第14卷 第24期 2014年8月
1671—1815(2014)24-0248-04

科 学 技 术 与 工 程
Science Technology and Engineering

Vol. 14 No. 24 Aug. 2014

计算机技术

基于大数据的对账算法的设计

闵孝忠 朱林立
(江苏理工学院信息中心,常州 213001)

摘 要 介绍了在大数据以及高并发的情况下,对大型交易系统中对账子系统进行模型定义以及对账算法设计,并把设计的算法应用于工程实践中,在实际的工业生产中取得较好的运行性能,其稳定度、可靠度、准确度都表现出了工业生成中的优异性能。

关键词 大数据 数据分片传输 对账算法
中图法分类号 TP311.1; **文献标志码** A

随着大数据时代的到来,大数据相关的应用呈现出爆炸式增长,尤以大型交易系统最为典型,交易系统能稳健可靠运行,背后是大量后台关键子系统对其的支撑,大并发 Web 请求和智能手机终端请求导致交易系统异常繁忙,随即而来是数据大量增加,由于数据急剧膨胀[1],交易系统对后台子系统的数据处理提出了更高稳定性,可靠性,快速性等要求,对账子系统就是其中一个比较重要的子系统。

本文主要提出了在大数据情况下,对交易系统中对账子系统进行快速实现对账,提出一种对账数据模型以及在此模型基础上构建数据分片并发传输算法和快速对账算法,把算法实现后直接应用到现实的大型交易系统中并取得了很好的效果。

1 问题描述

大型交易系统每天处理海量数据,包括各种商品信息,物流信息,客户信息,有效订单信息,交易系统通常需要跟其他第三方系统进行对接,譬如彩票平台供应商,火车票平台供应商等,这样交易系统经常需要与第三方系统供应商进行对账清算,因为数据量庞大,常规的人工方式、半自动化对账方式已经远远不能满足系统高性能,高稳定性等实现需求,甚至常规的全自动化处理方式采用大量数据逐步读取,读取到一条数据然后和目标平台的相对应的数据进行匹配,决定该条记录是否对账成功,这样的处理方式能解决对账问题,但是在大数量大并发的情况下,性能很差,往往对账需要几个小时才能完成,这也是很多系统现在采用的对账方式,在这种情况下,为了提升处理效能,高可靠性快速性的全自动化的对账方式应运而生,在有的交易系统中已经逐步被采用,但是即使在这些系统中很多没有解决海量数据模式下的对账高效对账算法和大量对账数据的传输的问题。

2014年3月15日收到 江苏省高校自然科学研究项目(10KJD520002)资助

第一作者简介:闵孝忠(1978—),男,江苏常州人,工程师,硕士。研究方向:人工智能,软件工程,组件技术开发,系统架构。E-mail:mxz55@163.com。

2 数据模型定义

2.1 对账基本要素定义

为了建立交易系统的对账模型,首先应该确定模型中的基本要素,以及这些要素之间的相互关系。

定义 $IS = (U,C,V,f)$ 为一个交易系统对账子系统,其中,

U:系统的非空有限唯一对象集合,称为对象唯一域;

C:属性的非空有限集合,定义为 $C = \{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\cdots,\alpha_n\}$,它们的属性反映 n 个不同条件的各类属性,每个属性可以赋予不同的权重;

V:全体属性的值域,$V = \bigcup_{\alpha \subset C} V_\alpha$,$V_\alpha$ 表示属性 $\alpha \in C$ 的值域;

f: $U \times C \rightarrow V$ 的一个映射,称为映射函数。映射函数 $f = \{f_{\alpha_i} \mid 1 \leqslant i \leqslant n\}$ 为每个唯一对象的每个属性赋予一个信息对,对象的信息通过指定对象的各属性值来表达。映射函数非常重要,如果其不存在,那么对象集 U 和属性集 C 是孤立的,所以映射函数表达了两者之间的关联,是对账系统模型建立的关键。

在交易系统对账子系统中主要涉及对账元素,对账结果,操作等3大元素,本文对参与对账双方平台进行预定义,交易系统主平台方简称为 A 平台,

第三方专用服务供应商平台称简称为 B 平台。A 平台对账目标数据定义为 V_a,B 对账目标平台数据为 V_b。

由于对账数据量巨大,对 B 平台的数据 V_b 进行分片处理,定义数据片的块状大小为 PS。

2.1.1　对账元素约简

对账元素由对象唯一域 U,属性集 C 和值域 V 根据映射关系组成超级矩阵,根据对账系统关注点以及属性的冗余性,对 C 进行约简,定义约简后属性集 $F(C)$。属性约简的原则是根据属性的权重不同进行计算,约简掉权重小于某个阈值的属性集,属性权重值根据参数文件提供,系统刚开始运行,阈值可以根据人为设置,随着系统不间断运行,数据量的不断变多,可以不断调整。原则是保证通过阀值过滤掉的属性不影响对账结果的准确性[2]。

