基于MapReduce的分布式ETL体系结构研究

宋 杰 郝文宁 陈 刚 靳大尉 赵水宁
(解放军理工大学工程兵工程学院 南京 210007)

摘 要 针对传统ETL工具集中式执行方式的不足,提出了一种基于MapReduce的分布式ETL体系结构——MDETL(MapReduce Distributed ETL)。该体系结构采用MapReduce并发处理海量数据的并行编程模型,结合分布式ETL的集群运算方法,实现了集群分布式执行ETL流程,从而提高了整个ETL系统的灵活性和吞吐率,并具有较好的可扩展性和负载平衡性能,提高了执行效率。

关键词 ETL,MapReduce,分布式
中图法分类号 TP311 **文献标识码** A

Research of Distributed ETL Architecture Based on MapReduce

SONG Jie HAO Wen-ning CHEN Gang JIN Da-wei ZHAO Shui-ning
(Engineering Institute of Corps of Engineers, PLA University of Science & Technology, Nanjing 210007, China)

Abstract Aiming at deficiency of centralized execution mode of traditional extraction-transformation-loading (ETL) tools, this paper put forward the architecture of distributed ETL based on MapReduce——MDETL (MapReduce Distributed ETL). The ETL architecture which uses a parallel programming model of massive data parallel processing with cluster computing methods of distributed ETL, achieves the cluster distributed ETL processing. It improves the whole ETL system's flexibility and throughput rate, and has better expansibility and load-balancing, raises the performance efficiency.

Keywords ETL, MapReduce, Distributed

ETL是指在构建数据仓库的过程中对数据源中的数据进行抽取(Extract)后经过数据转换(Transform)加载(Load)到数据仓库的过程。ETL整合了数据从分布异构数据源的收集、数据清洗、数据重构的流程和数据加载到目的端数据库、数据集市、数据仓库的流程,是构建数据仓库系统的关键。

分布式ETL是指利用现有技术将顺序执行的ETL流程转换为多个并行处理的过程[1]。这种并行处理分为垂直扩展和水平扩展两个方面:垂直扩展是尽可能地使用单台服务器上的多个CPU核;水平扩展是使用多台机器资源,使其实现并行计算。在保证数据并行处理的前提下,分布式ETL过程必须保证数据的完整性、流程的可靠传输以及断点恢复的能力。

传统的ETL工具采用的都是集中式架构,将ETL的设计、运行、管理都集中在一点上,随着数据集的不断增加,通常只有在价格昂贵的高性能服务器上才能保证系统具有较高的效率,构建数据仓库的硬件成本不断增加[1];而且,在当前大数据环境下,数据逐渐呈现大容量、多格式、频繁交互等特征,这就对高效、高质量地处理海量数据提出了要求。为此,在传统的ETL工具之上如何利用现有资源研究实现分布式ETL,就显得尤为重要。本文针对传统ETL工具集中式执行方式的不足,提出了基于MapReduce的分布式ETL体系结构,利用MapReduce简单而强大的数据处理接口和对大规模并行执行、容错及负载均衡等实现细节的隐藏,解决了分布式ETL的负载平衡和效率问题,从而保证了在降低硬件成本的同时,提高了ETL执行效率的稳定性和高效性,从而实现了分布式ETL的水平扩展。

1 MapReduce并发处理海量数据的并行编程模型

MapReduce是Google公司于2003年提出的能并发处理海量数据的并行编程模型,其主要原理是将数据处理任务抽象为一系列的Map(映射)-Reduce(化简)操作对。Map主要完成数据的过滤操作,Reduce主要完成数据的聚集操作。输入输出数据均以〈key,value〉格式存储。用户在使用该编程模型时,只需按照自己熟悉的语言实现Map函数和Reduce函数即可,MapReduce框架会自动对任务进行划分以做到并行执行[2]。MapReduce主要是面向由数千台中低端计算机组成的大规模机群而设计的,其扩展能力得益于shared-nothing结构、各个节点间的松耦合性和较强的软件级容错能力:节点可以被任意地从机群中移除,而几乎不影响现有任务的执行。MapReduce具有完全的开放性:其〈key,value〉存储模型具有较强的表现力,可以存储任意格式的数据;Map和Reduce两个基本的函数接口也给用户提供了足够的发挥空间,

