ETL 任务集群调度方法

李 磊
(北方工业大学 计算机学院,北京 100144)

摘 要:随着数据仓库规模越来越大,ETL 任务也不断增多,单机调度 ETL 任务导致多数 ETL 任务不能按时运行或者不能运行情况时常发生。对基于 Kettle 的 ETL 任务调度方法进行了研究,根据这种 ETL 任务特性,ETL 任务调度方法作用的对象是一批相互没有制约的任务。把 ETL 任务调度分为两个阶段:任务分配与任务执行。为了避免集群负载的不均衡,根据 ETL 任务的关键特性数据源的数据量,使用贪婪调度算法进行 ETL 任务分配。为了避免一些 ETL 任务获取不到机会执行,采用动态调整任务优先级的方法,使用高响应比优先调度算法执行 ETL 任务。通过 ETL 任务测试该集群调度方法的效率,主要比较 ETL 任务执行时所消耗的 CPU、内存,以及一次全部的 ETL 任务执行完成后使用的总时间,并与轮转调度算法进行对比,结果表明效率高于轮转算法。

关键词:数据仓库;抽取-转换-加载;quartz 集群调度;贪婪调度算法;Kettle
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2018)11-0035-04
doi:10.3969/j.issn.1673-629X.2018.11.008

One Scheduling Method for ETL Task Cluster

LI Lei
(School of Computer, North China University of Technology, Beijing 100144, China)

Abstract: As the scale of data warehouse becomes larger and larger, ETL tasks are also increasing. As a result of single-machine scheduling of ETL tasks, most ETL tasks cannot run on time or cannot run frequently. The ETL task scheduling method based on Kettle is studied. According to the ETL task characteristics, the ETL task scheduling method acts on a batch of tasks with no restriction on each other. ETL task scheduling is divided into two stages: task assignment and task execution. In order to avoid unbalanced cluster loads, greedy scheduling algorithm is used to allocate ETL tasks according to the data volume of key characteristic data sources of ETL tasks. In order to avoid some ETL tasks being unable to get the opportunity to execute, we adopt the method of dynamically adjusting task priority and use priority scheduling algorithm of high response rate to execute ETL tasks. The efficiency of the cluster scheduling method is tested by the ETL task, and the CPU and memory consumed during the execution of the ETL task as well as the total time used after the completion of all ETL tasks are mainly compared, which shows that its efficiency is higher than the rotation scheduling algorithm.

Key words: data warehouse; extract-transfer-load; quartz cluster scheduling; greedy scheduling; Kettle

0 引 言

ETL(transformation and loading, extraction)技术是构建数据仓库的基础技术,也是批量数据交换的基础技术[1]。ETL 是将数据从源抽取、转换、整合、清洗并加载到目标的过程。ETL 过程是构建数据仓库的重要环节,甚至占到了整个构建过程的 80%[2]。随着时间变化,数据仓库中 ETL 任务增多,数据量也不断增多。若在单机上运行,任务执行花费大量时间,数据的时效性、可用性将严重受限。因此,研究 ETL 任务集群调度算法十分必要。如何将大量的 ETL 任务分配给集群中每个节点来获得数据仓库构建的最小执行代价,是 ETL 任务集群调度方法中主要研究的问题。数据仓库中的任务由于受到多种因素的影响,导致 ETL 任务执行时间不一,所以设计算法保障每个作业都能获取到执行的机会也是必要的。

文中基于贪婪调度算法[3]的思想,根据 ETL 任务数据源数据量的大小来合理分配任务,让集群中工作节点的负载达到均衡,使得总的任务执行时间最短。使用高响应比优先调度算法[4]动态调整任务的优先级,保障任务在节点上公平执行。

收稿日期:2018-01-08 **修回日期**:2018-05-06 **网络出版时间**:2018-06-29
基金项目:北京市自然科学基金(4172018)
作者简介:李 磊(1988-),男,硕士生,研究方向为大规模流数据处理、大数据分析。

1 问题描述

1.1 传统 ETL 流程分析

首先,数据从数据源(例如:文本文件、关系数据库、XML 文件、非结构文档等)抽取出来。其次,合适的转换、清洗、交集或并集等活动处理抽取的数据,使它们满足目标结构的需求。最后,加载处理过的数据到数据仓库[5]。

