doi:10.3772/j.issn.1002-0470.2017.04.001

面向多用户环境的 MapReduce 集群调度算法研究①

陈重韬②

(中国科学院计算技术研究所计算机应用研究中心 北京 100190)

(中国科学院大学 北京 100049)

摘 要 针对 MapReduce 集群现有调度策略在多用户环境下无法根据用户的实际资源需求实现动态资源分配的问题,提出了一种基于历史执行信息(HEI)的 MapReduce 集群调度算法——HEI Scheduler。该算法通过建立集群作业执行信息的收集和分析机制,得到各用户组资源需求随时间变化的规律,并以作业实际占用 slot 的时间作为作业占用资源量的衡量标准,进而动态地确定资源池的最小共享资源以及集群剩余资源分配的权值。实验结果表明,执行信息分析机制能够更准确地表征作业对资源的需求,采用集群调度算法 HEI Scheduler 能够有效地缩短作业的整体执行时间。

关键词 MapReduce 集群, 多用户环境, 调度算法, 作业执行信息收集

0 引 言

科技的飞速发展带来了数据规模的爆炸式增长。面向不同的应用场景,众多公司提出了各自的海量数据处理解决方案,为了满足迅速增长的数据处理需求,Google 设计并实现了面向海量数据处理的 MapReduce[1] 编程模型。MapReduce 是海量数据处理领域第一个获得极大成功的编程模型,得到了非常多的研究与应用,从而推动了海量数据处理技术的不断演变。Hadoop[2] 作为 MapReduce 和 Google 文件系统(GFS)[3] 的开源实现已广泛应用于工业界和研究领域,其中 MapReduce 作业调度策略作为一个关键问题,直接关系到海量数据处理的执行效率。

通过分析国内某互联网企业内部使用场景,我们发现,绝大部分作业为定时任务,每个用户组的作业提交都具有明显的规律性,而现有的调度机制并没有充分考虑这一特征,并且不同类型的作业占用系统的资源情况也有所不同。因此,如何改进作业调度算法,使其能更合理地分配资源,成为查询优化的一个重要问题。

本文针对上述场景特点,在现有调度算法基础上,提出了一种基于历史执行信息(historical execution information, HEI)的 MapReduce 集群作业调度算法,该算法通过收集和分析集群中作业的历史执行信息得到各用户组资源需求随时间变化的一般规律,以此为依据确定原有公平调度算法的最低资源限额,并随时间变化进行动态调整,除此之外,以历史执行信息为依据,通过分析各用户组的作业资源占用特征,更准确地描述作业的资源消耗情况,确定各资源池的权重分配剩余资源,最终使集群资源分配更加合理。实验结果表明,运用面向多用户环境的 MapReduce 集群作业调度算法,能够有效地缩短集群作业的整体执行时间,提高集群利用率。

① 国家科技支撑计划(2012BAH46B03),国家自然科学基金(61402473),核高基(2013ZX01039-002-001-001)和中国科学院先导专项(XDA06030200)资助项目。

② 男,1986 年生,博士生;研究方向:海量数据的存储与处理,联系人,E-mail: chenzhongtao@ncic.ac.cn

(收稿日期:2016-05-11)

1 相关工作

1.1 MapReduce 编程模型

作为一种处理海量数据的并行编程模型,MapReduce 以其良好的可扩展性、可用性、容错性[4]得到了学术界和工业界的极大关注,成为目前最为成功的海量数据处理模型之一。MapReduce 编程模型由 Map 和 Reduce 两个阶段组成。Map 阶段主要负责数据的扫描筛选,对输入的键值对 (k_1,v_1) 进行处理,生成中间结果集 $list(k_2,v_2)$;而 Reduce 阶段则按照用户定义,对数据进行归约处理,依据中间结果的键值 k_2 将相关的中间结果数据拉取到对应的 Reduce 进行处理,形成最终结果 $list(k_3,v_3)$ 输出。整个数据处理过程可以表示为

$$\begin{aligned} &Map(k_1,v_1) \rightarrow List(k_2,v_2) \\ &Reduce(k_2,List(v_2)) \rightarrow List(k_3,v_3) \end{aligned} \tag{1}$$