到稿日期:2012-08-28 返修日期:2012-11-09 本文受国家自然科学基金项目(70971137)资助。

宋 杰(1986—),男,硕士生,主要研究方向为海量数据ETL、分布式计算,E-mail:admsongs@163.com;郝文宁(1971—),男,博士,副教授,主要研究方向为高维数据归约、数据工程;陈 刚(1974—),男,硕士,副教授,主要研究方向为数据工程、作战模拟;靳大尉(1979—),男,硕士,讲师,主要研究方向为数据管理、分布式计算;赵水宁(1971—),男,博士,讲师,主要研究方向为数据管理、作战模拟。

可以实现各种复杂的数据处理功能[3,4]。

MapReduce 并行编程模型的最大优势在于能够屏蔽底层实现细节，有效降低并行编程难度，提高编程效率。其主要贡献在于：①使用廉价的商用机器组成集群，费用较低，同时又具有较高的性能；②松耦合和无共享结构使之具有良好的可扩展性；③用户可根据需要自定义 Map、Reduce 和 Partition 等函数；④提供了一个运行时支持库，它支持任务的自动并行执行，提供的接口便于用户高效地进行任务调度、负载均衡、容错和一致性管理等；⑤MapReduce 适用范围广泛，不仅适用于搜索领域，也适用于满足 MapReduce 要求的其它领域的计算任务[5]。

2 分布式 ETL 体系结构设计

在高性能计算领域中，以分布式计算为理论基础构建的集群系统，由于其成本低、可扩展性好、容易构建等优点已经成为高性能计算机的主流系统[6,7]。MDETL 采用 MapReduce 并发处理海量数据的并行编程模型，利用集群系统的优势，改变传统集中式架构的 ETL 的格局，将设计、运行这两个可并行性高的任务分散到网络中由提供服务的各个节点并行处理，以提高系统的运行效率。其体系架构如图 1 所示。

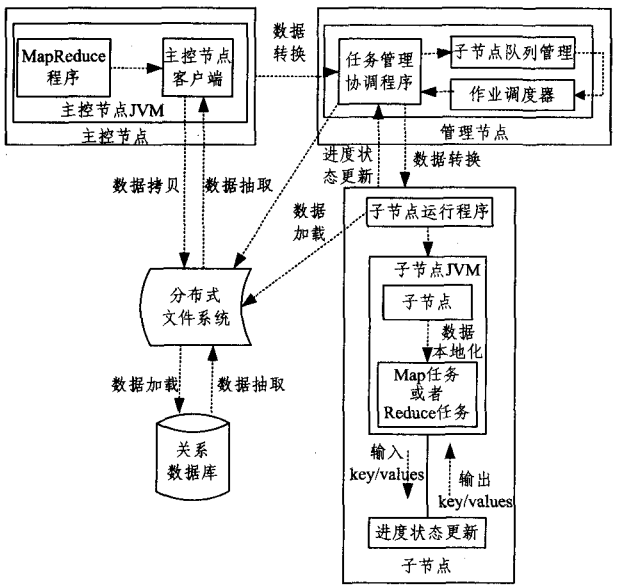

图 1 分布式 ETL 体系结构设计

分布式 ETL 体系结构主要包含以下 4 个独立的实体：

- 主控节点：负责提交 MapReduce 作业；
- 管理节点：协调整个作业的运行；
- 子节点：运行作业划分后的任务；
- 分布式文件系统及关系数据库：用来在实体间共享作业文件。

在主控节点中，通过 MapReduce 程序将 ETL 程序分解，在分布式文件系统中抽取数据，创建新的作业实例。作业提交后，每秒轮询作业的进度，如果发现自上次报告后有改变，就将进度报告至控制台。作业完成后，成功则显示作业计数器，失败则将导致失败的错误记录到控制台。

作业实例创建后，向管理节点发出调用请求，检查作业的输出说明，并计算作业的输入分片，将运行作业需要的资源复制到以子节点命名的文件系统中。同时，管理节点收到调用请求后，会把此调用放入一个内部队列，交由作业调度器进行调度，并对其进行初始化。初始化包括创建一个表示正在运行作业的对象——封装任务和记录信息，以便跟踪任务的状态和进程。

管理节点为了创建任务运行列表，作业调度器也要从共享文件系统中获取主控节点计算好的输入分片信息，然后为每个分片创建一个 map 任务，创建的 reduce 任务数量由任务属性决定，然后由任务调度器创建相应数量的要运行的 reduce 任务。

子节点对于 map 任务和 reduce 任务有固定数量的任务槽。任务槽的数量由子节点核的数量和内存大小决定。子节点通过运行一个简单的循环来定期与管理节点通信，告知任务的状态和进程。