Kettle 工具把 ETL 工作流实现为一个可以在 Java 虚拟机上执行的文件,也即是 Kettle 把上述工作流映射为相应的操作组件,组合为一个完整的 ETL 任务,由 Kettle 引擎调度工作流中的各活动。文中 ETL 调度不再考虑各个活动之间如何调度执行。通常由 Kettle 工具生成的作业文件依赖于操作系统的调度工具调度,例如 Windows 的任务计划、Linux 的 crontab,调度必须手动配置,无法满足工程需求。

1.2 ETL 任务集群调度约束条件

ETL 任务定义:ETL 就是一个或多个数据源输出,经过加工处理,再输出到一个或多个数据源构成的流程[6],文中要调度的 ETL 任务就是 Kettle 工具生成的以上流程。

ETL 任务集群调度定义:中心调度节点负责分配 ETL 任务到集群中的执行节点执行。

每个 Kettle 作业包含若干作业或者转换,它们之间具有指定的先后次序关系,Kettle 转换包含的操作组件遵循的是并行执行的原则,Kettle 引擎负责每个操作组件之间的调度[7]。集群中所要调度的就是能够并行执行的 Kettle 作业。

每个 Kettle 作业同一时刻只能有一个运行在集群中。不同的 Kettle 作业可以同时运行在同一台机器或不同机器。

集群中每个 Kettle 工作节点执行能力相同。

文中选取影响任务执行时间最重要的源数据的数据量大小,作为 ETL 任务分配的依据。

1.3 ETL 任务集群调度目标函数

通常 ETL 任务执行考虑执行时间的代价、占用资源的代价以及两者相结合的代价[8]。文中选择执行时间代价。

$T(k_{ij})$:表示 i 任务在 j 机器上的执行时间代价,$1 \leqslant i \leqslant n, 1 \leqslant j \leqslant N$。

$T(m_k)$:表示机器 m_k 的执行时间代价,$1 \leqslant k \leqslant N$。

$$T(m_k) = \sum_{j=k} T(k_{ij})$$

$\sum_{j=k} T(k_{ij})$:表示截至到目前所有任务执行时间代价总和。

$T(\text{task})$:ETL 任务在所有机器上执行的最长时间。

$$T(\text{task}) = \max(T(m_k))$$

ETL 任务集群调度的目标函数可以定义为:

$$\text{Min}(T(\text{task})) = \text{Min}(\max(\sum_{j=k} T(k_{ij})))$$

2 ETL 任务集群调度方法

ETL 调度系统由调度模块与执行模块组成。调度模块(调度中心)负责管理调度信息,把所有的调度任务抽象为一个任务,按照调度配置发出调度请求,自身不承担业务代码。调度系统与具体任务解耦,提高了系统的可用性和稳定性。执行模块(执行器)负责接收调度请求并执行任务逻辑。

调度流程如图 1 所示。批量 ETL 任务,使用基于贪婪调度算法把 ETL 任务分配到执行器,把任务添加到 Quartz 调度器[9]定时执行,抽象的 Quartz 任务调用远程的执行器接口,当此执行器接收到新任务的执行调用时,会启动本地线程运行 ETL 任务。使用高响应比优先调度算法计算任务优先级,分配到执行器上的任务按照优先级获取线程资源运行 ETL 任务。