2.1.2　对账结果

结果 = {对账成功,A 平台有而 B 平台没有,B 平台有而 A 平台没有,对账明细错误} 对账成功定义为 A、B 平台对账元素中的各个元素依次数据完全相等;A 平台有而 B 平台没有则定义为 U 在 A 平台中而在 B 平台中没有;B 平台有而 A 平台没有则定义为 U 在 B 平台有而 A 平台没有;对账明细错误则定义为 A,B 平台都要对应的条目,对账元素所定义的属性只要有一个不相等,则对账明细错误。

2.1.3　操作

操作 = {A 平台元素 - B 平台元素,B 平台元素 - A 平台元素,A 平台元素 B 平台元素∩集后匹配}

2.2　模型定义

2.2.1　数据分片定义

在大数据模式下,对 B 平台数据进行分片,即按照 PS 对 V_b 进行划分,就可以获得 B 平台所有对账数据的子集 $(V_{b1},V_{b2},\cdots,V_{bn})$,其中 $V_i(1\leqslant i\leqslant n)$ 代表某个时间片里所有的数据。$V_{bi}\cap V_{bj}=\varnothing(\leqslant i,j\leqslant n)$ 并且 $V_{b1}\cup V_{b2}\cup,\cdots,\cup V_{bn}\subseteq V_b$。

2.2.2　对账运算模型定义

V_a 为 A 平台所有对账数据,V_b 为 B 平台所有对账数据,$\mathrm{SUB}(V_a,V_b)=\forall a\subseteq SA,a\not\subseteq SB$;$\mathrm{SUB}(V_b,V_a)=\forall a\subseteq SB,a\not\subseteq SA$;$SC=V_a\cap V_b$,且 $VC_{ai}(1\leqslant i\leqslant n)\equiv VC_{bj}(1\leqslant j\leqslant n)$,SUB 运算为差错运算,$SC$ 运算为平台数据对账运算,只有对账双方平台交集部分执行 SC 运算,即按照双方非空有限集合属性值按照顺序依次匹配,若完全匹配,则该条记录对账成功,否则失败。

3　算法设计

3.1　对账算法

步骤 1:B 平台上传 U、C、V 到指定的第三方服务器。

步骤 2:平台 B 向平台 A 发出即时通知。

步骤 3:A 平台获取通知后,解析数据包获取 IP 地址和文件名,跟第三方服务器建立数据交换通道,建立 FTP 通道。

步骤 4:在 FTP 获取的数据根据 2.1.1 提供的算法对元素进行约简,约简后属性集 $F(C)$。

步骤 5:在 $F(C)$ 基础上,根据预定义变量 PS 对数据按 2.2.1 定义进行分片。

步骤 6:对第 1 份分片数据按照 2.2.2 定义的对账运算模型算法进行计算并把对账结果写入目标库中。

步骤 7:循环直至重复处理第 N 份分片数据结束。

3.2　数据分片并发传输算法

数据分片并发传输即首先对数据分割,并对分割后的数据进行编号,同时生成 m 个线程抓取相对应的数据块向目标地址并发传输,这里涉及到传输过程中的容错处理。

假定 V_b 全集数据集合 $V_b=\{V_1,V_2,\cdots,V_n\}$,其中 $V_i(1\leqslant i\leqslant n)$ 为全集中的一个单元数据;有一组线程 $T=\{T_1,T_2,\cdots,T_m\}$,其中 $T_i(1\leqslant i\leqslant m)$ 为线程组中某一个线程;TV_{ij} 表示线程 T_j 是否访问并成功传输单元数据 V_i,如成功访问完毕用 1 表示,否则用 0 表示[3],则我们可以获得一组线程访问矩阵 V_T。

$$[V,T]_{VT}=\begin{bmatrix} VT_{11} & VT_{12} & \cdots & VT_{1m} \\ VT_{21} & VT_{22} & \cdots & VT_{2m} \\ \vdots & & & \vdots \\ VT_{n1} & VT_{n2} & \cdots & VT_{nm} \end{bmatrix}。$$

