Hadoop平台下的动态调度算法

高燕飞 陈俊杰 强 彦
(太原理工大学计算机科学与技术学院 太原 030024)

摘 要 目前,云计算环境具有动态、异构和海量多类型任务并发等特征,随着集群规模不断增大、用户QoS不断增多,现有调度算法越来越难以适应动态变化的环境及满足用户的需求。针对Hadoop平台下现有调度器不能根据作业运行状态和资源使用情况进行动态调整的问题,提出了Hadoop下基于作业分类的动态调度算法。该算法在使用朴素贝叶斯分类算法对队列中作业进行分类的过程中,根据各个作业的类型,预先设定类别权值,将队列中的作业分类,并引入效用函数,根据用户提交时的预期完成时间QoS和作业完成情况估算其作业完成时间,实现动态设置作业优先级。实验表明,使用提出的算法不仅能有效减少作业的分类时间,而且能明显提高动态性和用户QoS。

关键词 人机交互,Hadoop,动态调度,贝叶斯网络,QoS
中图法分类号 TP39 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.9.010

Dynamic Scheduling Algorithm in Hadoop Platform

GAO Yan-fei CHEN Jun-jie QIANG Yan
(College of Computer Science and Technology, Taiyuan University of Technology, Taiyuan 030024, China)

Abstract With the increasing of the clusters and the user's QoS in the cloud environment, it becomes much harder to meet the requirements of jobs and users using the traditional strategy. To adjust scheduler dynamically according to the status of the jobs and the resources, this paper proposed a dynamic scheduling method based on the job classification method in the Hadoop platform. The proposed method employs the Naïve Bayesian method to classify the jobs in which the human inferences are added to preset the jobs' weight according to the types. Then, the scheduling priority of the jobs is set dynamically using the utility function based on the user's expected completing time and the estimated completed time of jobs. The experimental results show that the proposed method can not only reduce the classification time, but also improve the scheduling dynamics and user's QoS greatly.

Keywords Human-computer interaction, Hadoop, Dynamic scheduling, Bayesian network, QoS

1 引言

为了整合优化资源、满足不同用户QoS并降低成本[1],云计算采用了资源池的方式对计算、存储、网络、软件等资源进行统一管理和调度[2]。云计算中的任务调度的好坏影响着整个集群的性能,不良的任务分配可能会导致网络流量增加、使TaskTracker负载过重、降低系统吞吐量、用户QoS差等问题。动态、异构和海量多类型任务并发等特点使得云计算环境下的资源调度非常具有挑战[3]。目前的调度算法中,均需要进行一些配置。如果预先设置的任务数过大,则会导致每个任务分得的资源过小,可能会因为任务间的不断切换而导致集群过载;反之,则会导致资源的浪费,降低集群的整体性能。要想准确地预先设置最合适的任务数,则需要详细的作业的资源使用情况和TaskTracker的资源情况,但由于集群中作业使用资源情况的不确定性,很难准确预设参数[4]。因此,提高云平台下调度器的动态性,对提高云平台下调度的效率和系统资源的利用率具有重要的意义。本文将借助机器学习的优势,在考虑用户完成时间QoS的基础上,提出适应于动态云计算环境Hadoop下的动态调度框架。该框架能够对提交作业进行分类,并可以根据用户提交预期完成时间QoS来动态实现为作业添加优先级[5],以提高云计算平台的多类型任务调度效率和调度准确率,实现动态调度。

2 国内外研究现状

在Hadoop的动态调度研究方面,Aysan Rasooli等人[6]提出了异构环境中的动态调度算法,通过估计作业到达率和平均作业执行时间来决定调度策略;并且在模拟软件MRSIM环境中对提出的算法进行了测试。Quan Chen等人关注现有MapReduce的调度器在异构环境中表现不好的问题,提出了动态MapReduce调度算法SAMR[7,8],它能够动态计算任务进展,自动适应不断变化的环境。但该调度器在运行备份时,未考虑数据本地化,同时也缺少一种动态参数调优机制。Hong Mao等人[9]关注如何在任务调度层对调度器进行优化,提出了一个负载驱动的动态任务调度器,它能够动态地

到稿日期:2014-06-11 返修日期:2014-07-29 本文受国家自然科学基金(61202163,61240035),山西省自然科学基金(2012011015-1),山西省科技攻关项目(20120313032-3)资助。
高燕飞(1983—),女,博士生,讲师,主要研究领域为数据挖掘、云计算、高性能计算;陈俊杰(1956—),男,博士,教授,主要研究领域为数据挖掘、高性能计算;强 彦(1969—),男,博士,副教授,CCF会员,主要研究领域为医学图像处理,E-mail:qiangyan@tyut.edu.cn。

