：通常企业决策分析系统需要对关键业务信息进行批处理操作，而这些业务的批处理操作是在没有用户进行交互的情况下对大量复杂的信息进行自动处理。为了解决批处理数据操作效率低下的问题，本文在对 SpringBatch 框架进行深入的研究和分析的基础上，选用 SpringBatch 框架，开源作业调度库 Quartz 以及关系型数据库 MySQL 和非关系型数据库 Cassandra，设计并实现了高效批处理的决策分析系统。同时本文还对该系统进行相关性能测试，测试结果证明使用 SpringBatch 框架的决策分析系统批数据处理能力得到了极大的提升。

关键词：SpringBatch；批量处理，决策分析系统
中图分类号：TP202 **文献标识码**：A **国家标准学科分类代码**：510.1050

Application of data batch processing in decision analysis system based on SpringBatch framework

Xu Yang Zhou Wenbin
(Department of Computer and Information, Hohai University, Nanjing 210098, China)

Abstract: Usually the enterprise decision analysis system needs to deal with the key business information. These operations are automaticly deal with lots of complex information without user interaction. In order to solve the problem of low efficiency of batch data operation, in this paper, based on in-depth research and analysis on the Spring Batch framework, we use SpringBatch framework, open source job scheduling Quartz and relational database MySQL and non relational database Cassandra to design and implement the decision analysis system of efficient batch processing. At the same time, this paper also carry out the performance test of the system, test results show that the decision analysis system which used the SpringBatch framework to implement is more efficient in batch data processing.

Keywords: SpringBatch; data batch processing; decision analysis system

1 引 言

随着计算机技术，信息技术与分布式计算环境为代表的高新技术不断发展，各个行业产生了大量的数据。Spring Batch[1-5]就是为了解决不同数据之间的批量处理而诞生的。SpringBatch 可以构建出轻量级的健壮的并行处理应用，支持事务、并发、流程、监控、纵向和横向扩展，提供统一的接口管理和任务管理。SpringBatch 以 POJO 和 Spring 框架为基础，并提供可重复使用的处理大量记录[6-8]的功能，包括必不可少的日志记录、跟踪，事务管理，作业处理统计数据，工作重新启动，资源管理等。

本文的决策分析系统中主要是以 Spring[9]和 Mybatis 框架为基础的 web 应用项目，需要从一个或多个数据库抽取数据，经过处理后再写入另一个或多个数据库，所以在 web 系统中 SpringBatch 也用于对大量数据进行并行处理。对此以上特性，本文在交易会员决策分析系统中的数据抽取模块使用了 SpringBatch 框架，用以提升不同数据库之间数据的批量处理[10-11]速率，为企业的数据整合和面向全局的数据分析工作提供了方便。

2 数据批处理模块需求分析

2.1 数据批量处理需求分析

会员决策系统主要功能结构如图 1 所示。图 1 中会员单位实现的所有的决策分析中的功能都是以当前的数据作为依托。而会员决策系统应包含两个部分，分为交易所端和会员端两个独立的子系统，并拥有各自独立的功能与结构。

