基于学习方式对Hadoop作业调度的改进研究

余正祥
(楚雄师范学院 楚雄 675000)

摘 要 随着并行计算、分布式计算和网格计算技术的发展,云计算作为一种新的模型被提出来,发展极为迅速。Hadoop作为一个开源的云计算系统,得到了广泛的运用。作业调度是Hadoop平台的核心问题之一,通过对Hadoop中已有调度算法的了解和分析后,基于学习的方式,利用过去的节点历史记录和作业属性来不断地改进作业调度;应用了基于特征加权的朴素贝叶斯分类器算法来改进任务的分配调度,并通过实验进行了验证,结果表明它对任务分配调度执行效率有一定的提高。

关键词 云计算,作业调度,特征加权朴素贝叶斯

中图法分类号 TP311 **文献标识码** A

Research on Improving Hadoop Job Scheduling Based on Learning Approach

YU Zheng-xiang
(Chuxiong Normal University, Chuxiong 675000, China)

Abstract With parallel computing, distributed computing and grid computing technology, cloud computing was proposed as a new model and developing fast. Hadoop is an open source cloud computing system that has been widely used. Job scheduling is one of the core problem on Hadoop platform. Through understanding and analyzing current scheduling algorithm that has already existed for Hadoop, based on learning approach, the past history of nodes and job attri-butes were used to improve job scheduling. Feature weighting-based naive bayes classification algorithm was applied to improve tasks scheduling, then it was verified through experiments. As a result, it improves the efficiency of scheduling of task allocation for Hadoop.

Keywords Cloud computing, Job scheduling, Feature weighted naive bayes

1 背景

随着并行计算、分布式计算和网格计算技术的发展,云计算[1]作为一种新的模型被提出来,在学术界和工业界产生了巨大的影响。云计算是多种技术混合演进的结果,发展极为迅速,Google、Amazon、IBM、Microsoft、Yahoo等大公司是云计算的先行者[2]。目前,大部分的云计算系统采用基于MapReduce[3]模型和分布式文件系统,模仿并实现Google和Amazon的云计算核心技术。

Google的云计算技术具体包括Google文件系统(GFS)[4]、分布式的计算编程模型MapeReduce、分布式锁服务和分布结构化数据存储系统Bigtable等。Amazon提供的ElasticMapReduce服务采用了托管的Hadoop框架,运行在Amazon EC2和Amazon S3的网络架构下。

Hadoop是Apache开源组织的一个分布式计算框架,可以在大量廉价硬件设备组成的集群上运行应用程序,在很多大型的网站都得到了应用。Hadoop最大的优点是实现了并行化和对应用开发者的透明性,开发者可以像开发普通程序那样来开发分布式的并行程序。Hadoop是一个正在不断发展和完善的平台,主要核心之一就是作业调度问题,这关系到Hadoop平台整体的性能和系统的资源使用效率。

2 Hadoop中的作业调度

Hadoop平台应用的作业类型,主要包括生产性应用:数据加载、统计值计算、垃圾数据分析等;批处理作业:机器学习等;交互式作业:SQL查询、样本采集等[5]。考虑在一个共享的Hadoop集群里,提高处理小作业的响应时间和Hadoop平台资源利用效率,以确保生产性作业的服务水平。

Hadoop平台为主/从(master/salve)式的结构。主节点称为名称节点,作业调度中称为JobTracker节点;从节点称为数据节点,作业调度中称为TaskTracker节点。这里研究的Hadoop版本是0.20.2,以下是对这3种调度器中调度算法的详细研究。

在Hadoop MapReduce计算架构中,进行作业调度使用的是FIFO算法。FIFO算法是Hadoop调度器的默认作业调度算法[6]。所有用户的作业都被提交到一个队列中,然后按照作业的优先级高低,再按照作业提交时间的先后顺序来选择将被执行的作业。其优点是调度算法简单,名称节点工作负担轻;主要的不足是忽略了不同作业的需求差异。