调整运行在各个 TaskTracker 上的 map 任务和 reduce 任务数量，在处理 10GB 的数据量时，作业运行时间有所缩短，CPU 利用率得到了提高。文献[10,11]讨论了基于不同作业的资源需求，把任务分配到不同的队列；设计了一个 MapReduce 负载预测机制（Workload Predict Mechanism）来检测 MapReduce 过程的负载类型，并对其分类，放到相应的队列中。针对需要频繁 I/O 的作业类型，Hui Kang 等[12]提出了 MapReduce 组调度算法，以降低上下文切换次数和解决 I/O 阻塞问题。Bikash Sharma 等[13]针对高吞吐量计算作业，提出一个算法框架，以调度空闲云节点的计算资源，从而提高了资源的利用率。

这些基于不同作业的调度算法各有侧重，针对不同的作业有不同的优化解决办法，但是 Hadoop 上的算法是需要预先设置的，设置好某类算法就只能优化解决某一类问题[14]。因此，在云计算平台中，如果采用动态技术，针对作业的类型和 QoS 特征，自动调用和配置不同的、与任务相适应的调度机制，有可能提高用户满意度和云平台的运行效率，降低硬件投入总成本，精细化平台功耗管理功能。

3 云平台下作业调度概述

Hadoop 下的每个作业被分为多个任务，作业调度的核心思想是 JobTracker 把作业划分后将合适任务分配到合适的 TaskTracker 上去执行。该过程包含 3 个步骤：(1)选择一个作业；(2)在该作业内选择一个任务；(3)把选定的任务分派到 TaskTracker 上执行。作业提交后，MapReduce 作业的整个生命周期分为 6 步：

1)提交作业：提交作业后，JobClient 实例会将作业的配置文件、分片元信息文件等上传到 HDFS 上。之后通知 JobTracker 作业已提交。

2)作业初始化：JobTracker 收到信息后，交由作业调度模块对作业初始化。对于每个作业，会有一个 JobInProgress 对象来记录其运行情况。同理，对于作业中的任务，由 TaskInProgress 对象来记录每个任务的运行情况。

3)任务调度与监控：任务调度和监控均由 JobTracker 完成。TaskTracker 负责向 JobTracker 实时反馈节点资源使用情况。当出现空闲资源时，JobTracker 将按照任务调度器的调度策略向合适的任务分配资源；当出现 TaskTracker 或任务失败，JobTracker 负责转移计算任务；当某任务进度远远落后同一作业的其他任务时，为之启动一个相同任务，并选取计算快的任务结果作为最终结果。

4)任务运行环境准备：包括启动 JVM 和资源隔离，由 TaskTracker 实现。TaskTracker 为每个 Task 启动一个独立的 JVM 的目的是避免不同任务在运行过程中相互影响。资源隔离的目的在于防止任务滥用资源的情况。

5)执行任务：运行环境准备好后，会启动任务。每个任务的执行进度会首先上报给 TaskTracker，再汇报给 JobTracker。

6)作业完成：整个作业执行成功的标志是所有任务执行完成。

4 动态调度算法

4.1 整体架构

根据以上描述，本文提出适应于动态云计算环境 Hadoop 的动态调度框架。设计思路为借助机器学习的优势对提交的作业进行分类，并可以根据用户提交的预期完成时间 QoS 来动态实现为作业添加优先级。预期完成时间不是任意的，由用户自己估计，在提交命令时一起提交，是命令的一部分。该动态调度算法的整体架构如图 1 所示。

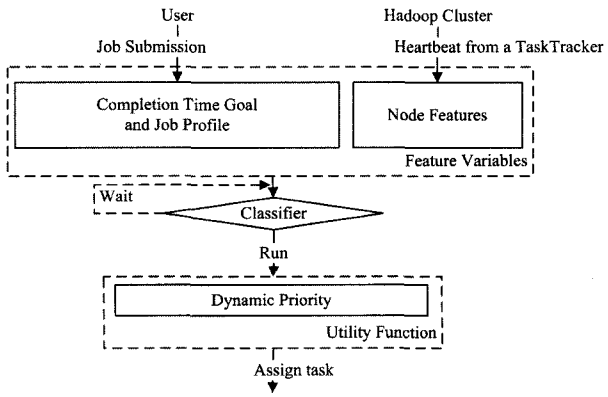


图 1 动态调度器的整体架构

其主要模块包括：

1)作业提交模块

该模块包含两部分：第一部分是作业特征，包括用户提交的作业和提交作业时附带的期望完成时间；第二个部分是节点特征，由 TaskTracker 向 JobTracker 汇报，包括每个节点上的 CPU 的使用率、I/O 读写速率、网络传输/接收速率、剩余内存大小以及磁盘剩余空间等。节点特征还会被用于过载判断模块，以判断上次任务分配是否会造成 TaskTracker 过载。