收稿日期：2015-08

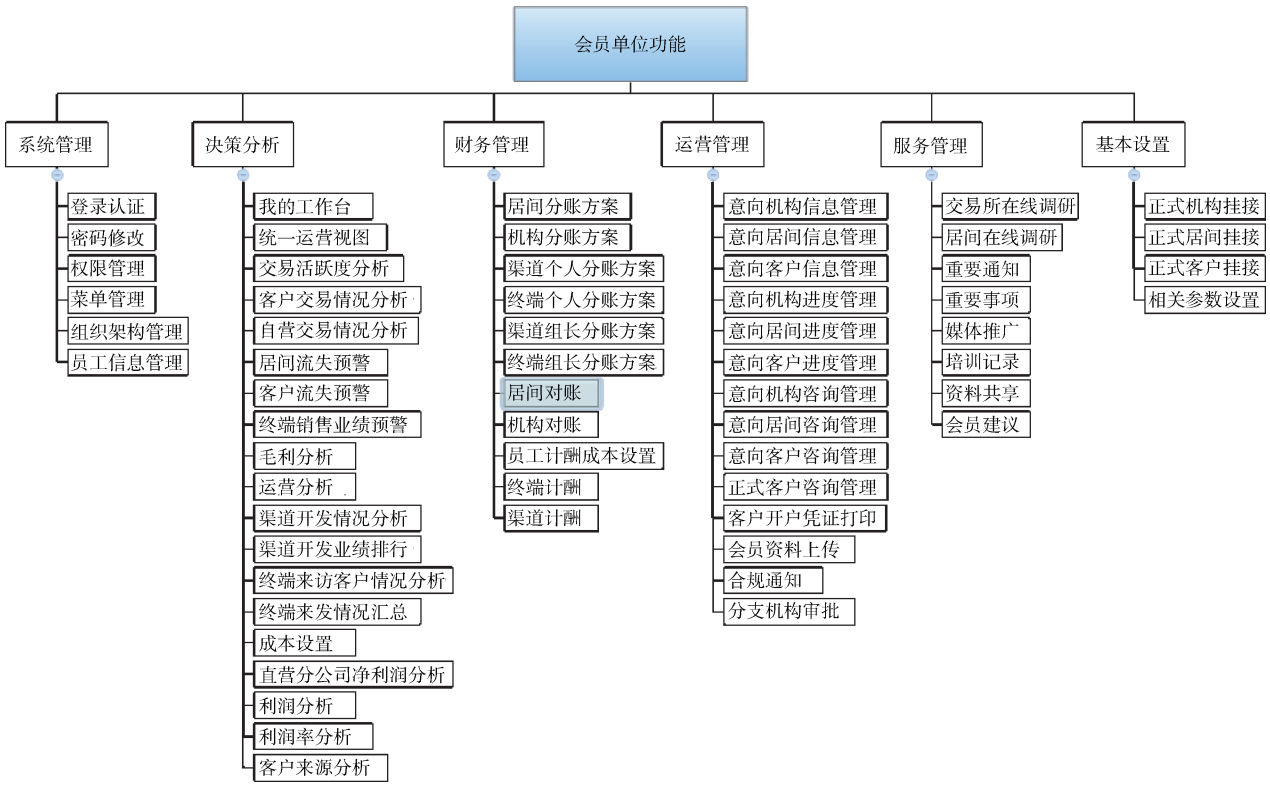

图 1　会员单位系统的全部功能图

决策分析系统的交易所的数据源繁多，在进行数据抽取时，需要批量从多个数据源中同时抽取数据加以整合。会员决策分析系统的部署是一个巨大的树形结构。有多家会员单位挂载在交易所下，又有多家居间用户挂载在会员单位下。分析决策交易所包含有一个自己的数据库，用来存放从实盘、模拟盘、网上开户、监控平台和 Cassandra[12] 数据库中抽取整合后的数据，在决策分析交易所下的每一个家会员单位都有其自己的数据库且彼此之间是相互独立的，每一家会员单位的数据库相互独立，数据不共享，每一家会员单位会根据其自身权限的不同从交易所中获取其会员单位与权限下对应的数据[13]。交易所端享有其下所有会员单位的所有数据权限。这样所有的数据操作，都是在目标数据库上进行，不会对源数据库造成影响；而由于目标数据库数据是以源数据库数据为依托，从整体角度而言，在目标数据库所在的服务器上进行数据集成与转换等操作，一旦有业务数据方面问题，也可避免更大损失，因而需要在目标数据库[14]端进行数据批处理操作。

2.2　Springbatch 与传统批量处理方式对比

用户或者系统加载大量数据或进行大批量数据读取写入时，系统通常会采取批量处理的策略，以增加用户体验、提高系统的处理速度、减少处理时间。传统的数据获取一般有两种方式：方式一，一次性读取所有数据缓存在客户端，再一次性写入的方式，这种方式在读取与写入是异步进行的，在读取与写入中读取模块与写入模块各自有一半的时间是处于闲置状态；方式二，每次固定地读取部分数进行读取，当这一批出具处理完毕后，在将这一批数据继续进行写入。该文基于 Spring 组件中 Springbatch 技术，综合两种处理方式优点，是一种近年来新的批量处理模型，其可以根据用户对响应时间的要求，用户自行决定每个批次处理数据的多少，并在系统处理当前数据的同时，自动取得后面批次的数据，以减少系统空闲的时间。

3 种处理方式比较如图 2 所示。从图 2 中不难看出，方式一在系统开始读取数据到系统数据写入完毕系统等待时间较长，方式二每处理一批数据都要等待一段时间，相比方式 1，方式二总的处理时间与方式一相等，但在用户需要数据进行交互时，可以显著的提高系统的相应度，提高用户体验，而 SpringBatch 方法在等待一个相对较短的首次读取后可以连续处理读取数据，在处理或者写入数据是，读取模块不会空闲，而是会继续读取下一批数据。