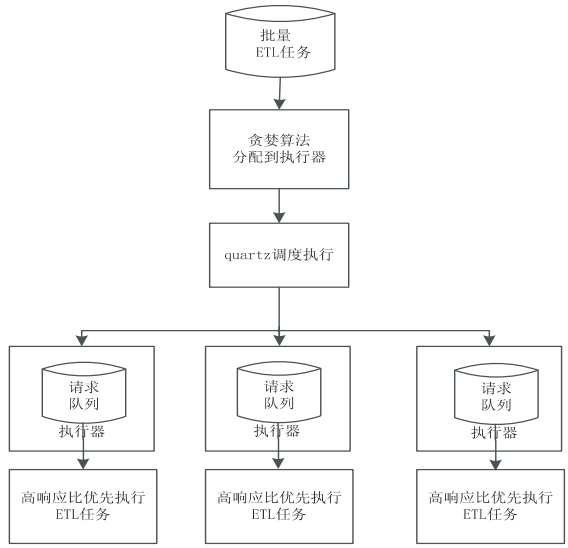

图 1 集群调度总体设计

贪婪算法是一种解决最优解的近似算法。在对问题求解时,总是做出在当前看来是最好的选择。也就是说,不从整体最优上加以考虑,所做出的是在某种意义上的局部最优解[10]。假设有一个任务集合 $S = \{s_1, s_2, \cdots, s_n\}$,$S$ 中每个任务所对应源数据量的大小 $W = \{w_1, w_2, \cdots, w_n\}$,$m$ 台工作节点 $U = \{u_1, u_2, \cdots, u_m\}$。该算法把 S 中的任务依据源数据量的大小平均分配到工作节点 U 中,求解总的任务执行时间最短的分配方法。假设每台机器分配到所有任务的源数据量总和越平均,硬件资源的负载越均衡,总任务执行时间代价越小。

贪婪分配算法具体步骤如下:

(1)对任务集合 S 按数据量从大到小排序,存入队列 A 。$A=\{(s_i,w_i)\cdots(s_j,w_j)\}$,其中 $w_i \geqslant w_j$,$1 \leqslant i, j \leqslant n$ 。

(2)从队列 A 中取出队首的任务,计算各个节点任务数据量的总量,把该任务分配到 u_i ,$1 \leqslant i \leqslant m$ 。u_i 为当前工作节点中数据量总量最少的节点。

(3)重复步骤 2 直到队列空,算法结束。

高响应比优先调度用在 ETL 调度系统中来动态调整 ETL 任务的优先级。ETL 作业的特点也即运行时间长短不一,如果动态调整 ETL 任务的优先级,有利于任务都能有机会获得资源运行。

将该算法应用到调度框架的执行器模块,主要设计如图 2 所示。执行器不断从调度中心接收到调度请求,然后缓存到优先级队列,在线程组资源充足的情况下,优先级队列的任务都能获得线程执行。当线程组资源不能满足需求时,一些 ETL 任务由于等待时间变长而优先级变高,如果有空闲资源,这些高优先级的任务将优先获取到执行机会。为避免有些任务占用时间过长,导致其他任务获取不到执行的机会,设计任务动态优先级调整任务的执行顺序,使用典型的消费生产者模型,优先级队列为生产者的缓冲区,任务监控线程为消费者代理(负责任务分配),线程组的线程为消费者。

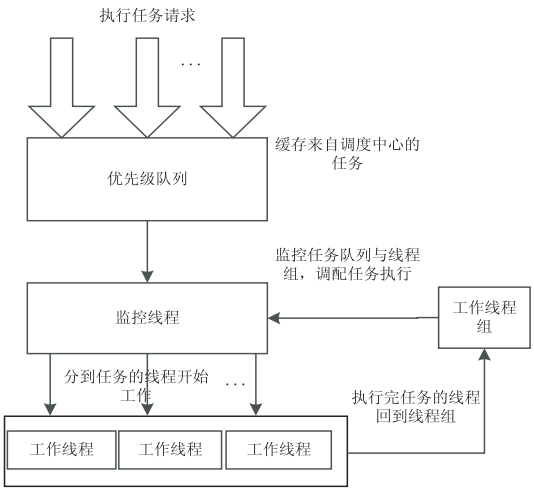

图 2　执行器任务执行策略

3　实验结果与分析

3.1　实验描述

实验中模拟 35 个 ETL 任务,其中 10 个 10 万,10 个 20 万,10 个 50 万,5 个 100 万条源表记录全量抽取任务,定时时间设置为每 16 分各个任务运行一次,记录 10 次全部任务运行完成各工作节点所需时间,求每台工作节点全部执行完的平均时间作为本节点的执行时间记录。实验使用 4 台虚拟机,1 台调度中心,3 台工作节点。配置见表 1。

表 1　集群配置表

ip	role	cpu	mem
10.61.4.100	调度中心	Intel Xeon E312xx(Sandy Bridge)	8 G
10.61.4.101	执行器 1	Intel Xeon E312xx(Sandy Bridge)	8 G
10.61.4.102	执行器 2	Intel Xeon E312xx(Sandy Bridge)	8 G
10.61.4.103	执行器 3	Intel Xeon E312xx(Sandy Bridge)	8 G