2)作业分类模块

作业分类模块将作业分为两类：运行作业和等待作业。运行作业指不会导致 TaskTracker 过载而被调度到的作业，用 run 表示。等待作业指可能导致 TaskTracker 过载而不进行调度的作业，用 wait 表示。选择出标签为“run”的作业。

3)动态设置优先级模块

当作业分类模块给多个作业标记为“run”时，引入效用函数来给这样的作业添加优先级，具体方法是利用作业提交模块中的作业提交时的作业特征（期望完成时间）和节点特征来计算每个作业的“期望效用”（期望效用中的效用函数通过对作业完成时间的估计进行计算）；然后选择期望效用值最大的那个作业，将该作业的任务分配到请求任务的 TaskTracker 上执行。

4.2 作业分类模块

Jaideep Dhok[15]曾将朴素贝叶斯分类器用于 Hadoop 作业调度中的任务分配，并通过实验证明了该分类器能够很好地解决分布式资源管理的问题。因此，本文将朴素贝叶斯分类器用于 Hadoop 作业调度中的作业分类环节。

朴素贝叶斯分类器的输入数据分为两类：作业特征和节点特征（用 V_i 表示），具体如表 1 所列。

分类器根据对以上特征的计算，对队列里的作业进行分类，该分类器将作业分为两类：运行作业和等待作业。运行作业指不会导致 TaskTracker 过载而被调度到的作业，用 run 表示。等待作业指可能导致 TaskTracker 过载而不进行调度的作业，用 wait 表示。

表1 朴素贝叶斯分类器的输入数据分类

输入数据	描述	包含	
作业特征	作业运行过程中的描述	作业执行情况中的平均CPU利用率、平均内存利用率、平均I/O率和平均网络使用情况等	
节点特征	描述了每个TaskTracker上的资源使用情况	静态资源	处理器数量、处理速度、总的物理内存大小、虚拟内存大小、磁盘个数等
		动态资源	CPU的使用率、I/O读写速率、网络传输/接收速率、空闲的物理内存大小、空闲的虚拟内存大小以及剩余的磁盘大小等

分类的具体计算过程为：

1)分别计算运行一个作业在某些条件下是“运行作业”还是“等待作业”的条件概率：

$$P(J_j=run|V_1,V_2,\cdots,V_n)$$
$$P(J_j=wait|V_1,V_2,\cdots,V_n) \quad (1)$$

其中，J_j 表示作业 j，V_i 为作业特性和节点特性。

2)根据贝叶斯公式 $P(B|A)=P(AB)/P(A)$ 得：

$$P(J_j=run|V_1,V_2,\cdots,V_n)=\frac{P(V_1,V_2,\cdots,V_n|J_j=run)P(J_j=run)}{P(V_1,V_2,\cdots,V_n)} \quad (2)$$

假设 V 之间相对独立，根据独立假设，其中，

$$P(V_1,V_2,\cdots,V_n|J_j=run)=\prod_1^n P(V_i|J_j=run)$$

3)实际计算中，$P(V_1,V_2,\cdots,V_n)$ 与作业无关，可忽略不计。因此，最终可得：

$$P(J_i=run|V_1,V_2,\cdots,V_n)=P(J_j=run)\prod_1^n P(V_i|J_j=run) \quad (3)$$

同理有：

$$P(J_j=wait|V_1,V_2,\cdots,V_n)=P(J_j=wait)\prod_1^n P(V_i|J_j=wait) \quad (4)$$

作业是“运行作业”还是“等待作业”取决于哪个概率值更大。

步骤3中，$P(J_i=run)$，$P(V_i|J_j=run)$，$P(J_i=wait)$，$P(V_i|J_j=wait)$ 为先验概率，其值由每次任务分配后 TaskTracker 的反馈结果而得，反馈结果可以通过这样的方式来影响 $P(J_j=run|V_1,V_2,\cdots,V_n)$ 和 $P(J_j=wait|V_1,V_2,\cdots,V_n)$ 的值，从而影响下次分类的结果。

4.3 作业完成时间估计 QoS

针对朴素贝叶斯分类器可能出现的分类结果标记多个作业为“run”状态，需要进一步确定到底哪一个作业应该被调度。默认情况下，该调度器会从标记为“run”的作业中选择能实现系统可用性最高的作业（$P(J_i=run|V_1,V_2,\cdots,V_n)$ 值最大的作业）来调度，这类作业使用资源最少，最不可能使 TaskTracker 过载，但是会产生饥饿现象。为了让每个作业都能公平地被调度，本节引入效用函数，通过考虑用户期望完成时间 QoS，估算作业完成时间，从而实现对作业优先级的设置。具体方法如下：