计算能力调度算法[7]包含多个队列,组织作业到队列中,每个队列可以共享集群资源的百分比。这样,可以限制每个队列的最大资源。每个队列里采用FIFO算法调度,每个队

余正祥(1980—),男,硕士,助理馆员,主要研究方向为云计算、数字图书馆,E-mail:zhengxiang3396y@gmail.com。

列内的作业支持优先级，每个队列强制地分配给每个用户受限制的容量。此外，计算能力调度还能够有效地对 Hadoop 集群的内存资源进行管理，以支持内存密集型应用[8]。

公平调度算法[9]是一种赋予作业资源的方法，它的目的是让所有的作业随着时间的推移，都能平均地获取等同的共享资源。当单独一个作业在运行时，它将使用整个集群。当有其它作业被提交上来时，系统会将任务空闲时间片赋给这些新的作业，以使得每一个作业都大概获取到等量的 CPU 时间。与 Hadoop 默认调度器维护一个作业队列不同，这个特性可以让小作业在合理的时间内完成又不"饿"到消耗较长时间的大作业。它也是一个在多用户间共享集群的简单方法。公平共享可以和作业优先权搭配使用，即优先权像权重一样作为决定每个作业所能获取的整体计算时间的比例。

目前有其它的一些调度，如 LATE(Longest Approximate Time to End)调度，其考虑了集群系统异构的环境[10]。Deadline 调度是根据作业的运行进度和剩余时间动态调整作业获得的资源量，以便作业尽可能地在截止时间内完成[11]。Deadline 限制调度根据作业的截止时间和当前系统中的实时作业运行情况，预测提交的作业是否在截止时间内完成，如果不能，则将作业反馈给用户，重新调整作业的截止时间[12]。在最新版本源代码贡献中由 HP 实验室开发的动态优先级调度，允许用户不断地增加或更改他们的队列优先顺序来满足他们当前负载的需要[13]。

3 基于学习方式的改进算法

3.1 算法产生的背景

希望通过利用过去的节点历史记录和学习作业属性来不断地改进作业调度，也就是通过一种机器学习的方式来实现调度算法。

对于基于学习方式的应用已经有很多，如随机学习自动机在调度器的负载均衡中应用[14]，决策树用在 Linux 的进程调度[15]。贝叶斯学习用在共享集群环境中有效地处理不确定因素[16]，作者使用一个贝叶斯决策网来处理不同因素的条件独立性，包括负载均衡问题。在做出一个决策时，首先决策变量能反馈给系统当前的状态，并基于条件概率，每个行为评估之后能够计算出期望得到的效用。效用计算是基于一个目标函数的值，当前资源的状态也能和预测的资源状态相结合，然后做出决策。这里考虑使用朴素贝叶斯分类器，假设能够做出决策的所有因素是条件相互独立的。尽管有假设条件，但是朴素贝叶斯分类器性能仍然很不错[17]。与贝叶斯网相比，其实现起来也更加简单一些。

研究现有调度算法存在的一些问题和分析各种基于学习的算法在调度中的应用，特别是在考虑不同类型的计算作业时，如 CPU 密集型，网络密集型，I/O 密集型等。在此基础上应用基于特征加权的朴素贝叶斯分类器算法来改进任务分配的调度。

3.2 算法设计思想

这里利用分类的思想把作业简单地分成两种，即好作业和坏作业。排除坏作业即导致资源过载的作业，这样能减少"拖后腿"的任务，更好地利用系统资源。此设计的程序运行在 JobTracker 上，每次 JobTracker 都从作业队列中选择好作业，然后从好作业中进行任务的分配；任务完成后，TaskTracker 上的信息又反馈到 JobTracker 上，这些反馈信息将影响到以后的作业分类和任务分配。而基于特征加权的朴素贝叶斯算法就是在任务分配的反馈信息中确定特征属性，获取训练样本，通过训练这些参数来完善和改进作业分类，达到期望的效果。