3　数据批量处理模块设计实现

3.1　项目环境搭建

本文所述基于 SpringBatch 框架的数据批量处理在决策分析系统中的应用是以 SpringBatch 框架在会员决策分析系统中的针对交易所端与会员端。

在交易所端我们采用了 IBM 服务器 X3650M4-79159Z1 服务器。会员分析决策系统的运行环境如下：应

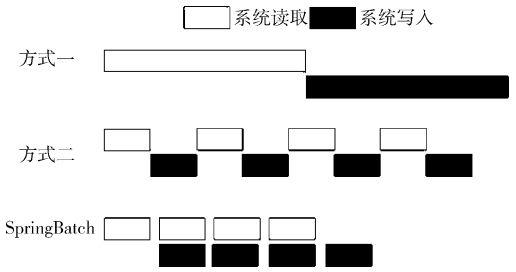

图 2　几种批处理方式比较

用服务器 Apache Tomcat7.0 以上版本;JavaJDK1.6 以上;数据库 MySQL5.1 以上版本;Meaven3.0.5 及以上;Mybatis3.0 及以上;Spring3.2.5 及以上;SpringBatch 版本 2.2.7 及以上访问终端 IE、Firefox、Chrome、Safari 最新发行版。整个会员决策分析系统是建立在 Spring3.2.5 版本基础之上。在不改动原有 Spring 配置和 java 文件的基础上,我们可以直接加入 Springbatch 模块。在导入了 SpringBatch 所有的 jar 后,就可以正常的开发,不需要做额外的配置工作。

3.2　数据批量处理模块

模块化就是把一个复杂的系统分解为若干个规模较小、功能较简单的、相对独立、更易于建立和修改的模块,分别加以设计实现,各模块在一定关系的约束下共同构成一个统一的整体,完成系统的功能。采用模块化设计思想,数据批量处理功能可分为两个模块:交易所端模块和会员端模块。由于数据库中包含关系数据库及非关系型数据库的多个数据库中的多个数据表信息,含有数据定义、查询、更新、控制等。

3.2.1　任务启动器

为了启动 Springbath 中的不同功能的 Job,首先需要一个公用的 JobLauncher 作为所有 Job 的启动器,外部控制器会调用 JobLauncher 去 Job 执行各自的任务,为了简便开发,减少配置的负责程度,仅创建一个公用的 JobLauncher,在外部控制器即 Quartz 触发器的配置文件中调用时会增加 job 名称属性,使 JobLauncher 可以确定具体去调用那一个 Job 任务。

在任务启动器类 JobLauncher 类中必须导入 org.springframework.batch.core 包中的 JobParametersBuilder、job 以及 JobLauncher,这些类来自 Springbatch 的核心类,为 Springbatch 提供必要的功能。需要注意的在申明 job 和 JobLauncher 时需要使用 transient 为了在一个特定对象的一个域上关闭被持久化,必须在这个域前加上关键字 transient。在 execute 函数中计算时间和会员单位相关参数,在 JobLauncher 启动前将相关参数放进 jobBuilder 并转变为 jobParameters,最后在执行 jobLauncher 的 run 方法时,将带有参数的 jobParameters 传入。jobLauncher 负责 batch 的启动工作,jobRepository 负责 job 的整个运行过程中 CRUD 操作,transactionManager 负责事务的管理操作。这些信息都需要在 Springbatch.xml 中进行配置。

在 jobLauncher 的配置文件中必须要配置的是 jobRepository 和 gnnt_transactionManager。job 仓库是 springbatch 底层基础的一个关键特征,它为 springbatch 的运行提供信息。job 仓库必须实现 JobRepository 接口,spring batch 只提供了一个具体的实现类:SimpleJobRepository。因为内存中的东西会丢失。这里采用 jdbc 元数据的持久化方法来配置 job repository。

在配置 jobLauncher 时引用的属性是 jobRepository 配置的 id 名,jobRepository 中的 transactionManager 属性是下面配置的 transactionManager 的 id 名称,这样 jobLauncher 启动是才能找到 jobRepository 与 transactionManager。那么这样一个公用的 job 启动器就配置完成了,下面就根据具体的业务创建 job 的 ItemReader、ItemProcessor、ItemWriter 并在配置文件中添加对应的配置。

3.2.2　交易所端模块

会员决策分析系统的交易所模块即数据处理模块,它主要包括两个子模块,其一是数据抽取分析处理模块,另一个是数据上传模块。数据抽取分析处理模块中的数据抽取即是从多个数据源中定时的批量的大数据量的抽取数据。数据抽取从多个数据库中的多个表中抽取必要的准备数据。