可将其映射到对账子系统 IS 上,其中 $U=V$ 是一个非空有限集,为单元数据集合的全域,$C=T$,线程集看成是个体属性的非空有限集,每个线程为一个属性,$V=VT$ 为属性 C 的值域,f 为映射函数。

步骤 1:定义线程数据结构,记录线程编号 TNo,读取块起始位置编码 $DOffset$,操作标记 F;通过解析协议数据包,获取总数 S。

步骤 2:根据 2.2.1 定义,按 PS 对 V_b 进行划分,定义 DS 为子集数据长度,传输数据块大小 $TPS=DS*N$,N 为连续子集数量,传输数据块大小决定了颗粒大小,也决定了单元线程处理的总数据量。根据公式 $I_i=S/TPS*i$,其中 $(1\leqslant i\leqslant n)$,把计算出来的 I_i 存入相应的划片索引 $DOffset$,标记初始化 $F=0$,TNo 自动累加存入每个独立线程。

表 1 属性权重

Table 1 Attribute weights

彩票编号	登录名	渠道编号	订单编号	手机号码	期次	彩种	注数	倍数	支付类型编号
0.4	0.08	0.12	0.2	0.2	0.2	0.3	0.4	0.4	0.05
支付金额	出票状态	业务编号							
0.6	0.4	0.1							

步骤 3:启动所有线程,每个线程根据步骤 2 中的 *DOffset* 定位数据源。

步骤 4:在线程运行中判断操作标记 F,若 $F=0$,则继续获取传输数据库 *TPS*,若出现错误,则 $F=1$,线程结束;若没有错误,向 Socket 传输数据,若出现错误,则 $F=1$,线程结束;若线程结束,$F=0$ 表明线程处理完毕,成功。

步骤 5:遍历所有 $F=1$ 的线程,重复步骤 3 和步骤 4。

步骤 6:按照编号进行数据合并入库。

4 运行结果

这里选用了江苏某大型移动通信公司的线上彩票交易系统的系统,我作为该系统的架构师和高级程序员的身份全程参与了系统设计开发工作,系统采用 JavaEE + Oracle 的技术架构实现,根据对账子系统 *IS* 的模型的定义,$C=\{$彩票编号,登录名,渠道编号,订单编号,手机号码,期次,彩种,注数,倍数,支付类型编号,支付金额,出票状态,业务编号$\}$,每个属性都计算相应的权重值。

4.1 重要参数的设置

权重的计算依据根据属性 C 在 $F(C)$ 中的重要性决定,属性值大小根据属性重要性决定,阈值设置为 0.18。见表 1。

根据权重和阈值进行属性过滤,获得 $F(C)=\{$彩票编号,订单编号,手机号码,期次,彩种,注数,倍数,支付金额,出票状态$\}$。

4.2 *B* 平台彩票原始数据的分片以及多线程数据解析

我们选取 *B* 平台运行时间段为某一天的对账数据进行运行结果分析,*B* 平台传入的对账数据为 125.8 万条,*PS* 设定为 3 000,每 3 000 条记录形成一个元组,连续子集数量 $N=5$,生成的线程数 $m=84$,为每个线程动态生成数据结构,每个线程计算 *DOffset*,初始化 $F=0$ 存入数据结构,同时启动 m 个线程开始工作[4]。每个线程定位到 *DOffset* 后开始解析数据,这里选取 *B* 平台其中一条数据进行如下定义。

定义 1 彩票业务结构 $=\{$*B* 平台订单编号,*A* 平台订单编号,手机号码,注数,金额 ,投注时间,彩种,期次 ,支付方式,投注信息,平台编号,来源$\}$

定义 2 投注信息结构 $=\{$彩票编号,投注内容,玩法,投注方式,倍数, 单票金额,彩票状态$\}$

彩票业务结构中至少包含一条或多条投注信息,因为数据量比较大,约定彩票业务结构采用符号/连接数据,投注信息采用符号@ 连接数据,多条投注信息之间采用符号^连接,最后形成一整串数据,解析数据示例如下:

1310280617160047807 9/13829116204828688/15951302144/5/10/20131028061716/01/2013127/1/1310280617160056019 7@04,19,20,25,26,32|03@1@1@1@2@3@0@0^1310280617160061688 4@02,10, 19, 25, 27, 29 | 04 @ 1 @ 1 @ 1 @ 2 @ 3 @ 0 @ 0 ^1310280617160064938 8@02,07,22,24,30,32|03@1@1@1@2@3@0@0^1310280617160066236 6@05,09,11,15,20,29|10@1@1@1@2@3@0@0^1310280617160067985 3@04,13,15,18,23,32|07@1@1@1@2@3@0@0/1/1