(1)对于每个标记为“run”的作业，首先计算其“期望效用”（用 $E\cdot U\cdot(j)$ 表示）：

$$E\cdot U\cdot(j)=U(j)P(J_j=run|V_1,V_2,\cdots,V_n) \quad (5)$$

其中，$U(j)$ 为作业 j 的效用函数，它由作业完成时间 QoS 估算而得。

$$U(j)=U^M(j)+U^R(j) \quad (6)$$

式中，$U^M(j)$ 是一个效用函数，表示给定的 map 任务的一个作业的满意度，$U^R(j)$ 是一个效用函数，显示给定的 reduce 任务的一个作业的满意度。这两个函数的定义如下：

$$U(j)=\begin{cases}\dfrac{s^j_{goal}-s^j_{req}}{s^j_{pend}}, & s^j_{goal}\geqslant s^j_{req}\\ \dfrac{\log(s^j_{goal})}{\log(s^j_{req})}-1, & s^j_{pend}<s^j_{req}\end{cases} \quad (7)$$

$$U(j)=\log(s^j_{goal})/\log(s^j_{req})-1 \quad (8)$$

(2)选择期望效用值最大的作业，即 $E\cdot U\cdot(j)$ 值最大的作业，然后从该作业中选择任务进行分配。

(3)在分配任务时，优先选择该任务所需数据就在该 TaskTracker 上的任务进行调度，否则再选择其所需数据不在本地的任务进行调度。

计算效用函数 $U(j)$ 时，为了根据观测到的作业进展率和作业提交时提供的作业完成时间目标，来实现 MapReduce 运行时动态地分配资源的目标，一个能够对作业完成时间估计的模块就必不可少。

作业完成时间估计模块能够实现在作业运行的过程中动态估计其完成时间，利用作业提交时附带的期望完成时间（T^j_{goal}）、作业中待处理的任务数（s^j_{pend}）和任务的平均长度（执行时间）来动态估算还需要为作业提供的资源数（s^j_{req}），进而指导任务分配，从而有助于其他模块动态地对该作业分配资源，其参数定义如表2所列。

表2 作业完成时间估计模块参数定义

定义	含义
j	作业j
t_i	每个作业j由一定数量的任务（t_i，i=1,2,…,n）组成
T^j_{goal}	作业j提交时会附带期望完成时间
TT_i	表示每个TaskTracker，可提供一定量的资源槽 $Slot_i$
U_j	未开始任务
R_j	正在运行任务
C_j	已完成任务
a^j_i	执行每个任务（用 t^j_i 表示）需要占用一个槽，并用时间 a^j_i 完成
β^j_i	任务已用时间
δ^j_i	任务剩余时间
ave_j	正在运行中的作业j中已完成任务的平均运行时间，$ave_j=\dfrac{\sum_{i\in c_j}a^j_i}{\lvert c_j\rvert}$
T_{curr}	当前时间
s^j_{pend}	完成作业j仍然需要的槽数

其中，$a^j_i=\beta^j_i+\delta^j_i$。由于 a^j_i 和 δ^j_i 均未知，假设对于每个正在运行的任务 t^j_i，执行一个任务所需时间等于已完成任务的平均长度，即假设 $a^j_i=ave_j$，因此任务的剩余时间 $\delta^j_i=ave_j-\beta^j_i$。

知道了作业 j 中一个任务的剩余时间，就可以估算还需要为该作业的任务分配多少资源，也就是任务调度的过程。该过程首先会为每个作业指定优先级，然后根据作业的优先级为其分配空闲槽。指定优先级时，根据分配给每个作业的槽数进行动态计算，因此需要估计每个作业待处理的工作量。

待处理的任务量包括：R_j 中的剩余量和 U_j。由此可得 s^j_{req}：

$$s^j_{req}=[(\frac{\sum_{i\in R_j}\delta^j_i}{ave_j}+|U_j|)\cdot ave_j]/(T^j_{goal}-T_{curr})-|R_j| \quad (9)$$

任务列表将根据每个作业计算而得的 s^j_{req} 值动态排序，即动态指定调度优先级的过程。

调度策略环节还需要考虑一些特殊情况，比如，刚刚提交

作业后若没有可用的数据，则它所需要的槽数和完成时间不可估计。因此，调度的机会会被还没完成的作业和正在运行的任务长期占用。如果这样的任务不止一个，则先来的作业优先调度。还有另外一种情况需要注意：对于那些已经错过最后期限的作业，调度器将优先调度它们，以便让它们尽快完成，从此避免饥饿。综上所述，作业将按这样的优先级计算：首先是那些错过自己最后期限的作业，然后是最近提交但没有可用数据的作业，最后执行按照 s_{req}^{j} 排序的作业。

5 实验验证

本节将通过实验来验证本文提出的动态调度算法。实验环境为 6 台 PC 机搭建而成的 Hadoop 完全分布式集群，每个节点拥有 1.8GHz 的四核 AMD 处理器，4GB 内存，节点间采用 100M 带宽相连。其中一台为主节点 namenode1，其余 5 个节点为从节点，分别为 datanode1－datanode5。每个节点上的操作系统为 64 位的 CentOS 6.4。Hadoop 版本为 1.1.2，数据块大小为默认的 64MB。采用分布式监控系统 Ganglia 来监控作业运行情况。