1.2 MapReduce 作业调度算法

在 Hadoop MapReduce 中,作业是由作业跟踪器(JobTracker)来统一调度的。面对多用户环境,JobTracker 会根据系统设置的调度器为各用户分配资源,保证作业的执行。目前,Hadoop MapReduce 集群调度算法有 3 种基本调度算法——FIFO 调度算法、公平调度(Fair Scheduler)算法[5]、能力调度(Capacity Scheduler)算法[6],以及在此 3 种基本调度算法基础上的几种扩展调度算法。下面分别进行分析。

(1)FIFO 调度算法是最简单直观的 MapReduce 集群调度算法,所有作业均被提交到一个队列中,按照提交的先后顺序选择作业执行。FIFO 调度算法实现简单,调度开销小,能够保证作业按照时间顺序运行,但 FIFO 调度容易出现大作业独占集群资源,造成任务的阻塞,从而使小作业往往得不到快速的响应,而一般情况下小作业均有实时性要求。

(2)Fair Scheduler 算法由 Facebook 提出,目标是使每个用户都能公平地共享整个集群的计算能力;Fair Scheduler 调度为每个用户组建立一个单独的作业池,每个作业池都拥有最小共享资源,集群剩余的资源以尽量公平的方式分配给各用户组。当用户组需要的资源小于最小共享资源时,空闲资源可根据公平原则分配给相应的用户组进行使用,待该用户组需求资源量增加并超过等待时间时,再以抢占的方式将属于自己的资源占用。Fair Scheduler 算法,保证了不同的用户能够公平地获取集群资源,然而它只专注于用户的公平而忽视了用户的需求变化以及不同作业之间的差异。

(3)Capacity Scheduler 算法由雅虎公司提出,它由多个队列组成,每个队列可配置一定的资源;与公平调度相似,Capacity Scheduler 允许各个队列分享已经被分配但处于空闲状态的资源,但不支持抢占,只有当前作业释放相应资源且原队列有新的资源请求时,才会将资源分配回所属队列。Capacity Scheduler 算法避免了同一用户的作业独占集群资源的情况,并且能够有效地处理各类型的作业,但是它也未考虑不同的作业特征与资源分配之间的动态关系。

在上述 3 种基本调度算法基础上又产生了如下几种扩展调度算法:

(4)在 Fair Scheduler 算法的基础上,Isrand 等人提出了 Quiency 调度器[6],在保证集群中的作业尽可能使用本地数据的基础上,得到公平的调度机会。

(5)通过形式化定义公平调度算法分配资源和作业调度的过程,Wolf 等人提出了一种启发式的资源调度分配策略[7],通过对资源调度参数进行优化,力求更合理地进行资源分配。

(6)TDWS[8]按照业务类型将系统资源分为 3 个组,并根据系统的负载,在组之间对计算资源进行重新分配。

(7)还有很多其他关于 MapReduce 集群任务调度研究的工作[9-12],它们都从满足服务质量约束以及动态资源分配两个方面对调度进行优化。

综上所述,虽然现有的调度算法在保证作业的执行以及资源的合理分配上做了一定的优化,但是都没有充分考虑各用户组资源需求随时间的规律变化,尤其是在某一时间段内,某一个或多个资源组出现作业集中提交的情况,造成资源分配不合理。除此之外,对于作业所需资源的衡量标准,现有调度算法大多以其占用的 slot 数量来衡量,并未考虑不同

作业类型占用计算资源的差异。

2 面向多用户环境的 MapReduce 集群调度算法

前述已知,现有调度算法并未考虑各用户组资源需求的规律变化,也没有对所提交的作业类型进行详细区分,仅以占用 slot 的数量表征作业资源的占用情况,即并没有按照作业的实际资源需求类型进行区别标识,容易造成资源分配不公平情况的出现。本文提出的基于历史执行信息(HEI)分析的 MapReduce 集群调度算法,通过分析集群中不同用户组提交作业的历史执行信息,以达到如下两个目的:

(1)找到各用户组提交作业的规律,也即用户组对资源需求的规律;

(2)找到不同用户提交的作业的权值,更准确地估算剩余作业需要的计算资源。

2.1 作业执行信息收集

首先,为了更好地分析和监控 MapReduce 作业的执行情况,我们设计了 MapReduce 集群作业执行信息收集系统,如图 1 所示。该系统完全独立于原有 Hadoop,不需要对 Hadoop 自身代码进行修改,且不会对原有 Hadoop 系统的运行造成额外的影响。该系统功能为:对集群中作业的执行信息进行收集和存储,并提供相关信息的检索服务。通过对执行信息进行分析,能够更好地了解日常任务的运行状况,以及 MapReduce 集群的健康状况,为 MapReduce 集群的优化提供数据依据。

