

一种基于动态资源采集的 Hadoop 作业调度算法

彭志伟, 谷建华

(西北工业大学 计算机学院, 陕西 西安 710072)

摘 要: 为了克服 Hadoop 中基于槽的资源模型对集群资源的浪费, 提高 Hadoop 集群的资源利用率, 提出了一种基于动态资源采集的 Hadoop 作业调度算法. 通过在作业运行过程中动态采集部分任务的 CPU、内存和 IO 的资源利用率来评估同一作业中其他任务的资源需求, 然后根据任务的实际资源需求以及 TaskTracker 节点的负载情况进行任务调度, 以充分利用各 TaskTracker 节点的计算资源. 通过对比实验表明该调度算法能明显提高集群的资源利用率, 缩短作业的完成时间.

关键词: Hadoop; MapReduce; 调度; 资源需求

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2014)04-0064-04

A Scheduler for Hadoop Based on Dynamic Resource Collection

PENG Zhi-wei, GU Jian-hua

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: In order to overcome the problem of wasting of resources in hadoop cluser based on the slot resource model, a scheduler based on the dynamic resource collection has been proposed. According to collect the CPU, Memory and IO resource utilization of part of tasks in the hadoop job, We estimate the resource demand of the rest tasks in the job, and then schedule these tasks according to their resource demand and the load of every Tasktracker. The contrast experiment shows the scheduler we proposed can increase the resource utilization of the hadoop cluser and reduce the completion time of the Hadoop job.

Key words: Hadoop; MapReduce; schedule; resource demand

1 引言

随着互联网技术的迅速发展,越来越多的企业面临着海量数据处理的问题. Hadoop^[1]作为一个能在大规模集群上运行大型数据处理应用的开放源代码框架,实现了 MapReduce^[2]编程模型,并提供了一个高可靠分布式文件系统 HDFS^[3],被广泛用于企业的大数据处理中. 作业调度^[4]是 Hadoop 系统的一个重要的组成部分,它直接影响到作业的完成时间和集群的资源利用率.

目前 Hadoop 的调度算法大多以槽作为调度的基本单元,一个槽运行一个任务. 这种槽的机制存在着明显的缺点. 首先,槽不能正确的反映运行作业的资源需求. 在一个多用户的集群环境下,不同的用户

提交的作业类型不同,这些作业对 cpu、内存和 IO 等资源的需求不一样,基于槽的调度机制会浪费集群的资源. 其次,TaskTracker 节点槽的数量配置完成后,在集群运行期间不能修改,这是一种静态机制. 节点在不同时刻运行的作业往往不同,这些作业的资源需求也不同,这会造成节点在某些时刻负载较重,而某些时刻负载较轻,不利于充分利用节点的计算资源.

现有的 Hadoop 的调度算法大多基于这种静态槽的机制,包括 FIFO, FairScheduler^[5], Capacity^[6]等. 文献[7]中作者提出了一种基于负载的动态槽的调度算法,它根据节点的负载情况动态的调整槽的数量. 文献[8]提出一种根据作业资源利用率进行任务分配的算法. 该算法使用静态的方式来获取作业

的资源利用率.本文提出了一种通过作业运行过程中动态采集已运行任务资源利用率来评估未运行任务资源需求然后依据任务资源需求进行调度的 Hadoop 调度算法 DRCS(Dynamic Resource Collection Based Scheduler).

2 DRCS 调度算法设计与实现

提出一种基于动态资源采集的任务调度算法(DRCS).我们的设计目标是通过动态测量作业中任务的实际资源需求以及 TaskTracker 节点的实际负载情况来合理的分配任务.调度算法主要分为 2 个模块:资源采集模块和任务调度模块.

2.1 资源采集模块

资源采集模块主要完成对作业中正在运行任务的资源使用情况的采集,并估计未执行任务的资源需求.