图 1 描述了基于特征加权的朴素贝叶斯分类器算法的任务分配过程。

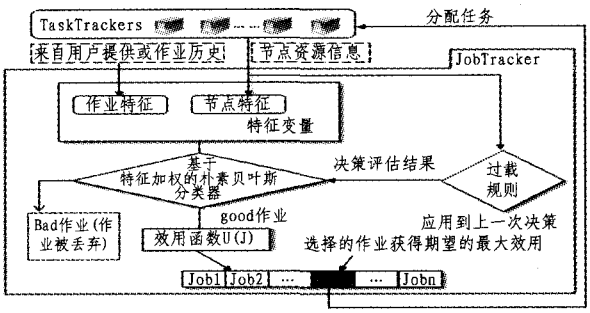

图 1 基于特征加权的朴素贝叶斯分类器算法的调度过程

首先是建立后选作业队列，每个作业包括 Map 和 Reduce 两部分，这里使用分类器把后选作业分为两类：good 类和 bad 类。在 TaskTracker 上执行的 good 类作业任务不会使资源过载。bad 类作业因会产生过载而不考虑任务分配，如果分类器标记所有作业都是坏的，则没有任务分配给 TaskTracker。分类后，如果有多个作业是 good 类，就选择下面的公式来计算，使任务的分配达到期望的效用。

$$E.U.(J)=U(J)P(\tau_J=\text{good}|F_1,F_2,\cdots,F_n)$$

式中，$E.U.(J)$为期望效用概率值。$U(J)$是作业 J 相关的效用函数，τ_J 表示作业 J 的任务，$P(\tau_J=\text{good}|F_1,F_2,\cdots,F_n)$ 表示任务 τ_J 是 good 类的可能性。可能性是有条件的，是在 $F1,F2,\cdots,Fn$ 这些特征变量下的可能性。

这里调度是本地的，因为是只考虑与某个 TaskTracker 相关的状态来做出的分配决策，这个决策不依赖于其它 TaskTracker 的资源状态。

可以跟踪任务分配决策，一旦一个任务被分配，就观察这个任务的效果，这个信息来自相同的 TaskTracker，包含在以后心跳信息中。如果基于这个信息，这个 TaskTracker 是过载的，则可以判定上次的任务分配是不正确的。这个分类器就可以训练，以便在将来避免这样的分配。如果这个 TaskTracker 不是过载的，则任务分配决策被认为是正确的。

配置过载规则是基于需求的，例如，如果大多数提交的作业是 CPU 密集型的，那么判断节点过载可能用到 CPU 利用率和平均负载。如果作业有很繁重的网络活动，网络的利用率也要考虑在过载规则中。在云计算环境中，这些资源的使用是收费的，也可以将其考虑到过载规则中去。例如，其中节约带宽是很重要的，如果该用户网络带宽的使用超出限制范围，过载规则认为这个任务分配是不正确的。过载规则唯一的要求就是能够正确识别一个节点的状态是过载还是没有过载。

从图 1 中可以看出，设计的核心是基于特征加权的朴素贝叶斯分类器，包括两个重要部分，即特征变量和效用函数。

3.2.1 特征变量

在分类中，分类需要考虑很多特征变量，这些特征变量将影响分类。这里把特征变量分为两大类，即作业特征和节点

特征。

作业特征描述一个作业对资源的使用情况，这些特征可以用来计算和分析过去作业的执行情况。在这样的系统下，用户可以利用这些特征来提交关于作业性能信息的“提示”给分类器，如果有足够多的关于作业性能的数据。提示信息就映射为资源的使用信息，这里主要的作业特征考虑有：CPU的使用状况、网络使用状况、磁盘I/O的使用状况。把使用状况分为10个等级范围，资源值为1表示最小使用范围，资源值为10表示最大使用情况。对于一个作业，它的Map部分和Reduce部分的资源使用是不同的。

节点特征表示一个节点计算资源的状态和质量。有两种类型节点特征：节点静态特征和节点动态特征。静态特征变化很少，在系统运行时保持不变。主要有：处理器的个数、总的物理内存、磁盘的数量和在TaskTracker上运行的名称和操作系统的版本等。动态特征主要包括一定时间内平凡发生变化的属性，如CPU平均负载、CPU使用率、I/O的读/写速率、在TaskTracker上正在运行处理器数目、空闲内存大小、磁盘剩余空间等。