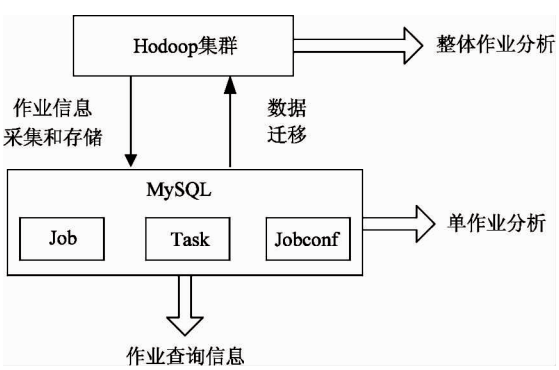

图 1 MapReduce 集群作业执行信息收集系统示意图

由图 1 可知,作业执行信息收集通过部署于 Hadoop 中 JobTracker 端的监控线程对每个作业生成的 Jobconf 和 JobHistory 文件进行解析来完成,而后对得到的信息进行格式化并将数据插入到相应的 MySQL 数据库表中。在数据库中需创建三张表,分别为 Job 表,Task 表以及 Jobconf 表,这些表记录了作业的基本信息以及运行时信息,用户可以对存储在 MySQL 中的数据进行查询,了解作业执行的详细信息,亦可对数据按照自身需求进行分析处理。由于集群中每天运行的 MapReduce 作业数量以及相应的 Task 数量都很庞大,使得 MySQL 服务器的存储压力过大,因而系统会以天为单位定期将存储于 MySQL 数据库中的历史信息迁移到 Hive 集群中做永久存储,作为整体作业分析的原始数据。

2.2 作业执行信息分析

在作业执行信息分析过程中,本文定义了一些指标,用来评估 MapReduce 集群的运行状况,并为集群调度算法的优化提供数据依据。由于不同类型作业中 Task 的执行时间不同,即每个 Task 消耗的计算资源不同,因而在本文中以作业中包含的所有 Task 消耗计算资源之和作为该作业的资源需求量,而不仅仅使用占用 slot 数量去表征。

某 Task 消耗的计算资源 R_i 可以表示为

$$R_i = N^i_{\text{slot}} \times T^i_{\text{Ocup}} \tag{2}$$

其中 R_i 表示第 i 个 Task 消耗的计算资源,N^i_{slot} 表示第 i 个 Task 占用的 slot 数量,T^i_{Ocup} 表示第 i 个 Task 的运行时间。由于现有系统中每个 Task 执行均占用一个 slot,即 $N^i_{\text{slot}} = 1$,则系统中第 i 个 Task 的计算资源使用情况便可以用执行时间来衡量:

$$R_i = T^i_{\text{Ocup}} = T^i_{\text{finish}} - T^i_{\text{start}} \tag{3}$$

其中 T^i_{start} 表示第 i 个 Task 执行开始的时刻,T^i_{finish} 表示第 i 个 Task 执行结束的时刻。因此,某作业的计算资源使用情况 R_{total} 即为该作业包含的所有 Task 消耗的计算资源之和,可以表示为

$$R_{\text{total}} = \sum R_i = \sum T^i_{\text{Ocup}} = \sum T^i_{\text{finish}} - T^i_{\text{start}} \tag{4}$$

依据上述公式及描述,我们将作业及其内部所包含任务的计算资源消耗量转化为相应的执行时间,从而更加贴近集群中各作业真实的资源占用。

经过对每个 Map Task 的处理数据量以及处理时间进行进一步的分析,我们发现,Map Task 的执行时间与其所处理的数据量基本成正比例关系,这是由 Map 阶段的处理性质决定的。在 Map 阶段处理数据主要以数据的筛选等简单处理为主,不会有过于复杂的计算出现,因而我们提出了单位数据计算资源消耗量的概念,来描述 Map Task 的计算资源消耗。