3.2　实验结果

这 35 个任务按照贪婪分配原则分配之后的结果是:执行器一分到 3 个 10 万,3 个 20 万,3 个 50 万,2 个 100 万数据量的任务,执行器二少分配一个 20 万,多分配一个 10,相比于执行器一,执行器三分配到 3 个 10 万,5 个 20 万,4 个 50 万与 1 个百万数据量的任务。其他几次测试分配情况就不一一列举。使用 ETL 中基于贪婪算法的任务调度方法研究中,基于任务的平均时间分配得到的分组结果为:5 个 100 万与 10 个 50 万与 10 个 10 万结果同上,30 万 101 多一个,102 多一个,103 少 2 个。分组结果基本上与按源表数据量大小分配相同。任务为 35 个时,各个执行器使用贪婪算法与轮转算法的资源利用率如图 3、图 4 所示。任务分别为 10、35、45 个时,使用贪婪、轮转分配算法执行完所有任务消耗的时间如图 5 所示。

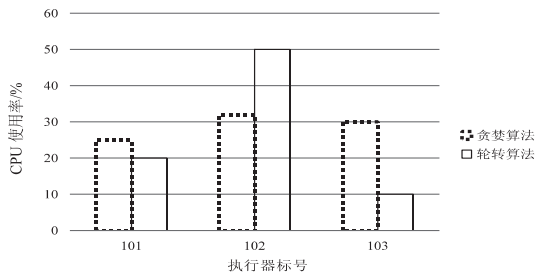

图 3　各执行器 CPU 使用率

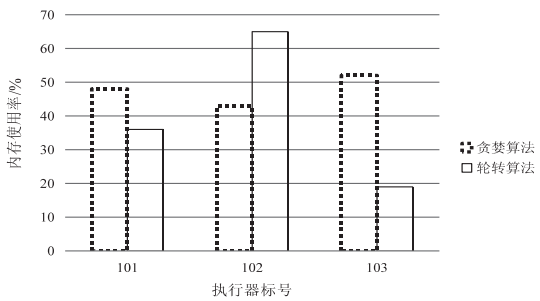

图 4　各执行器内存使用率

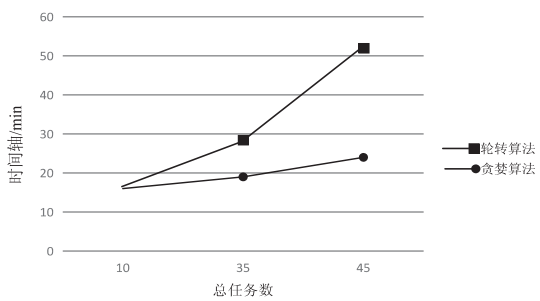

图 5　执行任务的总时间

3.3 实验分析

轮转调度算法是以轮询的方式依次将作业分配到不同的服务器,即将所有服务器组成一个循环队列,每次取头服务器分配。从任务数量上看是平均分配到各个服务器,但是由于任务执行时间的大小不等,很大可能造成服务器负载的不均衡[11]。实验中通过比较轮转与贪婪调度算法所使用资源的情况,可以看出贪婪调度算法要优于轮转调度算法。比较相同任务条件下,轮转算法总耗时多于贪婪算法。轮转只考虑了任务的数量,并未考虑任务的其他特性,造成负载的不均衡[12]。而贪婪算法考虑 ETL 任务中关键因素任务数据量的大小,以达到负载相对均衡的目的。对于 ETL 中基于贪婪算法的任务调度方法,文中按任务平均时间作为分组凭据,同样使用相同数量任务的测试,得到的分组结果基本相同,但是分别测试每个任务的执行时间增加更复杂的步骤。同时为了保证所测试时间的准确性,还需要连续测试多天的数据,选择同一任务时间相近的数据作为最终分组的凭据。

4 相关工作

现有的 ETL 调度算法主要集中在 ETL 工作流逻辑转换的过程优化,通过减少处理过程数量或改变过程执行顺序来减少 ETL 工作流的执行代价[13],或者根据 ETL 活动的分配和调度而建立数据仓库 ETL 任务调度模型。如基于粒子群算法分布式 ETL 任务调度[14],设计 ETL 任务调度模型,将连续空间域映射为离散整数域解决工作调度问题,利用启发式算法找到最佳调度策略;基于遗传算法的 ETL 任务调度[15],采用同层划分的思想进行模型化求解。这些 ETL 任务调度算法,不适合现有成熟 ETL 工具生成的任务调度。