3.2.2 效用函数

效用函数用于作业的优先次序和策略实施。效用函数一个重要的作用是保证在任务分配时不总是选择“容易”的任务。如果作业的效用是一样的，那么这个调度器在选择标记为good的作业中更容易选择那些需求资源较少的作业。因此，适当地调整作业效用，可以使每个作业获得一定的被选择机会。在实施不同的调度策略时可以通过效用函数。为了能实施混合调度策略，可以使用一个或多个效用函数。

下面是具体的效用函数例子：

(1)Map在Reduce之前：在MapReduce中，一个作业的Reduce任务开始前所有的Map任务必须结束，这里可以通过设置Reduce的任务效用为0，直到Map任务完成来保证。

(2)先来先服务(FCFS)：FCFS的实施可以保证作业的效用和作业年龄(就是提交作业后等待的时间)成正比，在提交时作业的年龄为0。

(3)预算约束：在此策略中，一个作业的任务分配，需要满足用户的帐户有足够余额。只要余额达到零，就表示该作业效用变为零，因此作业任务分配到TaskTracker节点上。

(4)付费方式：在此策略中，作业的效用是直接与用户愿意支付并成功完成作业的费用成正比，这可以确保总是选择为其服务支付更多费用的用户作业。

3.2.3 基于特征加权的朴素贝叶斯分类器

将贝叶斯定理应用到公式中，得到：

$$E.U.(J)=U(J)\frac{P(F_1,F_2,\cdots,F_n|\tau_J=\text{good})P(\tau_J=\text{good})}{P(F_1,F_2,\cdots,F_n)} \quad (1)$$

式中，分母被看作是一个常量，因为作业之间是相互独立的，在比较时计算可以忽略。

这里需要计算 $P(\tau_J=\text{good}|F_1,F_2,\cdots,F_n)$ 和 $P(\tau_J=\text{bad}|F_1,F_2,\cdots,F_n)$，作业被标记为good还是bad取决于两种可能性哪个更高。下面假设 $F_1,F_2,\cdots,F_n$ 特征条件是相互独立的，可以得到式(2)：

$$P(F_1,F_2,\cdots,F_n|\tau_J=\text{good})=\prod_{i=1}^{n}P(F_i|\tau_J=\text{good}) \quad (2)$$

将式(2)代入式(1)中，并且忽略 $P(F_1,F_2,\cdots,F_n)$，得到式(3)：

$$E.U.^*(J)=U(J)P(\tau_J=\text{good})\prod_{i=1}^{n}P(F_i|\tau_J=\text{good}) \quad (3)$$

在这里考虑不同类型的计算作业，如：CPU密集型、网络密集型、I/O密集型或者是几种混合。在 $F_1,F_2,\cdots,F_n$ 这些特征变量中，不同类型的作业其特征变量权重是不一样的，于是对 $F_1,F_2,\cdots,F_n$ 进行加权，得到式(4)：

$$E.U.^*(J)=U(J)P(\tau_J=\text{good})\prod_{i=1}^{n}P(F_i|\tau_J=\text{good})^{w_j} \quad (4)$$

对于 W_j 的确定，设 $SGF(f_j,D)$ 表示特征 F_j 相对于决策属性 D 的重要性[18]，则属性 F_j 的权值为：$w_j=\frac{SGF(f_j,D)}{\frac{1}{n}\sum_{i=1}^{n}SGF(f_j,D)}$。

4 实验验证

实验采用5台普通的PC机进行测试，硬件配置：CPU 2.4G双核，内存2G，硬盘160G。这5台PC机通过一个普通的千兆交换机相连。软件环境为Ubuntu-desktop 10.10操作系统，安装java jdk1.6.20，hadoop-0.20.2软件，开发工具：Eclipse 3.5.0。

在原型设计中，简化 $SGF(f_j,D)$ 的计算，$SGF(f_j,D)$ 的值根据作业类型通过配置文件手工指定。原型设计实现了FCFS的效用函数。使用最常用的WordCount(从文本数据统计单词次数)作业来测试调度的改进。