5.1 测试用例选择

在实验环节，我们使用 Hadoop 源码中的 4 个 MapReduce 测试程序，分别为 Teragen、Wordcount、Terasort、Pi Estimator。为模拟实际情况，3 个作业的提交顺序没有特定要求，是随机的，更改作业的提交顺序不影响实验结果。整个集群配置的最大同时运行 map 任务数设置为 12，最大同时运行 reduce 任务数为 12。测试用例及其描述如表 3 所列。

表 3 测试用例及其描述

负载	介绍	资源特性
Teragen	产生数据(2.5GB)用于后续程序	I/O 密集型
Wordcount	单词计数	CPU 密集型
Terasort	排序	在 map 阶段代表 CPU 密集型，reduce 阶段代表 I/O 密集型
Pi	使用蒙特卡罗方法估算 π 值	CPU 密集型

5.2 实验结果分析

实验环节把 4 个不同类型的测试用例组成一个混合的序列，在资源共享的环境下运行；同时提交作业 Wordcount、Terasort 和 Pi Estimator，分别在公平调度器、计算能力调度器和本文提出的动态调度器上运行。3 种调度器上的运行时间统计情况如表 4－表 6 所列。

由表 4－表 6 可知，动态调度器下，wordcount 和 terasort 的运行时间明显缩短，pi 的运行时间较长，这是因为公平调度器和计算能力调度器考虑作业间公平性的缘故，先给每个作业池分配了最小的共享资源数，然后把剩余资源均分给各个作业。因此组合中的每个作业在公平调度器下运行时获得的资源较为平均，但由于没有考虑实际的负载状态和当前系统各节点的负载水平，导致整个系统的响应时间受到了实际负载不均衡的影响。这样有利于小作业的执行，但是却拖慢了整体运行时间，降低了作业序列中其他作业的满意度。动态调度器下整个作业序列的运行时间最短，为 548s，比公平调度器快 20%，比计算能力调度器快 19%。

表 4 公平调度器下作业序列运行时间统计

顺序	运行时间(s)	Map 用时(s)	Reduce 用时(s)	CPU 花费时间(ms)		
				Map	Reduce	总
Wordcount	659	370	638	290780	101490	392270
Terasort	691	344	664	139670	72910	212580
Pi Estimator	74	6	29	450	1410	1860
总时间	691s					

表 5 计算能力调度器下作业序列运行时间统计

顺序	运行时间(s)	Map 用时(s)	Reduce 用时(s)	CPU 花费时间(ms)		
				Map	Reduce	总
Wordcount	641	121	620	277950	99640	377590
Terasort	671	278	559	149730	74940	224670
Pi Estimator	25	3	16	630	1320	1950
总时间	679s					

表 6 动态调度器下作业序列运行时间统计

顺序	运行时间(s)	Map 用时(s)	Reduce 用时(s)	CPU 花费时间(ms)		
				Map	Reduce	总
Wordcount	265	176	245	268470	97810	366280
Terasort	544	186	274	130480	73890	204370
Pi Estimator	247	98	101	510	1890	2400
总时间	548s					