5 结束语

分析了 ETL 任务的特点,设计与优化 ETL 集群调度框架,加入批量 ETL 任务分配算法和动态优先级算法。贪婪调度算法依据 ETL 任务特点实现集群负载均衡,优点是把 ETL 中任务数据总量均匀分配到各个执行节点,减少任务执行时间。高响应比算法,优点是在任务执行时动态调整任务的优先级,以避免 ETL 任务获取不到运行资源。结合两种算法,文中设计的 ETL 集群任务调度系统具有高效稳定的特点。同时也存在不足:由于 ETL 任务执行涉及到数据量、网络带宽、数据转换复杂度等诸多不确定因素[16],实现负载均衡任务分配只从数据量单方面考虑,做不到准确,所以设计一种能衡量一个完整 ETL 任务权重的算法十分必要;由于贪婪调度算法是一种静态的分配算法,在分配任务完之后,如果有工作节点的资源使用率过高,不能做到动态调整,所以下一步设计动态调整任务的工作节点,以达到资源使用率的均衡,从而达到所有任务执行总时间最短的目的。

参考文献:

[1] 彭木根. 数据仓库技术与实现[M]. 北京:电子工业出版社,2002.

[2] 宋旭东,刘晓冰. 数据仓库 ETL 任务调度模型研究[J]. 控制与决策,2011,26(2):271-275.

[3] 王 珊,陈 琨. ETL 中基于贪婪算法的任务调度方法研究[J]. 微电子学与计算机,2009,26(7):130-133.

[4] 江志华,齐文静. 常用作业调度算法的分析与评价[J]. 乐山师范学院学报,2008,23(12):57-59.

[5] VASSILIADIS P, SIMITSIS A, GEORGANTAS P, et al. A generic and customizable framework for the design of ETL scenarios[J]. Information Systems,2005,30(7):492-525.

[6] KARAGIANNIS A, VASSILIADIS P, SIMITSIS A. Scheduling strategies for efficient ETL execution[J]. Information Systems,2013,38(6):927-945.

[7] CASTERS M, BOUMAN R, DONGEN J V. Pentaho Kettle 解决方案:使用 PDI 构建开源 ETL 解决方案[M]. 北京:电子工业出版社,2014.

[8] KIMBALL R, CASERTA J. The data warehouse? ETL Toolkit: practical techniques for extracting, cleaning, conforming, and delivering data[M]. [s. l.]: John Wiley & Sons,2011.

[9] 胡利强,周冬初,王 伟. Quartz 调度器与 Web 程序整合的研究和应用[J]. 计算机与现代化,2010(8):98-99.

[10] WANG Shan, LI Yueping. Task scheduling strategies of ETL for banking off-site audits[J]. Information Technology Journal,2013,12(21):6273-6276.

[11] 孙凌宇,冷 明,朱 平,等. 云计算环境下基于禁忌搜索的负载均衡任务调度优化算法[J]. 小型微型计算机系统,2015,36(9):1948-1952.

[12] 陈 亮,王加阳. 基于粗糙集的负载均衡算法研究[J]. 计算机工程与科学,2010,32(1):101-104.

[13] 朱虹宇,李 挺,闫健恩,等. 基于动态负载均衡的分布式任务调度算法研究[J]. 高技术通讯,2014,24(12):1261-1269.

[14] 王春阳,赵书良,王长宾. 粒子群算法在分布式 ETL 任务调度中的应用[J]. 计算机工程与应用,2013,49(9):150-155.

[15] WANG Tingting, LIU Zhaobin, CHEN Yi, et al. Load balancing task scheduling based on genetic algorithm in cloud computing[C]//Proceedings of the 2014 IEEE 12th international conference on dependable, autonomic and secure computing. [s. l.]: IEEE,2014:146-152.

[16] THIELE M, LEHNER W. Evaluation of load scheduling strategies for real-time data warehouse environments[C]// International workshop on business intelligence for the real-time enterprise. Berlin: Springer,2009:84-99.