为了计算各个特征变量的权值，需要计算 $SGF(f_j,D)$ 的值，$SGF(f_j,D)$ 表示特征 F_j 相对于决策属性 D 的重要性。在实验中，$SGF(f_j,D)$ 值的范围为 $1\leqslant SGF(f_j,D)\leqslant 5$。对WordCount作业的 $SGF(f_j,D)$ 值的设置如表1所列。

表1 $SGF(f_j,D)$ 值预分配分配表

作业	CPU	内存	磁盘	网络
WordCount	3.8	4	4.5	3.8

先测试调度器的学习行为。在这个实验中首先运行WordCount作业，让这个作业运行多次并处理随机产生1GB大小的文本。在每次运行前输入的文本是重新产生的。图2显示了节点的效用随时间不断变化。

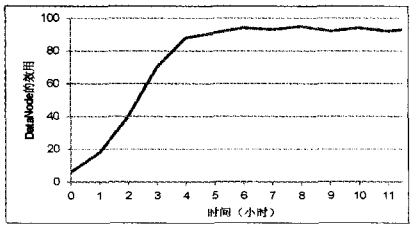

图2 DataNode节点效用与时间关系图

该调度器希望达到100%的效用。如图2所示，刚开始时其效用值低于期望的效用，这个阶段是学习阶段；这个阶段之后，节点就很少发生过载情况，效用达到90%以上，接近理想的值。

如图3所示，WordCount每次运行的时间有减少的趋势，这里只描述了7次的运行时间，6次之后运行时间保持在33分钟左右。运行减少最快的就是开始的2次作业。在头2次

(下转第256页)

1）改进后的算法使用相似度删除了部分冗余概念，对一般概念提取的准确率较高。

2）根据候选概念上下文的语义情境给出了低频同义词和整体-部分关系词的判别准则，在实际应用中执行力更强。

结束语 本文同时考虑了因为同义关系和整体-部分关系低频词带来的领域概念筛选问题，通过更细致的考虑改进了原来的计算公式，并与现有方法进行了实验比较，表明改进后的方法有明显效果。

参 考 文 献

[1] 朱礼军，陶兰，黄赤. 语义万维网的概念、方法及应用[J]. 计算机工程与应用，2004，3：79-84

[2] Maedche A. Staab S. Ontology learning for the semantic Web [J]. IEEE Intelligent systems，2001，16(2)：72-79

[3] Shamsfard M，Barforoush A. Learning ontologies from natural language texts[J]. International Journal Human-computer Studies，2004，60(1)：17-63

[4] Missikoff M，Navigli R，Velardi P. Integrated approach for Web ontology learning and enginee-ring[J]. IEEE Computer，2002，35(11)：60-63

[5] Moldovan D，Girju R，Rus V. Domain-specific knowledge acquisition from text [C]//Proc. of The Sixth Conference on Applied Natural Language Processing. 2000：268-275

[6] Velardi P，Fabriani P，Missikoff M. Using text processing techniques to automatically enrich a Domain ontology [C]//Proc. of the Formal Ontology in Information Systems. 2001：270-284

[7] 贾秀玲，文敦伟. 面向文本的本体学习中概念提取及关系提取的研究[J]. 中南大学，2007

[8] 张玉芳，杨芬，熊忠阳. 基于上下文的领域本体概念和关系的提取[J]. 计算机应用研究，2010

[9] 王红滨，刘大昕，王念滨. 一种本体学习中的领域概念筛选算法[J]. 系统工程与电子技术，2010

[10] Liu Bai-song，Gao Ji. General ontology learning frame-work[J]. Journal of Southeast University(English Edition)，2006，22(3)：381-384