为了更好地描述 Map Task 的执行性能,我们引入单位数据计算资源消耗量的概念,即相应 Task 在处理单位数据量的数据时消耗的计算资源量。某 Task 的单位数据资源消耗量可以表示为

$$D_i = \frac{R_i}{M_i} \tag{5}$$

其中 D_i 表示第 i 个 Task 单位数据资源消耗量,R_i 表示第 i 个 Task 消耗的计算资源,M_i 表示第 i 个 Task 处理的数据量。

根据式(4),我们可以得到某作业的单位数据资源消耗量,可以由该作业的所有 Task 的执行时间除以 Task 处理的数据量来等价描述,即

$$D_{\text{average}} = \frac{R_{\text{total}}}{\sum M_i} = \frac{\sum R_i}{\sum M_i} = \frac{\sum T^i_{\text{Ocup}}}{M} \tag{6}$$

其中 D_{average} 表示某作业的单位数据资源消耗量,R_{total} 表示该作业中所有 Task 占有计算资源,M 表示该作业中 Task 处理的数据量。

在 MapReduce 中,Task 分为 Map Task 和 Reduce Task 两类,并且由前文分析可知,Map Task 主要负责对原始数据进行处理和筛选类的操作,并不存在过于复杂的计算过程,Map Task 的执行时间与其处理的数据量基本成比例关系,因此根据式(6)我们得出 $D^{\text{Map}}_{\text{average}}$ 如下式所示:

$$D^{\text{Map}}_{\text{average}} = \frac{\sum T^p_{\text{Ocup-Map}}}{P} \tag{7}$$

其中 $\sum T^p_{\text{Ocup-Map}}$ 表示该作业中所有 Map Task 的执行时间之和,P 表示 Map 阶段由 HDFS 中读取的总数据量。

则某作业在 Map 阶段需要的计算资源可以表示为

$$R_{\text{Map}} = D^{\text{Map}}_{\text{average}} \times P' \tag{8}$$

其中 R_{Map} 表示某作业在 Map 阶段需要的计算资源量,P' 表示 Map 阶段待处理的数据量。

以上完成了对于 Map Task 资源消耗量的衡量标准的选择。

对于 Reduce Task,由于它的执行时间由数据拉取、数据排序和数据计算三部分构成,首先,现有系统中 Reduce Task 的启动并没有考虑数据本地性的原则,因而 Reduce Task 的数据拉取时间受网络等影响都较大,并不能完全确定其与数据量的关系,其次,数据计算过程一般较为复杂,占用整个 Reduce 过程的大部分时间,其时间开销并不一定与数据量完全成比例关系。

经过以上分析,为了更好地预测作业的执行时间,我们根据式(4),最终采用 Reduce Task 的平均执行时间来衡量其计算资源消耗量,即

$$R^{\text{reduce}}_{\text{average}} = T^{\text{reduce}}_{\text{average}} = \frac{\sum T^l_{\text{Ocup-reduce}}}{L} \tag{9}$$

其中 $R^{\text{reduce}}_{\text{average}}$ 表示 Reduce Task 平均消耗计算资源量,$T^{\text{reduce}}_{\text{average}}$ 表示 Reduce Task 的平均执行时间,$\sum T^l_{\text{Ocup-reduce}}$ 表示作业中 Reduce Task 整体时间,L 表示 Reduce Task 的个数。

作业执行信息分析主要分为两个层面:针对单个作业的分析以及集群整体作业提交信息的分析。

对于单个作业的分析,发生在监控线程完成某个作业的 Jobconf 和 JobHistory 文件的解析并成功插入到相应数据库表之后。数据分析线程会对该作业的执行信息进行进一步处理,处理过程需要的信息包括 Task 的开始时间和结束时间、Task 输入数据量以及 Task 的执行节点,具体处理流程(如图 2 所示)如下:

(1)首先在 Task 表中找到该作业对应的所有 Task,包括所有的 Map Task 和 Reduce Task。

(2)依据式(2)、式(5)和式(9)得到该作业中各 Task 消耗的资源量、各 Map Task 的单位数据消耗资源量,以及 Reduce Task 的平均计算资源消耗量,并将得到的数据插入到相应 Task 表中,以备后续作为节点速度的衡量标准。

(3)当对涉及的所有 Task 的计算都完成之后,计算所有 Map Task 的单位数据消耗资源量的均值,

然后计算出所有 Map Task 的单位数据消耗资源量的标准差,找到偏差超过阈值的 Task,将其所在的执行节点标记为可能的慢节点,该处阈值为经验值。

(4)依据式(7)在去除检测出的单位数据消耗资源量超过阈值的 Task 的情况下,计算出该作业的 Map 单位数据计算资源消耗量,该值可用来对集群中该作业类型的 Map 的处理速度进行衡量,同时依据式(4),计算出该作业需要的集群资源。