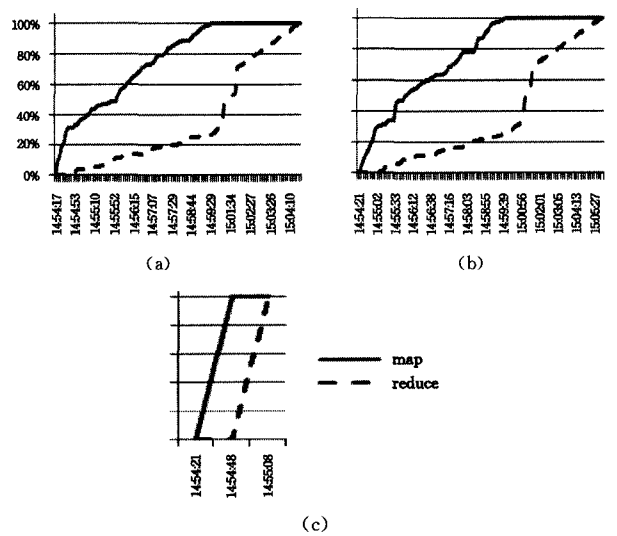


图 2 公平调度下作业序列的 map 和 reduce 运行百分比

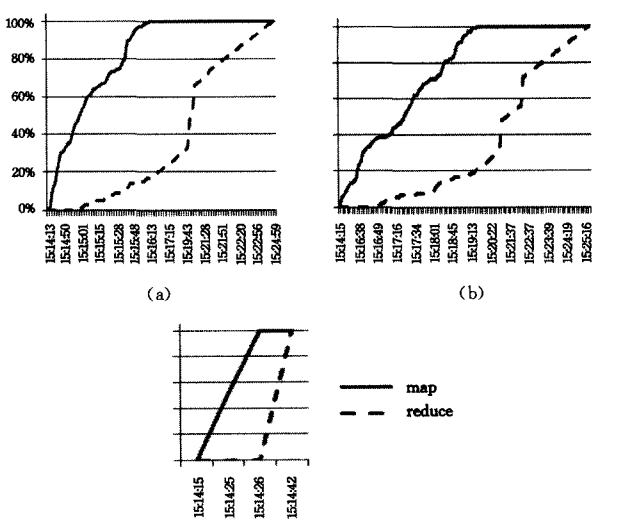


图 3 计算能力调度器下作业序列的 map 和 reduce 运行百分比

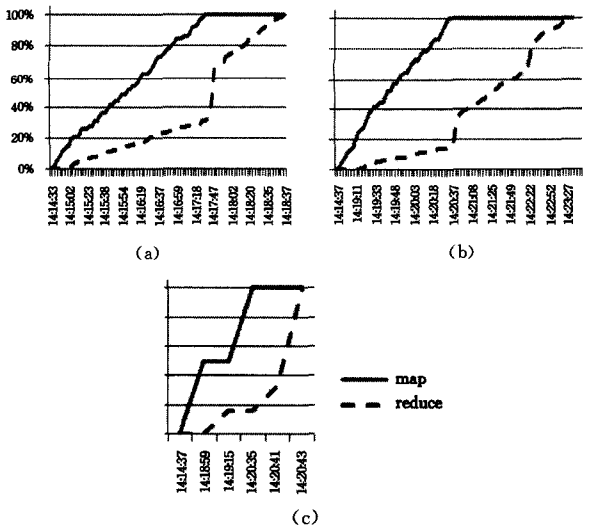
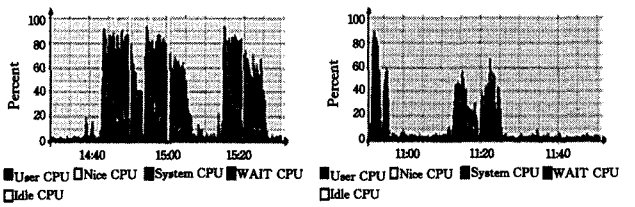


图 4　动态调度器下作业序列的 map 和 reduce 运行百分比

由图 2 到图 4 可知，公平调度器下，3 个作业的 reduce 任务在 map 运行 31%、30%、100%后才开始运行；计算能力调度器下为 47%、38%、100%；而在动态调度器下，map 任务分别运行到 15%、26%、50%时就开始调度 reduce 任务。图 5 所示为 3 种调度器下集群 CPU 利用率，图 5(a)描述了公平调度器和计算能力调度器下集群 CPU 利用率，图 5(b)中间曲线描述了动态调度器下集群 CPU 利用率。从图中可得，等待的 CPU 明显减少，等待作业的数量有所降低，作业并发度有所提高，集群资源得到了充分利用，因此，作业序列的整体运行时间会缩短。



(a)公平调度器和计算能力调度器下集群 CPU 利用率　(b)动态调度器下集群 CPU 利用率

图 5　3 种调度器下集群 CPU 利用率

对于公平调度算法和计算能力调度算法而言，在初始化作业时需要设置初始化参数；作业运行时，需要适当地调整参数，如果参数配置不合适则会出现运行效率低的情况。而本文提出的动态调度算法参数设置相对简单，可以实现较高的运行效率，运行时间比较稳定。

结束语　在动态、异构和海量任务的云计算环境下，由于随着集群规模的增大和用户 QoS 的增加，现有调度算法越来越难以适应动态变化的环境及满足用户的需求。本文以 Hadoop 为例，针对其平台下现有调度算法的不足，提出一种 Hadoop 下的作业动态调度技术，来提高云平台调度效率和资源利用率。该算法使用朴素贝叶斯网络算法对提交作业进行分类，在考虑用户提交的作业预期完成 QoS 的基础上，采用估算作业完成时间的方式来动态添加优先级；并通过实验证明该算法能提高云计算平台下多类型任务调度效率和调度准确率，实现调度的动态调度性。

参 考 文 献

[1]　朱晓敏，祝江汉，马满好. 异构集群系统中具有 QoS 需求的实时任务容错调度[J]. 软件学报，2011，22(7)：1440-1456
Zhu Xiao-min, Zhu Jiang-han, Ma Man-chao. Fault-Tolerant Scheduling for Real-time Tasks with QoS Requirements on Heterogeneous Clusters[J]. Journal of Software, 2011, 22(7): 1440-1456