每个线程负责解析对应空间的数据段,解析程序首先通过解析符号/分割数据,然后判断分割后的数据是否是元数据,如果不是元数据则通过符号^和符号@ 再次分割数据,分割后数据形成 2.1 定义的属性集 $F(C)$,$F(C)$ 中数据集构成数据矩阵。解析过程中线程实时跟踪当前线程工作状态 T,若 $T=0$ 表明,对账操作成功,$T=1$ 表明对账操作出现异常,同时标记该线程的工作状态。

4.3 对账

数据矩阵按照 2.2.2 对账模型算法进行对账,为了提升性能,减少 IO 操作[5],所有分片数据全部载入内存,在内存中完成对账,主要完成 3 个步骤运算,1:SUB(V_a,V_b),2:SUB(V_b,V_a),3:SC 运算,其中 SUB 运算的结果直接作为差错结果记录到目标库中,SC 运算先计算交集,然后根据 $F(C)$ 属性集进行相等匹配,$F(C)$ 属性集采用或操作,其中有任何一项相等操作出错,则把记录入差错库,例如其余属性数据完全相同,但只要彩票号码 05,09,11,15,20,29|10 与目标平台为 05,09,11,15,20,29|11 的号码不同,则这条记录对账失败。

4.4 性能分析

通过 Log4J 记载的日志,这里随机截取了 10 个线程的运算时间,线程编号及对应的运行时间数据整理见表 2。

表 2　线程运行时间/s

Table 2　Thread running time/s

线程 2	线程 5	线程 9	线程 13	线程 40	线程 34	线程 55	线程 19	线程 28	线程 79
0.051	0.053	0.048	0.05	0.061	0.056	0.063	0.058	0.049	0.064

在高并发的环境下,各个线程完成各自数据块对账运算的时间非常接近,性能非常优异,上述表格的测试数据是在服务器空闲时间段如凌晨 1 点的时间记录的,这样的性能足以应对大数据量高强度对账运算的需求,真实的工业平台运行也验证了分片对账算法的可靠性稳定性和高速性。

5　结语

本文提出了交易系统在大数据量情况下平台对账优化算法,通过对大数据进行分片并发传输后目标平台,把分片数据通过定义的 SUB 和 SC 操作进行内存对账,从而完成分片数据快速传输及在内存中完成对账,对账算法已经在真实环境中测试,并且运行良好。系统也有可以继续完善的地方,如属性过滤阶段,可以通过数据量大小适当调整阈值过滤掉更多无关对账结果的属性,这样运算的数据量会大幅减少,从而更多的提升性能。

参 考 文 献

1　Tiem J M. Big data: unleashing information. Journal of Systems Science and Systems Engineering, 2013;22(2):127—151

2　刘　山,张　慧. 基于条件信息量的动态属性约简方法. 计算机工程, 2007;33(11): 182—183
Liu S, Zhang H. Methods of dynamic attribute reduction based on information quantity. Computer Engineering, 2007;33(11): 182—183

3　吴润秀,吴水秀,刘　清. 基于粒计算的数据分片算法. 计算机应用, 2007;27(6): 1388—1392
Wu R X, Wu S X, Liu Q. Datafragment algorithm based on granular computing. Computer Applications, 2007;27(6): 1388—1392

4　孙彩霞,张民选. 使用取指策略控制同时多线程处理器中个体线程的性能. 计算机学报, 2008;31(2): 309—317
Sun C X, Zhang M X. Using instruction fetch policy to control performance of a thread in SMT processors. Chinese Journal of Computers, 2008;31(2): 309—317

5　朱　平. HPC 海量存储系统 Pass_Through 访问策略研究. 计算机研究与发展, 2013;50(8):1667—1673
Zhu P. An Access pass-through policy of storage unit under HPC mass storage system. Journal of Computer Research and Development, 2013;50(8):1667—1673

Design of Reconciliation Algorithm Based on Large Data

MIN Xiao-zhong, ZHU Lin-li

(Information Center, Jiangsu University of Technology, Changzhou 213001, P. R. China)

[**Abstract**] This paper describes the situation in the big data and high concurrent, large-scale trading system Reconciliation subsystem model definitions and a design of reconciliation algorithm, and the design of the algorithms used in engineering practice, get in the actual industrial production better operating performance, its stability, reliability, accuracy, have shown excellent performance in generating industry.

[**Key words**] big data　　data fragmentation transfer　　reconciliation algorithm