子节点在运行过程中，首先通过在共享文件系统中把作业的 JAR 文件复制到子节点所在的文件系统，从而实现文件本地化，同时子节点要将应用程序所需要的全部文件进行分布式缓存；然后子节点为任务新建一个本地工作目录，并把相关运行文件解压到文件夹下；最后子节点创建一个作业实例来运行该任务。

3 基于 MapReduce 的分布式 ETL 调度算法

ETL 过程通常需要处理大量的数据，如果能让 ETL 工作流中可以并行执行的活动实例并行执行，将能大大提高整个 ETL 工作流的执行效率[8,9]。基于 MapReduce 并发处理海量数据的并行编程模型，其分布式 ETL 处理核心过程如下：

从 ETL 抽取、转换、加载程序开始，ETL 程序链接 MapReduce库，实现了最基本的 Map 函数和 Reduce 函数，按照图 2 所示顺序执行流程。

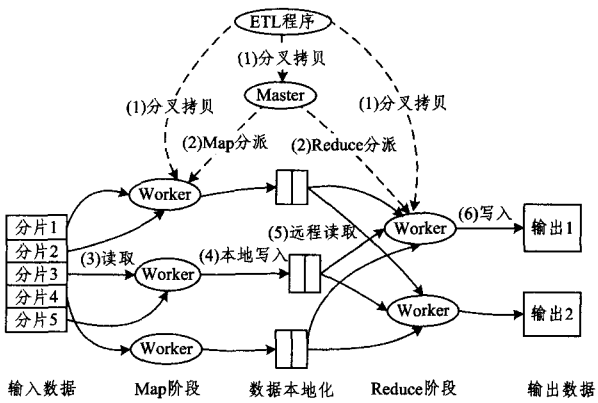

图 2 基于 MapReduce 的分布式 ETL 调度算法

①MapReduce 库先把 ETL 程序的输入文件划分为 M 份（M 为用户定义），每一份通常有 16M 到 64M，如图 2 分了 5 个分片。然后使用分叉拷贝将用户进程拷贝到集群内其他机器上。

②ETL 程序的副本分配的机器中有一个为 Master，其余均为 Worker。Master 是负责调度的，为空闲 Worker 分配作业（Map 作业或者 Reduce 作业），Worker 的数量也由用户指定。

③被分配了 Map 作业的 Worker 开始读取对应分片的输入数据，Map 作业数量由 M 决定，和分片一一对应；Map 作业从输入数据中抽取出键值对，每一个键值对都作为参数传递给 Map 函数，Map 函数产生的中间键值对被缓存在内存中。

④缓存的中间键值对会被定期写入本地磁盘，而且被分

为 R 个区，R 的大小也是由用户定义的，将来每个区会对应一个 Reduce 作业；这些中间键值对的位置会通报给 Master，Master 负责将信息转发给 Reduce Worker。

⑤Master 通知分配了 Reduce 作业的 Worker 它负责的分区在什么位置，当 Reduce Worker 把所有它负责的中间键值对都读过来后，先对它们进行排序，使得相同键的键值对聚集在一起。因为不同的键可能会映射到同一个分区也就是同一个 Reduce 作业，所以排序是必须的。

⑥Reduce Worker 遍历排序后的中间键值对，对于每个唯一的键，都将键与关联的值传递给 Reduce 函数，Reduce 函数产生的输出会添加到这个分区的输出文件中。

⑦当所有的 Map 和 Reduce 作业都完成了，Master 唤醒 ETL 程序，MapReduce 函数调用返回 ETL 程序的代码。

当所有执行完毕后，MapReduce 输出放在了 R 个分区的输出文件中(分别对应一个 Reduce 作业)。用户通常并不需要合并这 R 个文件，而是将其作为输入交给另一个 MapReduce 程序处理。整个过程中，输入数据是来自底层分布式文件系统的，中间数据是放在本地文件系统的，最终输出数据是写入底层分布式文件系统的。

整个流程主要分为 3 个阶段：

• 包括①②，主角是 MapReduce 库，完成拆分作业和拷贝 ETL 程序等任务；

• 包括③④⑤⑥，主角是用户定义的 Map 和 Reduce 函数，每个小作业都独立运行着；

• 扫尾阶段，把作业结果放在输出文件里，供用户其他应用。

4 应用实例与性能分析

4.1 测试环境与评价标准

测试中采用加速比作为本系统的性能指标。加速比是指单机完成某一应用所花时间与多机并行完成某一应用所花时间的比值。系统测试环境如图 3 所示，所有机器通过 1000Mbps 以太网连接在一起，由于实验室条件有限，实验时节点最多为 4 个。