[2]　Yang Y, Xiang P, Mantor M, et al. CPU-Assisted GPGPU on Fused CPU-GPU Architecture[C]// the 8th International Symposium on High Performance Computer Architecture (HPCA-18). 2012: 103-114

[3]　Dutta D, Joshi R C. A genetic-algorithm approach to cost-based multi-QoS job scheduling in cloud computing environment[C]// ICWET. 2011: 422-427

[4]　王凯，侯紫峰. 动态调整虚拟机权重参数的调度方法[J]. 计算机研究与发展，2012，48(11)：2094-2102
Wang Kai, Hou Zi-feng. An Adaptive Scheduling Method of Weight Parameter Adjustment on Virtual Machines [J]. Journal of Computer Research and Development, 2012, 48(11): 2094-2102

[5]　王守信，张莉，李鹤松. 一种基于云模型的主观信任评价方法[J]. 软件学报，2010，21(6)：1341-1352
Wang Shou-xin, Zhang Li, Li He-song. Evaluation Approach of Subjective Trust Based on Cloud Model[J]. Journal of Software, 2010, 21(6): 1341-1352

[6]　Rasooli A, Down D. An Adaptive Scheduling Algorithm for Dynamic Heterogeneous Hadoop Systems [C]// Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research. IBM Corp., 2011: 30-44

[7]　Chen Q, Zhang D, Guo M, et al. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment[C]// 2010 IEEE 10th International Conference on Computer and Information Technology(CIT). IEEE, 2010: 2736-2743

[8]　陈全，邓倩妮. 异构环境下动态的 Map-Reduce 调度[J]. 计算机工程与科学，2009，31(S1)：168-171，175
Chen Quan, Deng Qian-ni. Self-Adaptive Map-reduce Scheduling Under Heterogeneous Environment[J]. Computer Engineering & Science, 2009, 31(S1): 168-171, 175

[9]　Mao Hong, Hu Sheng-qiu, Zhang Zhen-zhong, et al. A Load-Driven Task Scheduler with Adaptive DSC for MapReduce[C]// GREENCOM. 2011: 28-33

[10]　Kaushik R T, Bhandarkar M. Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster[C]// Proceedings of the USENIX Annual Technical Conference. 2010

[11]　Lu P, Lee Y C, Wang C, et al. Workload Characteristic Oriented Scheduler for MapReduce[C]// 2012 IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2012: 156-163

[12]　李强，郝沁汾，肖利民，等. 云计算中虚拟机放置的动态管理与多目标优化[J]. 计算机学报，2011，34(12)：2253-2264
Li Qiang, Hao Qin-fen, Xiao Li-min, et al. An Integrated Approach to Automatic Management of Virtualized Resources in Cloud Environments [J]. Chinese Journal of Computers, 2011, 34(12): 2253-2264

[13]　Sharma B, Chudnovsky V, Hellerstein J L, et al. Modeling and synthesizing task placement constraints in Google compute clusters[C]// SOCC. 2011

(下转第 69 页)

fansID	总嘀咕数	领域嘀咕数	领域深度	粉丝好友数	粉丝粉丝数
1	42	2	0.047619048	42	15
2	9	1	0.111111111	16	5
3	17	1	0.058823529	14	2
4	16	1	0.0625	22	6
5	7	1	0.142857143	15	3
6	6	1	0.166666667	119	20
7	20	1	0.05	20	4
8	17	1	0.058823529	29	5
9	22	2	0.090909091	64	23
10	27	2	0.074074074	23	10
11	7	1	0.142857143	16	2
12	31	4	0.129032258	251	53
13	13	3	0.230769231	25	3
14	16	1	0.0625	32	6
15	14	1	0.071428571	28	3
16	43	4	0.093023256	31	13
17	10	1	0.1	58	2
18	27	2	0.074074074	30	8
19	55	47	0.854545455	82	28
20	32	1	0.03125	25	7
21	17	2	0.117647059	51	10
22	10	1	0.1	4	5
23	46	2	0.043478261	55	20
24	43	13	0.302325581	109	8
25	47	6	0.127659574	62	11

图 2　美食领域部分用户的领域深度

以 27 号用户和 42 号用户为例，经过计算，在美食领域的领域影响力 $FI_{美食,27}$ 为 10.26，$FI_{美食,42}$ 为 0.57。如表 4 所列，第一行数据是 27 号用户的，第二行数据是 42 号用户的。

表 4　27 号用户与 42 号用户的领域影响力数据

深度	领域嘀咕数/总	信息曝光度	领域粉丝数	粉丝质量	影响力
0.12	283/524	0.00048	168	9.46	10.26
0.20	267/1000	0.0021	149	0.35	0.57