三元向量 S_t 表示任务 t 的资源利用率,

$$S_t = (\text{CPU}_t, \text{MM}_t, \text{IO}_t)$$

式中, CPU_t 是任务 t 的 CPU 利用率, MM_t 是 t 的内存利用率, IO_t 是 t 的 io 利用率.在任务 t 的运行过程中,我们通过 top 以及 iotop 工具周期性的测量它的资源使用情况,并通过 TaskTracker 的心跳包报告给 JobTracker, JobTracker 对采集到的数据进行处理,计算出作业中正在运行任务的平均资源利用率.计算公式如下:

$$S_t = \frac{\sum_{i=0}^n S_{t,i}}{n} \quad (1)$$

式中, n 是采样点数量, $S_{t,i}$ 是任务 t 在采样点 i 采集到的资源利用率.当任务 t 运行完成时,我们以 S_t 来作为该任务的资源需求,也就是说我们以任务运行过程中各个采样点的资源利用率的平均值作为该任务对 cpu,内存以及 io 资源需求.同一作业中的 map 任务具有相同的程序代码以及数据分块大小,因此可以利用已经运行完成的任务的资源需求来评估作业中其他任务的资源需求.这里我们以已运行任务的资源需求的平均值来估计未运行任务的资源需求.计算公式如下:

$$D_m = \frac{\sum_{t \in F_m} S_t}{|F_m|} \quad (2)$$

式中, D_m 表示作业中 map 任务的资源需求. F_m 表示作业中已运行的 map 任务的集合.

对于 reduce 任务,和 map 任务一样,用 D_r 表示

reduce 任务的资源需求, F_r 表示作业中已运行完的 reduce 任务的集合

$$D_r = \frac{\sum_{t \in F_r} S_t}{|F_r|} \quad (3)$$

D_m 和 D_r 都是一个三元向量,分别表示作业中 map 任务和 reduce 任务对 cpu,内存和 io 的需求.计算出了作业中任务的实际资源需求,就可以按照实际的资源需求进行任务的调度的了,而不用考虑 TaskTracker 槽的数量.

2.2 任务调度模块

本模块主要完成任务的调度.当有 TaskTracker 请求任务时,根据 2.1 节中资源采集模块所测量出的任务资源需求以及 TaskTracker 的可用资源情况进行任务分配.

(1) 资源池

在任务调度模块,我们记录了所有 TaskTracker 的可用资源信息,称之为资源池.这些资源信息包括节点可用的 CPU,内存以及 IO.这些信息是通过 TaskTracker 的心跳包获取.当有 TaskTracker 通过心跳包请求任务时,在作业队列中找一个不使其资源过载的任务分配给他,同时更新资源池中该 TaskTracker 的可用资源信息.当任务运行完成以后,释放该任务占用的资源.

(2) 任务分配

MapReduce 作业中任务的分配包含 map 任务的分配和 reduce 任务的分配.在本算法中两种任务的分配方式基本相同,下面的介绍中不予区分.

任务的调度分两个阶段进行,第一阶段是测试任务的调度,通过调度一部分任务运行,测量其资源利用率的平均值估计未运行任务的资源需求.这一阶段任务由于无法知道其实际的资源需求,因此调度任然按照原有的槽的方式进行.第二阶段,对作业中剩下的任务进行调度.根据第一阶段计算到的任务资源需求对作业中剩余的任务进行调度.图 1 给出了任务调度的流程图.

当 TaskTracker TT 通过心跳包向 JobTracker 请求作业时,调度器扫描作业队列中的所有作业,如果发现该作业的资源需求没有测量出来,那么此时进行第一阶段任务调度,即基于槽的分配策略.我们用 R_{slot} 代表一个槽所代表的资源,它的大小通过配置文件指定,这和传统的槽的配置方式相同.