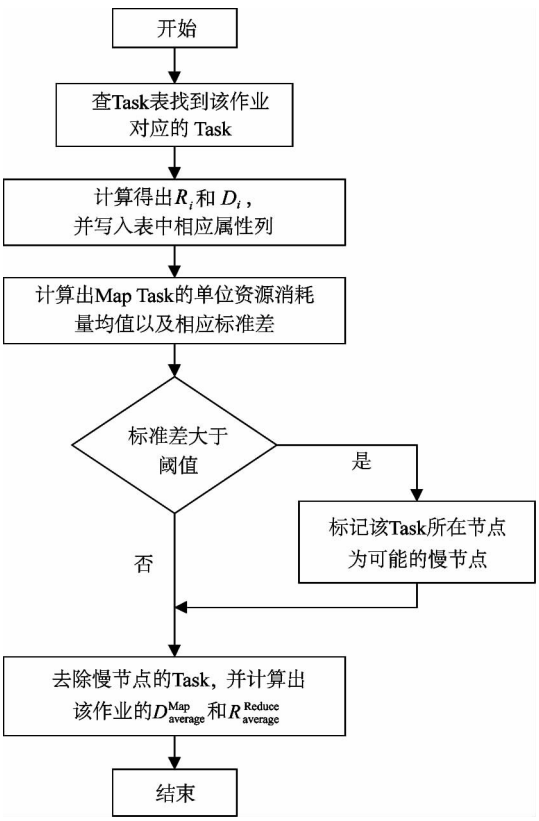

图 2　单个作业分析执行流程

对于集群整体作业提交信息的分析,会在一天的数据成功迁移到 Hadoop 集群中后进行。分析流程如下:

(1)按照自然小时和用户组对作业的提交进行统计,依据式(4)将某小时内用户组提交的作业需要的计算资源需求量,转化为该用户组提交的作业的所有 Task 的执行时间之和。

(2)通过对三张表进行操作,找到相应的 Task,并计算所有 Task 执行时间之和,即为该用户组该小时需要的计算资源。

2.3　面向多用户环境的 MapReduce 集群调度算法

经过对集群中各用户组作业的提交进行长时间的跟踪分析发现,集群中大部分作业均为定时任务,作业提交有明显的小时特征,即同一天内按照小时对比,每小时提交的作业的数量和类型都有不同,但是天与天之间对比,连续两天中同一用户组相同小时内提交的作业数量和类型基本相同。在现有的 MapReduce 集群调度算法及相关优化中均没有对这一特点的针对性的考虑。除此之外,在现有的调度算法中,仅以作业占用的 slot 数量来表征计算资源的需求量,并没有考虑各种作业占用计算资源上的差异。在集群中会出现作业占用 slot 数量少,但占用时间长,或某个作业占用 slot 数量多,但每个 Task 占用时间短的情况出现,这些情况现有的调度算法均不能很好表征。

基于以上因素,本研究提出了面向多用户环境的基于历史执行信息的 MapReduce 集群调度算法。此算法根据各用户组作业提交的规律,由历史数据中某小时的资源需求量来估算未来相应小时的计算资源需求量。同时,在计算某作业需要的计算资源的过程中,对不同的作业包含的 Task 的执行特点进行分析,以相应 Task 实际占用 slot 的时间作为表征,计算 Task 需要的计算资源,使之更加贴近作业的实际计算资源需求。在此基础上,采用两种不同的统计方式对作业中 Map Task 和 Reduce Task 的计算资源占用进行衡量,更好地对计算资源的消耗进行估算。

历史执行信息(HEI)调度算法以 Fair Scheduler 算法为基础,资源分配的步骤如下:

(1)确定各资源池的最小资源共享量

新的小时开始,查找历史数据,更新相关资源池该小时需要的历史计算资源,分别计算出所有资源池需要的计算资源量,最终按比例求得每个资源池的最小共享资源量 minshare;

(2)集群剩余资源分配

作业执行的过程中,某一类 Job 的剩余需求资源量由两部分组成,分别是 Map Task 的单位数据计算资源消耗量乘以待处理的数据量,以及Reduce

Task 的平均计算资源消耗量乘以待运行的 Reduce Task 的数量,两部分相加即为所求。而某用户组的剩余集群资源分配量等于该用户组待处理数据消耗资源量占所有用户组待处理数据消耗资源量的比重乘以集群剩余可分配资源量。