由于交易所数据库的特殊安全性以及诸多会员单位的分布考虑,采用独立的数据分发抽取的模块来解决这个问题。数据分发和抽取模块先将需要处理的数据,从交易所端的数据中抽取出来,在模块汇总,根据具体的业务需求进行处理,然后将整理后的数据上传至 ftp 服务器后,由会员端系统将数据下载保存至其自己的数据库为了在各个不同是数据库中批量处理、导入和导出数据,选用了 SpringBatch 框架来解决大数据量的数据批量处理的问题。

数据批量处理模块是决策分析系统交易所系统中的一个负责从 5 个数据源抽取和整合数据的并将经过整合处理的数据保存至交易所自身的数据库。同时,数据批量处理模块会将数据根据会员单位的权限和名称加以分类,上传至 ftp 文件服务器。由于交易的实时性原则,每天都有新的交易和注册用户,所以批量处理模块每天会在交易所结算过后进行一次批量的数据抽取和数据处理,整理后的数据被已 txt 文本文件的格式存储并根据抽取数据当天的日期命名。为了确保每日抽取的数据的准确性,防止某个会员单位的数据被遗漏或者由于服务器问题导致 SpringBatch 任务中途结束数据没有抽取处理完整,我们设计在每一个会员单位的数据抽取处理完毕后生成一个同样名称以.ok 结尾的确认文件,在整个流程结束后数据上

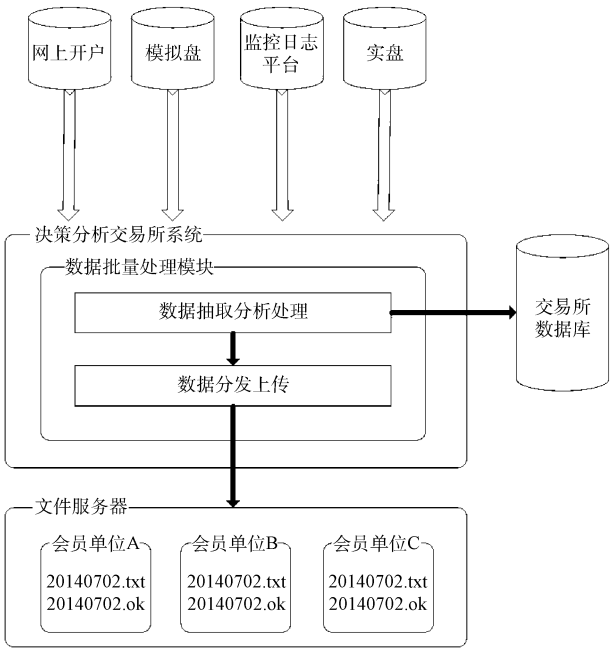

图 3　数据批处理模块搭建图

传模块的功能就是将已经生成好的数据和确认文件上传到 stp 文件服务器中。

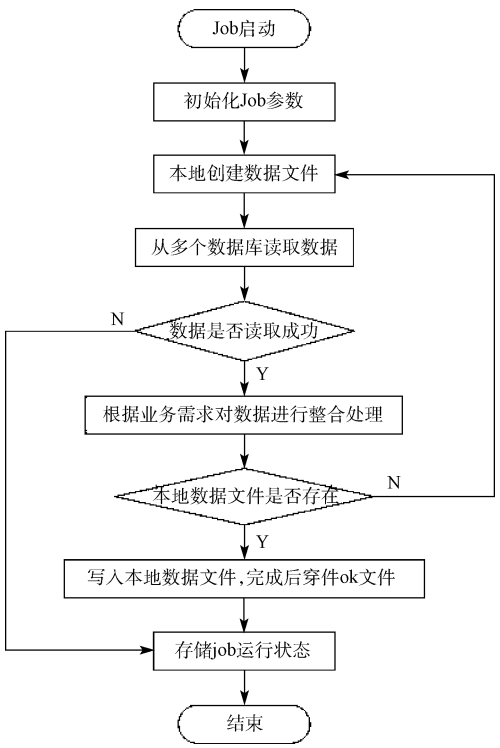

图 4　交易所端流程图

如图 4 所示，为几个源数据库与目标数据库之间进行数据抽取、处理、上传和下载的流程设计框架，交易所端详细步骤如下：

1)在会员决策分析系统中的服务器中启动项目，在配置文件或者 web 项目界面中手动配置启动 SpringBatch 任务，并运行在服务器中，每天定时执行。

2)Quarz 定时器启动 SpringBatch 中的多个 job 任务，根据配置的 step 依次执行。

3)执行准备工作的 step，根据 job 启动时候的会员单位编号查询配置该会员单位的相关准备数据。

4)在系统的指定数据文件目录中创建已该单位的文件夹以及当前 job 执行的数据名称及时间的数据文件。根据数据的内容创建数据文本的头文件，文件格式为版本号、数据内容行数，默认为 0，以及根据数据字段的表头。