如果当前请求任务的 TaskTracker TT 当前可用的资源模大于 R_{slot} ,那么我们就将任务分给 TT,

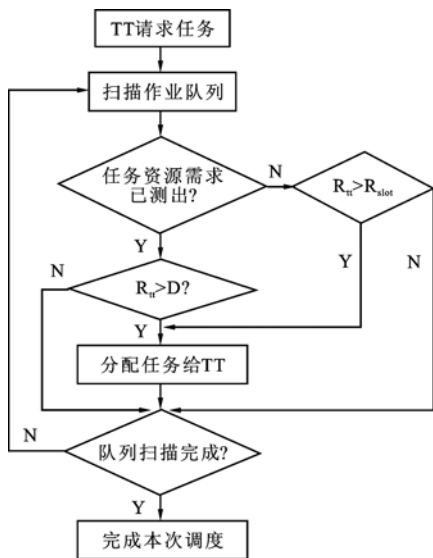


图1 调度流程

同时更新资源池中 TT 的可用资源,方法是用当前可用资源减去 R_{slot} . 任务分配出去后,调度器通过 TaskTracker 反馈回来的心跳包根据 2.1 节中的方法计算其资源需求. 当作业中任务的资源需求确定后,这时候将进行第二阶段调度,将按照其实际的资源需求进行任务的分配. 如果当前 TT 的可用资源 R_u 大于任务的资源需求 D ,那么就将任务分配给 TT, 同时更新资源池中 TT 的可用资源. 否则继续扫描队列中的其他作业. 一直到队列中所有的作业处理完成,本次任务调度才完成.

(3) 任务分配中的死锁问题

在传统的 Hadoop 作业调度中, map 任务和 reduce 分别运行在 map 槽和 reduce 槽中,这两种槽只能运行相应类型的任务,因而 map 任务和 reduce 任务不存在资源的竞争. 在 DRCS 算法中,我们不再区分 map 槽和 reduce 槽,而采用统一的资源池来管理节点中的资源,因而 map 任务和 reduce 任务之间存在资源竞争. 我们知道,同一作业中 map 任务和 reduce 任务的执行存在依赖关系,即 reduce 必须采集到所有 map 任务输出的中间数据才能进行接下来的 sort 和 reduce 阶段的操作. 这样就会出现一种情况,即一个作业中还有部分 map 没有调度到节点上执行,而此时 reduce 任务已经消耗完了资源池中的所有资源,导致剩下的 map 任务无法调度到任何节点上执行. 然而在 map 任务没有全部执行完毕的情况下,reduce 任务没法继续执行,reduce 任务将一直占有这资源却无法往下运行,这样就导致了作业的死锁. 有以下几种方法可以解决死锁问题:方法一,发生死锁时,杀死一部分导致死锁的 reduce

任务;方法二,分配任务时,在 map 任务没有分配完之前,不分配 reduce 任务;方法三,分配 reduce 任务时,检查当前资源池中可用资源大小可以容纳还未分配的 map 任务. 方法一是在死锁发生后所采取的策略,这种方式会浪费 reduce 任务执行部分结果,最终浪费系统的资源. 方法二可以有效地避免死锁,但是在所有 map 任务都分出去之后才分配 reduce 任务,导致 map 任务和 reduce 任务串行执行,减低了任务的并行度,不利于缩短作业的运行时间. 方法三可以有效地避免死锁的发生,同时也不存在方法二中的问题,是一种较好的解决方式,因此本调度算法采用的是这种方式.

3 实验验证

3.1 实验环境

实验环境是一个由 8 个节点组成的 Hadoop 集群. 其中一个作为 JobTracker, 另外 7 个作为 TaskTracker. 所采用的 Hadoop 版本是 1.1.1. 每个节点的配置为 2 GB 内存, 1 个 Intel Core i3 处理器, 1 个 500 GB SATA 硬盘.

采用的测试程序是 Hadoop 自带的基准测试程序 sort. 它是一个对输入数据进行并行排序的 MapReduce 程序. sort 程序在提交时可以通过命令行参数指定其 reduce task 的数量. 分别测试了当 reduce 任务数从 100 到 1 000 变化时,调度算法 DRCS 以及 Hadoop 基于槽的调度算法 FIFO 的性能.

3.2 实验结果

在 sort 程序的运行过程中,我们记录了不同时刻资源采集模块所计算的任务的资源需求,通过实验表明,DRCS 能明显提高集群各节点的资源利用率,提高作业中任务的并发程度,从而缩短作业完成时间.