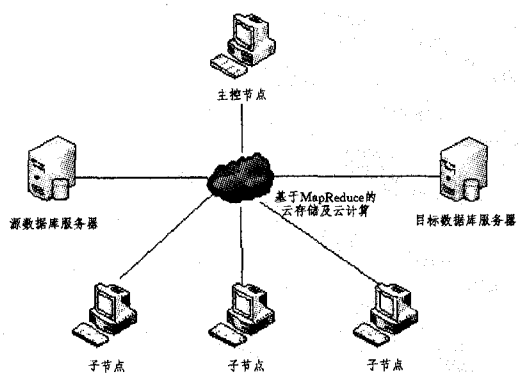

图 3 分布式 ETL 集群实验测试环境拓扑图

计算机 CPU 均为 Intel(R) Xeon CPU W3503 @ 2.40 GHz/2.39GHz，内存为 24GB，操作系统为 Windows 7，Hadoop 版本为 0.20.2。

4.2 实验结果与分析

实验数据来自实验室采集的训练数据，测试结果如下，图 4 为加速比在计算节点变化时的变化曲线，图 5 为加速比在数据集变化时的变化曲线。

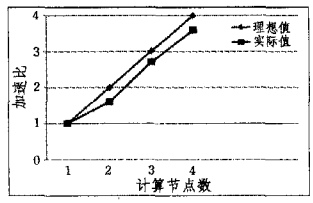

图 4 加速比在计算节点变化时的变化曲线

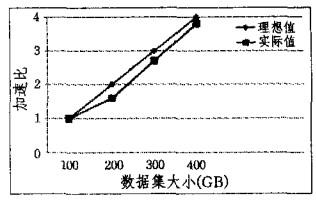

图 5 加速比在数据集变化时的变化曲线

可以看出，随着计算节点数的增加，加速比经历了由上升到下降的趋势。这是因为当计算节点增加后，网络通信的开销会相应增加，由于 MapReduce 采取了基于扫描的处理模式和对中间结果步步物化的执行策略，因此导致较高的 I/O 代价和网络传输，在数据集不是很大的情况下，节点数的增加反而会影响到处理效率。

而随着数据集的加大，加速比是逐渐上升的趋势。这是因为 MapReduce 的设计初衷是面向大数据量和复杂数据的处理，而且往往是一次性处理。基于 MapReduce 平台的分析，无需复杂的数据预处理和写入数据库的过程，而是直接基于平面文件进行分析，并且其采用的计算模式是移动计算而非移动数据，因此可以将分析延迟最小化，故数据集的加大反而使其处理效率相对而言得到提升。

结束语 本文针对传统 ETL 工具集中式执行方式的不足，提出了一种基于 MapReduce 的分布式 ETL 体系结构。该体系结构采用 MapReduce 并发处理海量数据的并行编程模型，结合分布式 ETL 的集群运算方法，实现了集群分布式执行 ETL 流程。实验结果表明，MDETL 具有良好的可靠性和可扩展性，消除了节点单点故障的影响，从而提高了整个 ETL 系统的灵活性和吞吐率，并具有较好的可扩展性和负载平衡性能，提高了执行效率。

参 考 文 献

[1] 许力，等. 并行 ETL 过程的研究与实现[J]. 计算机工程与应用，2009，45(13)：170-172

[2] 王珊，王会举，等. 架构大数据：挑战、现状与展望[J]. 计算机科学，2011，10：1741-1752

[3] Guo Lei-tao，Sun Hong-wei，et al. A data distribution aware task scheduling strategy for mapreduce system[A]// First International Conference on Cloud Computing[C]. Berlin：Springer，2009：694-699

[4] Chen Quan，Zhang Da-qiang，et al. SAMR：A self-adaptive mapreduce scheduling algorithm in heterogeneous environment[A]//Proc of IEEE International Conference on Computer and Information Technology[C]. Los Alamitos：IEEE Computer society，2010：2736-2743

[5] 李建江，崔健，等. MapReduce 并行编程模型研究综述[J]. 电子学报，2011，11：2635-2642

[6] 陈伟江，郭朝珍. 分布式 ETL 中协同机制的研究与设计[J]. 通信学报，2006，11：177-182

[7] 徐艳华，郭朝珍. 基于 MAS 的分布式 ETL 模型[J]. 郑州大学学报，2007，12：118-121

[8] 夏秀峰，等. 一种改进的分布式 ETL 体系结构[J]. 计算机应用与软件，2010，4：174-176

[9] 张亮，夏秀峰. 分布式 ETL 负载均衡策略研究[J]. 计算机与现代化，2011，9：201-204