5)从多个数据库中抽取文件，根绝具体的内容进行处理，执行 ItemReader、ItemProcesser。

6)检查步骤 4 创建的文件是否存在，将处理后的数据写入到步骤 4 创建的文件中，写完后更新文本文件的表示文件行号。如果数据写入成功，在写入文件的当前目录下穿件同名的数据确认文件保正数据写入完毕。同时向系统数据中记录 job 任务状态的表中写入该 job 的执行相关信息，状态以及完成情况。如果该 job 写入失败，同样写入。

7)确认数据写入完成，校验 ok 文件后，将数据文件上传至 FTP 服务器。各个任务的逻辑处理流程大体一致，区别就在于数据库操作和相关业务处理，前一部分根据不同的表使用不同 sql 语句，由 Mybatis 封装完成；后一部分在各个 job 任务的 ItemReader 或 ItemProcessor 完成。

在公用的 job 启动器启动时，会有一个参数被传入，告诉 job 启动器启动的具体的任务，这里传入的参数为表名。由于交易端的数据要分发给多个会员单位，且每个会员的数据相互独立，所以必须分别生成每个会员的数据文件。所以第一个 step 的任务，就是根据传入的当前执行的表的表名从系统配置文件中获取部署当前系统的去眼部会员单位的会员编号。在获取之后写入交易所 T_executeCondition 表中，做一个记录。

在 Step 中包含了一些其他的步骤。在会员表 step 中使用了一个 tasklet 去进行获取会员单位编号后对 T_executeCondition 的处理。一个 tasklet 对应于一个事务性的、潜在可重复的过程中发生的一个步骤，可以自己实现 Tasklet 接口，定义自己的 tasklet。这个特性很有用，比如：用于解压文件，执行系统命令，执行存储过程等待。Ref 用以指定应用的对应 bean，配置 transaction-manager 事物管理器。

Step 有一系列的 listener 来跟踪 step 的处理过程。这里所有的 listener 接口都继承了 StepListener 接口。Step listener 作为 tasklet 标签的一个子标签进行配置，这里的 listener 用来监听这个 step 运行的最后结果，如果 id 为 memberFirstStep 运行结束会返回为 COMPLETED，则进入 id 为 memberCreateFile 的 step，否则将跳入

memberFailStep。memberFailStep 会记录当前执行的 job 名称,执行时间,会员单位编号等相关信息,并进行持久化。

如果 memberFirstStep 执行成功,将继续执行下一个 step,这个名为 memberCreateFile 的 step,用来创建数据文件并根据上一个 step 得到的信息,创建数据文件的头文件,包括表名,会员单位号码,表中字段以及数据行数,配置和实现方法同 memberFirstStep 基本相同。这个 step 正常结束之后会进入下一个 step:memberStep,这是整个 job 的核心步骤,数据的读取,处理,写入都实在这个 step 中完成。

在配置文件中,与前面 step 显著的区别在于 tsklet 中多了 chunk,tasklet 和 chunk 是用来指定处理过程的。一个 tasklet 对应于一个事务性的、潜在可重复的过程中发生的一个步骤。而 chunk"块处理",配置 tasklet 很简单,但是配置"块处理"就会复杂一点,因为它涉及的内容更多。这里用到的 chunk 的属性有:reader、processor、writer、commit-interval 事物提交一次处理的 items 的数量。也是 chunk 的大小、retry-limit 最大的重试次数、cache-capacity 重试的缓存策略。

在上述的 step 配置中,一个 step 使用 gnnt _ transactionManager 定期提交读写的项目,当 commit-interval 配置为 1 的时候,表示它在每一次读写之后都要进行提交操作,这是一种很不理想的情况,开始和提交一个事务的开销是十分昂贵的,所以在理想的情况下,我们要尽可能的在每一次事务中进行更多的操作,这取决于数据操作的类型以及每一个 step 操作的开销。因此每一次事务提交的操作数量是可以被配置的。