也就是说在美食领域，27 号的影响力比 42 号大，他发布的关于美食的信息传播得更广泛、更具有权威性。而且可以看出：粉丝数与领域影响力并不成正比关系，也进一步验证了我们的设想，即影响力是分领域的，不能笼统地定义用户影响力的大小。通过计算用户在其他领域的领域影响力，发现用户在不同领域的影响力排名是不一致的，即用户在不同的领域具有不相同影响力，验证了我们的设想。

结束语　本研究主要对嘀咕网的用户领域影响力进行了度量。在获取嘀咕网在线用户数据的基础上，采用 SVM 等方法对用户发布的信息内容进行领域分类，建立领域影响力度量模型，提出领域信息质量和领域关系质量两方面的维度；并且采用样本数据来验证了领域影响力模型，度量了用户在不同领域的影响力。发现用户在不同领域的影响力是有差别的，领域影响力与粉丝数量不成正比关系，而是与领域内粉丝数量等因素有关。未来工作将与其他研究工作提出的度量维度进行对比分析，研究提出的领域影响力的稳定性及与其他全局性维度的相关性等问题。本研究提出的方法可以很好地识别用户在不同领域内具有的影响力。对于关注社交媒体分享行为的研究人员、社交媒体产品设计者以及社会化营销方案决策者来说，分领域影响力的提出可以更好地了解他们的目标用户，比如了解意见领袖或达人分享行为特点及受他们影响的用户的特征。本研究还可以用于研究意见领袖与普通用户进行交流的平台所提供的交互功能的局限性及用户使用体验等问题。而最被熟悉的应用还是社区意见领袖的发现、针对性的信息推荐及改善用户获取准确信息的体验。

参 考 文 献

[1] Kwak H, Lee C, Park H, et al. What is Twitter, a social network or a news media [C]// Proceedings of the 19th International Conference on World Wide Web (WWW'10). New York: ACM Press, 2010: 591-600

[2] Ye Shao-zhi, Wu Fei-ix. Measuring message propagation and social influence on twitter. com [J]. Social Informatics Lecture Notes in Computer Science, 2010, 6430: 216-231

[3] Cha M Y, Haddadi H, Benevenuto F, et al. Measuring user influence in Twitter[C]// the Milloin Follower Fallacy Proceedings of International AAAI Conference on Weblogs and Social Media (ICWSM' 10). Washington, Menlo Park: The AAAI Press, 2010: 10-17

[4] 肖宇，许炜，商召玺. 微博用户区域影响力识别算法及分析[J]. 计算机科学，2012，39(9)：38-42
Xiao Yu, Xu Wei, Shang Zhao-xi. Analysis on Algorithms of Identifying Regional Influential Users in Micro-blogging [J]. Computer Science, 2012, 39(9): 38-42

[5] 王菲. 一种改进的 HITS 算法在 SNS 类网站用户影响力评估系统中的应用[D]. 吉林：吉林大学，2012
Wang Fei. An Application of Improved Hits Algorithm in User Influence Valuation System of SNS Websites [D]. Jilin: Jilin University, 2012

[6] Weng Jian-shu, Lim Ee-peng, Jiang Jing, et al. TwitterRank: finding topic-sensitive influential twitterers[C]// Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM'10). New York: ACM Press, 2010: 261-270

[7] Cataldi M, Mittal N, Aufaure M A. Estimating Donain-based User Influence in Social Networks[C]// Proceedings of SAC' 13. Coimbra, Portugal, 2013: 1957-1962

[8] Liu Qing, Peng Geng, Wang Ping. PCA-Based evaluation system of micro-blog influence and an empirical analysis of Sina microblog[C]// Conference on Web Based Business Management (WBM). Shanghai, China, 2012: 697-700

[9] Shuai Xin, Ding Ying, Jerome B, et al. Modeling Indirect Influence on Twitter [J]. International Journal on Semantic Web and Information System (IJSWIS), 2013, 8(4): 20-36

[10] http://www.36kr.com/tag/klout

[11] http://www.leiphone.com/tag/klout

[12] http://www.tfengyun.com/

[13] http://data.weibo.com/

[14] Yan Q, Chen H, Zhang P Y, et al. Microblogging After a Major Disaster in China-a case study of the 2010 Yushu earthquake[C]// CSCW2011. Hangzhou, China, 2011: 19-23

(上接第 49 页)

[14] Deng Ke-feng, Song Jun-qiang, Ren Kai-jun, et al. Graph-Cut Based Coscheduling Strategy Towards Efficient Execution of Scientific Workflows in Collaborative Cloud Environments[C]// GRID. 2011: 34-41

[15] Dhok J, Varma V. Using pattern classification for task assignment in mapreduce[C]// International Institute of Information Technology, Hyderabad, India, 2005

[16] Polo J, Castillo C, Carrera D, et al. Resource-aware Adaptive Scheduling for MapReduce Clusters[C]// ACM/IFIP/USENIX 12th International Middleware Conference (Middleware 2011). 2011