4 结束语

作业调度是 Hadoop 框架的一个重要组成部分,提出了一种基于任务资源需求的调度算法. 调度分两个阶段进行,第一阶段是任务资源利用率采集和计算. 这一阶段任务的调度任然按照原有的槽的方式进行. 第二阶段,根据第一阶段计算到的任务资源需求进行任务的分配. 最后,通过一系列对比实验,验证了调度算法的性能.

当前,本文调度算法依据已经运行的任务的资源利用率的平均值来估计未运行任务的资源需求.

如果任务运行的不同时刻其资源需求变化较大,这时平均值并不能真实的反映任务的资源需求.因此可能会导致分配出去的任务过多而导致节点过载.在以后的工作中,我们将研究如果解决该问题.

参考文献:

- [1] Apache Software Foundation. Hadoop[EB/OL]. [20013-02-18]. <http://hadoop.apache.org/>
- [2] Apache Software Foundation. Hadoop MapReduce Tutorial [EB/OL]. [2013-03-05]. <http://hadoop.apache.org/mapreduce>.
- [3] Apache Software Foundation. Hadoop Distributed File System (HDFS) Architecture [EB/OL]. [2013-03-05]. <http://hadoop.apache.org/core/docs/current/hdfsdesign.html>.
- [4] Xia Yang, Wang Lei, Zhao Qiang, et al. Research on Job Scheduling Algorithm in Hadoop[J]. Journal of Computational Systems, 2011, 16(7): 5769-5775.
- [5] Matei Zaharia. The Hadoop Fair Scheduler[EB/OL].

[2013-04-13]. <http://developer.yahoo.net/blogs/hadoop/FairSharePres.ppt>.

- [6] Apache Software Foundation. Hadoop's Capacity Scheduler [EB/OL]. [2013-03-09]. http://hadoop.apache.org/core/docs/current/capacity_scheduler.html.
- [7] Mao Hong, Hu Shengqiu, Zhang zhenzhong, et al. A Load-Driven Task Scheduler with Adaptive DSC for MapReduce[C]//Green Computing and Communications, 2011 IEEE/ACM International Conference on. IEEE/ACM, 2011.
- [8] Jorda Polo. Resource-aware Adaptive Scheduling for MapReduce Clusters[EB/OL]. [2013-03-06]. <http://people.ac.upc.edu/dcarrera/papers/MIDDLEWARE11.pdf>.

作者简介:

彭志伟 男, (1988—), 硕士研究生. 研究方向为分布式计算.

谷建华 男, (1965—), 博士, 博士生导师. 研究方向为高性能计算和分布式计算.

(上接第 63 页)

参考文献:

- [1] Hsien-Po Shiang, Van Der Schaar. M. information constrained resource allocation in multicamera wireless surveillance networks[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2010, 20(4): 505-517.
- [2] Wu D, Ci S, Luo H, et al. Video surveillance over wireless sensor and actuator networks using active cameras[J]. IEEE Transactions on Automatic Control, 2011, 56(10): 2467-2472.
- [3] Thi Thi Zin, Pyke Tin, Takashi Toriu. A Markov random walk model for loitering people detection[C]//Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Darmstadt: IEEE, 2010: 680-683.
- [4] 张起贵. 基于视频序列的徘徊检测跟踪算法的研究与实现[D]. 太原: 太原理工大学, 2011.
- [5] Calderara S, Prati A, Cucchiara R. Mixtures of von

mises distributions for people trajectory shape analysis [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2011, 21(4): 1051-8215.

- [6] Chong Wang, Zhihong Xi, Chunli Xiao. Method of moving object detection based on background subtraction [J]. Applied Technology, 2009, 36(10): 16-18.
- [7] Kai Sun, Shirong Liu. Combined Algorithm with modified camshift and Kalman Filter for multi-object tracking [J]. Information and Control, 2009, 38(1): 9-14.
- [8] 郑江滨, 李秀秀, 张艳宁. 视频监控中的运动目标跟踪算法 [J]. 系统工程与电子技术, 2007 (11): 1991-1993.

作者简介:

瞿 中 男, (1972—), 博士、教授. 研究方向为数字图像处理、普适计算、物联网技术等.

张 宁 女, (1989—), 硕士研究生. 研究方向为数字图像处理.