（上接第 222 页）

运行中时间较长，由于调度器处于学习的阶段，因此运行的效率比较低。之后作业的运行时间就处于一个稳定的状态，运行时间收敛。

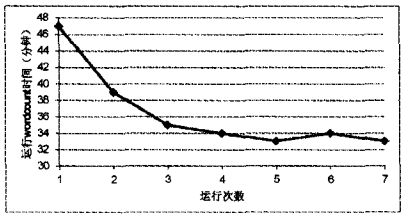

图 3 运行 WordCount 作业与运行次数关系图

为了观察特征加权对 WordCount 作业运行时间的影响，将 CPU、内存、磁盘、网络的 $SGF(f_i, D)$ 值设为相同，如都设为 4。此时根据公式计算，CPU、内存、磁盘、网络的权值都为 1，简化为不带权值的朴素贝叶斯分类调度。同样重复运行 7 次，WordCount 作业的两种比较如图 4 所示。

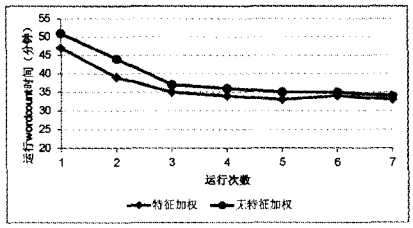

图 4 特征加权与无特征加权作业比较

从图 4 中可以看出，在对资源估计值都相同的条件下，通过调整特征变量的权值可以缩短作业运行的时间，有效地提高资源的使用效率。特别是在作业运行时，头 2 次作业运行时间缩短比较明显；通过学习之后，特征加权和无特征加权趋近相同，但总体上特征加权之后效率得到提高。

参 考 文 献

[1] Cloud computing [EB/OL]. http://en. wikipedia. org/wiki/Cloud_computing，2010-05-10

[2] 刘鹏. 云计算[M]. 北京：电子工业出版社，2010

[3] MapReduce[EB/OL]. http://zh. wikipedia. org/zh-cn/MapReduce，2010-03-05

[4] Ghemawat S，Gobioff H，Leung S T. The Google file system[J]. ACM SIGOPS Operating Systems Review，2003，37(5)：29-43

[5] 中科院计算所网络重点实验室. Hadoop 作业调度研究[EB/OL]. http://www. slideshare. net/YongqiangHe/hadoopv01，2010

[6] Zaharia M. Job Scheduling with the Fair and Capacity Schedulers [EB/OL]. http://www. cs. berkeley. edu/～matei/talks/2009/hadoop_summit_fair_scheduler. pdf，2009-07-10

[7] Capacity Scheduler Guide[EB/OL]. http://hadoop. apache. org/common/docs/r0. 20. 2/capacity_scheduler. html，2010

[8] 王峰. Hadoop 集群作业的调度算法[J]. 程序员，2009，12

[9] Fair Scheduler Guide[EB/OL]. http://hadoop. apache. org/common/docs/r0. 20. 2/fair_scheduler. html，2010-01-02

[10] Zaharia M，Konwinski A，Joseph A D，et al. Improving mapreduce performance in heterogeneous environments[C]//USENIX Association. 2008：29-42

[11] Polo J，Carrera D，Becerra Y，et al. Performance-driven task co-scheduling for mapreduce environments[C]//12th IEEE/IFIP Network Opertions and Management Symposium. 2010：373-380

[12] Kc K，Anyanwu K. Scheduling Hadoop Jobs to Meet Deadlines [C]//CLOUDCOM'10 Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science. 2010：388-392

[13] Sandholm T，Lai K. Dynamic proportional share scheduling in Hadoop[C]//Springer. 2010：110-131

[14] Kunz T. The Learning Behaviour of a Scheduler using a Stochastic Learning Automation[J]. Relation，1991，10(1-127)：2600

[15] Negi A，Kishore K. Applying machine learning techniques to improve linux process scheduling[C]//IEEE. 2005：1-6

[16] Santos L P，Proenca A. Scheduling under conditions of uncertainty：a bayesian approach[C]//Springer. 2004：222-229

[17] Zhang H. The optimality of naive Bayes[J]. A A，2004，1(2)：3

[18] 邓维斌，王国胤，王燕. 基于 Rough Set 的加权朴素贝叶斯分类算法[J]. 计算机科学，2007，34(002)：204-206