上述代码中的 3 个 bean 就是这个 step 中对数据操作的 3 个 item,读、操作、写。这里使用 Spring 的容器特性,配置 bean 容器。在配置 bean 的时候使用 step 作用域。当使用 SpEL(SpringExpressionLanguage)的时候,step 作用域很有用,来实现属性的绑定。step 的作用的实例范围包括:jobParameters、jobExecutionContext、stepExecutionContext。这里使用 #{jobParameters['memberno']}读取 jobParameters 中的 memberno 值,传到 bean 中。在 MemberReader 类必须实现 ItemReader 接口,MemberProcessor 和 MemberWriter 也是一样要实现各自的 Item 接口。Reader 根据业务 SQL 语句分页查询得到的数据后放入 list 中,MemberProcessor 拿到 list 进行遍历处理,后再把 list 交给 MemberWriter,把数据集按照格式写入 member.txt 中,如果在写入文件的过程中没有抛出异常,job 就会跳转到 memberCreateOkFile 的 step,创建与 txt 文件名相同的同名 ok 文件,并使用 JSch.jar 包实现 SFTP 协议的上传,memberLastStep 用来更新 T _ executeCondition 表中的数据,记录任务的成功信息,如果任务失败执行 memberFailStep。

3.2.3　会员端模块

在决策分析会员端系统中的数据批量处理模块的功能就是从 ftp 服务器中下载已经由决策分析交易所系统中上传的会员单位数据。这部分的功能也是使用 SpringBatch 框架,以每 5 min 一次的频率从服务器中轮询,如果发现服务器中的某一个本地没有的数据文件,立即检查是否存在该文件的确认 ok 文件,如果发现数据文件存在,而无确认文件,那么该数据文件很有可能是不完整的或者有问题的,会员端的系统会通讯交易所端系统,交易所端的人员可以在交易所端的 web 应用中选择某一个会员单位某一天的数据而手动发起任务。如果该数据文件的 ok 文件存在,会员端系统会立即下载数据文件,再将数据文件写入本地数据库。

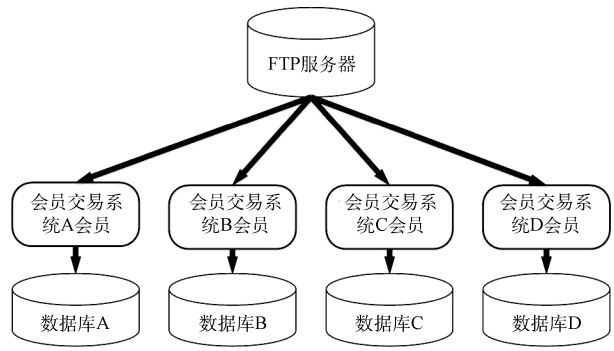

图 5　会员端数据库结构图

会员端是各个会员单位部署在各自本地的会员决策,与会员决策分析系统交易所端是相辅相成的部分。会员端会程序会轮询 FTP 数据服务器中的数据文件,将当前日期下的最新的文件下载到会员本地,再写入到会员本地的数据库中。各个交易所会员单位之间的数据相互独立,只有交易所端才拥有访问所有会员数据的权限,根据会员决策分析系统会员端的部署 IP 及权限等级,各自会员单位只拥有权限下载和读取本会员单位的数据权限,不能够跨数据库读取其他交易所会员单位的数据。这样确保了各个交易所会员单位的数据安全。

如图 6 所示,会员端详细步骤如下:

1)在服务器中启动部署并启动会员管理决策系统,根据 Quartz 定时器的配置,Springbatch 会定期轮询 FTP 服务器。

2)Job 任务启动检测服务器中的数据文件日期是否为当日最新的日期且本地数据库中没有该文件后,验证 ok 文件存在,及进行文件的下载。

3)在文件下载后调用 job 文件中的 step,执行 ItemReader、ItemProcessor、ItemWrite,将数据从文件读取出来写入到本地 Mysql 数据库中。