新的集群调度算法的优点在于依据历史信息,动态地调整用户组的最小资源量,并且在计算作业需求的计算资源量时,以作业中包含的所有 Task 需要的计算资源量之和来表征作业对于资源的需求,而不仅仅是用占用 slot 数量去表征。这种方法更接近实际,能够更准备地表征作业对于资源的实际需求。

3 实验结果与分析

3.1 实际部署和运行情况

作业执行信息收集和分析系统 MRDoctor 以及 HEI 调度算法已于 2013 年在国内某互联网企业上线。其中 MRDoctor 系统为用户提供了便捷的作业执行信息的查询服务,用于对多用户环境下 MapReduce 集群运行状况进行诊断和分析,已经成为用户发现和解决业务问题的重要手段。利用 MRDoctor 系统对用户组作业提交规律的统计,也为各用户组错峰提交作业,从而合理利用集群计算资源提供了很好的依据。除此之外,HEI 调度算法的使用,有效地提高了应用集群在高峰时段的集群利用率,缩短了作业的整体完成时间。

3.2 作业处理性能对比

测试平台由 20 台服务器组成,服务器硬件为每个节点四个频率为 2.2GHz 八核 CPU,32GB 内存和 1TB 的存储容量。节点运行的操作系统为 Red Hat 6.2,内核版本号 2.6.32,节点间通过千兆网互连。其中,1 台节点作为 Hadoop 集群的 master 节点,其余 19 台节点作为数据节点。实验中运行的 Java 环境为 1.7.0,使用的 Hadoop 版本为 0.20.2,每个节点配置 8 个槽位。

通过采用增加某用户组提交的作业数量计算出最终作业的完成时间的方法来验证本文提出的 HEI 调度器的有效性。其中,测试所提交的作业类型为占用 slot 数量少,但 Map 和 Reduce 占用 slot 的时间长的作业,这样更能体现出 HEI 调度算法在计算作业的实际资源需求量上的作用,选择与 Fair Scheduler 调度算法进行比较,结果如图 3 所示。

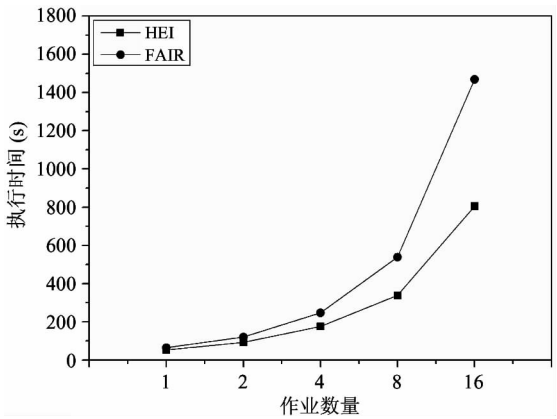

图 3 用户组作业完成时间对比

由结果可知,随着作业数量的增加,HEI 调度算法作业完成时间均比 Fair Scheduler 算法短,这是因为在作业提交后,HEI 调度算法会根据该作业类型的历史执行信息,来估算作业的 slot 的占用时间,因此更加接近作业的实际计算资源需求,通过进一步增大该资源池应该占用的系统资源,保证了相应的作业资源的分配。而 Fair Scheduler 算法由于仅以作业占用 slot 数量来对作业的资源需求进行衡量,因而分配给该作业的资源远远少于该组需要的计算资源。

如表 1 所示,测试作业临时提交到某资源池后,资源池分配资源与实际资源的对比。实验过程为:首先保证集群中运行相同的任务,并将 HEI 调度算法和 Fair Scheduler 算法的某相同资源池的最小共享资源量固定为该资源池的实际需求量,通过向该资源池中临时加入作业的方式来比较 FAIR 调度器对于该资源池计算资源的分配,如表 1 所示,其中响应时间为从作业的提交到第一个 Map 开始执行的时间。

表 1　作业临时提交模式下各算法效率对比

	HEI	FAIR
资源池分配资源(slot 数量)	104	76
实际需求(slot 数量)	130	130
响应时间(s)	3	6

由结果可知,HEI 调度器通过作业占用时间权值的计算,为该资源池分配了更多的资源,更加贴近作业的实际资源需求,因而 HEI 调度算法能够更好地表征用户的实际需求,达到缩短作业执行时间的目的。