4)判断文件写入 step 的返回状态,根据返回的完成状态,写入数据库中的 job 运行状态表,如果任务失败,则在

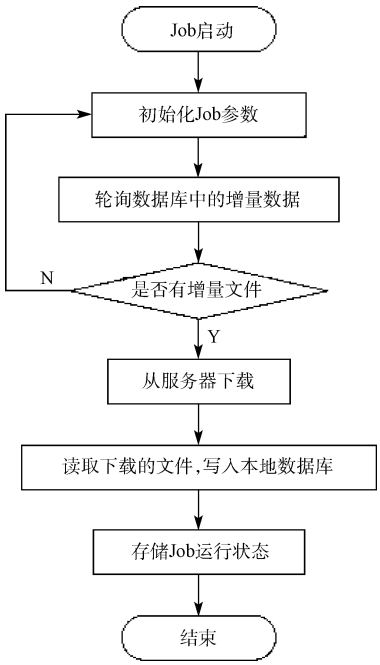

图 6　会员端流程图

web 管理界面中可以手动发起任务，并返回任务失败原因。

会员端功能的 SpringBatch 实现与配置方法与交易所端基本一致，但具体 step 流程与交易所端的步骤不同。会员端业务的主要功能就是从 SFTP 服务器中获取数据写入会员端本地数据库。所以在配置 job 任务 step 中的流程与交易所端的逻辑处理是不同的。

决策分析系统会员端会员任务的第一个 step 就是要去验证服务器中对应会员单位的目录下是否有正在执行的任务需下载文件的 ok 确认文件，在 tasklet 中，重写了 step 的返回值，使用一个监听器 listener 去检测，如果返回值是 STOP，那么这个 job 将不会继续向下执行，在执行了 validateMemberStep 后，job 会被停止；如果返回 OVER，说明这个 step 失败，将跳入处理失败的 validateMemberStep，如果执行正常，检测到 ok 文件，会继续执行 writeMemberStep，在这个 step 中会完成对数据的批量处理操作。

3.4.4　数据分发

数据分发是会员决策系统交易所端和会员端数据数据传输的纽带。会员分析决策系统的交易所端和会员端的数据传输完全是依赖 SFTP 服务器。在交易所端数据抽取处理完毕后，会由执行任务的 Job 将数据按照指定的格式封装完毕后上传至 SFTP 服务器中。而决策分析系统会员端部署后，Job 任务会自动轮询 SFTP 服务器，去搜索自身会员单位权限目录下的数据文件，更新并下载本地数据库中没有的数据增量。轮询去文件服务器指定目录对应日期文件下查找。有标示文件就取下数据文件。

文件服务器使用 sftp 服务器，为每个用户提供一个登陆账户和访问的目录结构，设置根目录名称为：decision-sftp（会员决策分析系统 sftp 目录）假设金三元（jinsanyuan）会员单位的根可访问目录结构：decision-sftp/jinsanyuan。根据不同模块在该目录下创建不同的文件，如模块 1（model1）的目录结构如下：decision-sftp/jinsanyuan/20140714_model1.txt。

表 1　数据文件及确认物件格式

文件路径名称	描述
decision-sftp/jinsanyuan/20140714_model1.txt	数据文件
decision-sftp/jinsanyuan/20140714_model1.ok	数据完成确认文件

每天该模块都会产生两个文件一个数据文件，一个数据完成的确认文件路径如表 1。文件名称：yyyyMMdd_{modelName}.txt。文件名称的命名规则为日期加模块名称作为文件名称，后缀为 txt。文件内容为文本格式，采用 UTF-8 编码。第一行第一部分是版本的信息，接下来的数据是汇总数据，总笔数为行记录数，总金额为指定一个金额字段的总额，如果未指定字段，总金额可以没有，使用方在使用行记录前，需要验证总笔数、总金额。第二行为字段描述，字段名之间用|分隔。从第三行开始以下为数据行记录，字段值之间用|分隔。数据文件内容示例如图 7 所示。

```
version:1|rows:10|sum_money:30.5
id|code|type|bank|date|money|description
20140522000000001|20140522000000000001|JJGSDM000|001|20140522|3.05|描述 1
20140522000000002|20140522000000000002|JJGSDM000|001|20140522|3.05|描述 2
20140522000000003|20140522000000000003|JJGSDM000|001|20140522|3.05|描述 3
20140522000000004|20140522000000000004|JJGSDM000|001|20140522|3.05|描述 4
20140522000000005|20140522000000000005|JJGSDM000|001|20140522|3.05|描述 5
20140522000000006|20140522000000000006|JJGSDM000|001|20140522|3.05|描述 6
20140522000000007|20140522000000000007|JJGSDM000|001|20140522|3.05|描述 7
20140522000000008|20140522000000000008|JJGSDM000|001|20140522|3.05|描述 8
```

图 7　数据文件格式

在数据 txt 文件生成完毕之后，系统会生成一个与数据 txt 文件同名的确认文件，以.ok 结尾，该文件没有任何数据内容，目的是确认数据文件已经生成完整。在会员决策分析系统会员端从 SFTP 上下载数据的同时会验证要下载的数据文件的同名的确认文件，例如下载 decision-sftp/jinsanyuan/20140714_model1.txt 文件的同时，会确认 decision-sftp/jinsanyuan/20140714_model1.ok 路径下的 20140714_model1.ok 文件，已确保要下载 20140714_model1.txt 的数据完整性。

4　数据批量处理模块测试分析

数据批量处理模块的测试目的主要是观察交易所端

系统与会员端系统的数据处理时间、处理断点续接率、异常数据数目。由于业务的不同,交易所端的数据处理方位为读取、会员端的数据处理方式为写入。本次数据测试采用的数据库为贵金属交易系统的模拟盘数据库抽取的数据信息。

利用该系统部署在交易所端和会员端进行测试,设置定时任务,按照业务进行的流程正常启动运行,运行信息会已日志文件的形式记录下载。以会员信息表、监控日志表、交易所返会员金额明细表、交易活跃度表和出入金表 5 张目的数据库表为实例,对数据库中的数据进行了测试和实验,测试结果如表所示。

表 2　实验测试结果数据

数据所属任务名称	数据量/条	数据处理方式	断点续接率(%)	异常(错误)及中断数据条数	处理时间/s
会员信息表	256243	读取	94	34	4536
		写入	99	13	1093
监控日志表	1125798	读取	88	98	5727
		写入	97	60	3480
交易所返会员金额明细表	86479	读取	90	14	501
		写入	97	7	359
交易活跃度表	854671	读取	86	77	4175
		写入	94	61	2792
出入金表	347836	读取	93	52	2238
		写入	95	44	1444

从表 2 可以看到该系统业务数据处理时间,表的写入速度处理速度相比于数据读取速度大概了快了 1.45 倍,断点续接也有较好的处理能力能力,减少了因异常(或错误)而中断数据处理的几率。从而表明该系统在批处理方面确实可以完成分析系统的业务处理需求,并具备一定的断点续接能力。

5　结　　论

本文采用 SpringBatch 配以及功能丰富的开源作业调度库 Quartz 代替传统的数据批量处理框架进行大数据批量处理,并将其应用在决策分析系统中。经过测试,实验证明使用了 SpringBatch 框架的会员决策系统其数据处理性能得到了极大的提升,同时还降低了数据异常的几率。

参考文献

[1]　刘扬. 基于 Spring 框架的数据集成与转换研[D]. 中国海洋大学,2013.

[2]　胡启敏,薛锦云,钟林辉. 基于 Spring 框架的轻量级 J2EE 架构与应用[J]. 计算机工程与应用,2008(05):115-118,133.

[3]　刘相. Spring Batch 批处理框架[M]. 电子工业出版社,2015.

[4]　翟剑锟. Spring 框架技术分析及应用研究[D]. 中国科学院大学,2013.

[5]　肖露. Spring 框架研究与应用[D]. 长沙理工大学,2011.

[6]　罗伟. 商业银行批量处理系统的设计与实现[D]. 吉林大学,2012.

[7]　周跃. 大数据处理性能及可靠性研究[D]. 复旦大学,2013.

[8]　李施施. 面向电子商务推荐系统的研究与应用[D]. 湖南大学, 2013.

[9]　张宇,王映辉,张翔南. 基于 Spring 的 MVC 框架设计与实现[J]. 计算机工程,2010(04):59-62.

[10]　李代伟,李蕖. 基于 Java 的通用批处理作业系统的设计与实现[J]. 软件工程师 Software Engineer,2014,17(7):9-12.

[11]　程学旗,靳小龙,王元卓,等. 大数据系统和分析技术综述[J]. 软件学报,2014(09):1889-1908.

[12]　苏翔宇,朱爱群. Hadoop 整合 Cassandra 处理海量数据[J]. 电脑知识与技术,2013(07):1491-1493.

[13]　雷璐宁,石为人,熊庆宇,等. 基础设施网络关联模型研究[J]. 仪器仪表学报,2013(12):2660-2665.

[14]　聂辉华,江艇,杨汝岱. 中国工业企业数据库的使用现状和潜在问题[J]. 世界经济,2012(05):142-158.

作者简介

徐杨,硕士,主要研究方向为数据分析、分布式计算等。

E-mail:xuyang091@sina.com

周文彬,硕士,主要研究方向为大数据处理。

E-mail: binbin_2020@qq.com