4　结　论

面对多用户环境对于集群调度算法提出的挑战,以及现有调度算法的不足,本文提出了一种基于历史执行信息的 MapReduce 集群调度算法。通过建立作业执行信息收集和分析机制,得到各用户组资源需求随时间变化的规律,并确定不同作业对资源需求的权值,进而动态地确定资源池的最小共享资源以及集群剩余资源分配的权值。实验结果表明,本文提出的执行信息分析机制能够更准确地表征作业对于资源的需求,采用基于历史执行信息分析的 MapReduce 集群调度算法能够有效地缩短作业的整体执行时间。未来将进一步细化参数的确定,力求对作业需求资源的描述与实际更加贴合,同时将研究最小共享资源量占总的集群资源量的比值变化对集群整体运行效率的影响。

参考文献

[1] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. *Communications of The ACM*, 2008, 51(1): 107-113

[2] Bialecki A, Cafarella M. Hadoop: A Framework for Running Applications on Large Clusters Built of Commodity Hardware, http://lucene. apache. org/hadoop: Apache, 2008

[3] Ghemawat S, Gobioff H, Leung S. File and storage systems: The Google file system. In: Proceedings of the ACM Symposium on Operating Systems Principles, Bolton Landing, USA, 2003, 37:29-43

[4] 李建江, 崔健, 王聃等. MapReduce 并行编程模型研究综述. 电子学报, 2011, 39(11):2635-2642

[5] Zaharia M, Borthakur D, Sarma J S, et al. Job scheduling for multi-user MapReduce clusters: Technical Report. EECS Department, University of California, Berkeley, USA, 2009

[6] Isard M, Prabhakaran V, Curr B, et al. Quincy: fair scheduling for distributed computing clusters. In: Proceedings of the ACM Symposium on Operating Systems Principles 2009, Big Sky, USA, 2009. 261-276

[7] Wolf J, Rajan D, Hildrum K, et al. FLEX: a slot allocation scheduling optimizer for MapReduce workloads. In: Proceedings of the 11th International Conference on Middleware, Bangalore, India, 2010. 1-20

[8] Zhao Y R, Wang W P, Meng D, et al. TDWS: A Job Scheduling Algorithm Based on MapReduce. In: Proceedings of the IEEE, International Conference on Networking, Architecture and Storage, Xiamen, China, 2012. 313-319

[9] Tian C, Zhou H, He Y, et al. A dynamic MapReduce scheduler for heterogeneous workloads. In: Proceedings of the 8th International Conference on Grid and Cooperative Computing, Lanzhou, China, 2009. 218-224

[10] Kc K, Anyanwu K. Scheduling Hadoop jobs to meet deadlines. In: Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science, Indianapolis, USA, 2010. 388-392

[11] Tian F, Chen K. Towards optimal resource provisioning for running MapReduce programs in public clouds. In: Proceedings of the International Conference on Cloud Computing, Washington DC, USA, 2011. 155-162

[12] Sandholm T, Lai K. MapReduce optimization using regulated dynamic prioritization. *Acm Sigmetrics Performance Evaluation Review*, 2009, 37(1):299-310

Research on a job scheduling algorithm for multi user MapReduce clusters

Chen Zhongtao
(Center of Computer Application Research, Institute of Computering Technology,
Chinese Academy of Sciences, Beijing 100190)
(University of Chinese Academy of Sciences, Beijing 100049)

Abstract

To solve the problem that the existing MapReduce scheduling strategy cannot realize dynamic resource allocation according to user's actual resource demand in a multi-user environment, an algorithm for MapReduce clusters scheduling based on historical execution information (HEI), called the HEI scheduler, was proposed. The algorithm obtains the rules of the variation of each user group's resource demand with time by establishing the mechanism for collection and analysis of cluster operation's execution information, and uses operation's actual slot occupying time to measure the occupied resource of operation to dynamically determine the minimum shared resource and weights of cluster remaining resource distribution. The experimental result indicates that the proposed execution information analysis mechanism can describe the resource demand of operation exactly. The MapReduce cluster scheduling algorithm based on historical execution information can effectively reduce the overall operation execution time.

Key words: MapReduce cluster, multi-user environment, scheduling algorithm, job implementation information collection