软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(7):1432−1447 [doi: 10.13328/j.cnki.jos.004601]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于机器学习特性的数据中心能耗优化方法*

王肇国, 易 涵, 张为华

(复旦大学 计算机科学技术学院,上海 201203)

通讯作者: 王肇国, E-mail: zgwang@fudan.edu.cn

摘 要: 随着互联网的发展,各种类型的数据呈爆炸式增长.通过机器学习的方法对大量数据进行实时或离线的分析,获取规律性信息,已成为各行业提升决策准确性的重要途径.因此,这些机器学习算法成为各个数据中心运行的主要应用.然而,随着数据规模的增大和数据中心面临的能耗问题的突出,如何实现这些算法的低功耗处理,已成为实现绿色数据中心亟待解决的关键问题之一.为了实现对这些机器算法的绿色计算,首先对运行在数据中心中的关键算法进行了深入的分析,并观察到在这些算法中存在大量的冗余计算.在此基础上,设计和实现了一种面向数据中心典型应用的低功耗调度策略.该算法通过对不同计算部分的输入数据进行匹配来判断计算过程中的冗余部分,并对算法进行调度.实验数据显示,对于数据中心的两种典型应用 *k*-means 和 PageRank,该算法可以实现 23%和 17%的能耗节约.

关键词: 节能;分布式计算;机器学习;MapReduce;PageRank

中图法分类号: TP316

中文引用格式: 王肇国,易涵,张为华.基于机器学习特性的数据中心能耗优化方法.软件学报,2014,25(7):1432−1447. http://www.jos.org.cn/1000-9825/4601.htm

英文引用格式: Wang ZG, Yi H, Zhang WH. Power saving based on characteristics of machine learning in data center. Ruan Jian Xue Bao/Journal of Software, 2014,25(7):1432−1447 (in Chinese). http://www.jos.org.cn/1000-9825/4601.htm

Power Saving Based on Characteristics of Machine Learning in Data Center

WANG Zhao-Guo, YI Han, ZHANG Wei-Hua

(School of Computer Science, Fudan University, Shanghai 201203, China)

Corresponding author: WANG Zhao-Guo, E-mail: zgwang@fudan.edu.cn

Abstract: With the development of the Internet, the scale of data center increases dramatically. How to analyze the data stored in the data center becomes the hot research topic. Programmers resort to the machine learning to analyze unstructured or semi-structured data. Thus, energy efficient machine learning is crucial for green data centers. Based the observation that there is redundant computation in the machine learning applications, this paper proposes a system which can save the power usage by removing the redundant computations and reusing the previous computation results. Evalution shows that for the typical *k*-means and PageRank applications the presented algorithm results 23% and 17% power saving.

Key words: power saving; distributed computing; machine learning; MapReduce; PageRank

随着互联网的发展,各种不同类型的数据呈爆炸式增长.如在 2011 年,Twitter 每天产生 8TB 的数据[1];2012 年,Facebook 每天产生超过 500TB 的数据[2];而作为最大的互联网公司,Google 在 2008 年每天会对 20PB 的数据进行处理[3].为了应对数据规模的不断扩大和对这些数据进行实时分析以及处理的需要,各个互联网公司都不断扩大数据中心的规模和数量.如 Google,Facebook 和微软等公司都在全球拥有多个规模超过十几万个节点的数据中心.

* 基金项目: 上海市科委科技攻关项目(13DZ1108800); 国家自然科学基金(61370081); 国家高技术研究发展计划(863)(2012AA010901)

收稿时间: 2013-12-31; 定稿时间: 2014-03-17

随着数据规模的扩大,如何利用机器学习算法对这些数据进行实时或离线的分析,发掘数据的内在规律,已成为各行业提升决策准确性的重要途径.如:Google 公司基于机器学习算法对每分钟新涌现的 571 个新的网站进行创建索引和排名[4];全球最大的零售商沃尔玛(Wal-Mart)超市需要对每小时超过 100 万笔的交易数据进行分析和处理,从而为其进一步的商业活动提供决策;微软亚洲研究院提出根据现有监测站所提供的空气质量数据以及城市里的其他多种数据来源,运用机器学习技术对大数据加以充分分析,可以实时推断出包含细颗粒物信息的城市空气质量数据[5].因此,机器学习已成为数据中心中必不可少的功能性模块.在这种背景下,各种运行于数据中心的机器学习系统不断涌现,如 GraphLab[6],Mahout[7]以及 MadLINQ[8]等.其中,Mahout 是基于 Hadoop 实现的分布式机器学习类库,已被很多公司(如 Yahoo,Twitter,LinkedIn 等)广泛使用.

然而,随着数据规模的急剧增加以及数据中心数量及规模的不断扩大,数据中心的能量消耗也飞速增长.根据美国环保署报告,2011 年,美国的数据中心全年累计耗电 1 000 亿千瓦时,其耗电量是美国当年总耗电量的 1.5%,而电费则高达 74 亿美元[9].全世界的数据中心在 2010 年共耗去 1 988 亿千瓦时的电力,约是全球 2010 年总发电量的 1.1%~1.5%[10].2011 年,我国数据中心总耗电量达到 700 亿千瓦时,数据中心能耗占全国能源耗电总量的 5%.随着云计算的快速发展,未来 5 年,我国对数据中心流量处理能力的需求将增长 7~10 倍[11].Google 曾经公布其用电量数据,2010 年,Google 用电量为 22.6 亿千瓦时[12].到目前为止,单个数据中心的耗电量已经上升到万千瓦时级别.因此,节约数据中心的能耗已成为当前数据中心所面临的主要挑战之一.如何实现数据中心的绿色处理,也成为研究的热点之一.目前,数据中心的能耗管理主要使用 DVS/DVFS 或休眠/唤醒技术将空闲节点置于低能耗状态,对于数据中心的计算任务的节能,目前大部分研究都是针对 MapReduce 实现任务节能.虽然这些算法都可以在一定程度上降低数据中心的能量消耗,但这些算法在设计过程中并没有充分利用运行于数据中心的典型应用(机器学习任务)的特点,从而使其效果受到一定的限制.

为了进一步对数据中心的能量消耗进行优化,我们首先对数据中心的典型机器学习算法进行了深入的特性分析.机器学习任务属于计算密集型应用,因此其主要的能耗体现在数据的分析和计算上.然而,这些算法在机器学习的过程中需要不断地对数据进行迭代归类和分析,在这个过程中存在大量的冗余计算,这些冗余计算带来了不必要的能耗开销.与此同时,在数据中心中,机器学习主要用于进行数据分析,发现数据潜在的规律,所以用户对其计算结果的精准度并没有非常严格的要求.如在推荐系统中,只需要得到用户倾向于哪个物品,过于精确的计算结果在这里并没有太大意义.基于机器学习算法的以上特点,我们设计和实现了一种面向数据中心中机器学习计算的节能机制.提出了通过匹配输入数据来去除冗余,从而达到节能的方法.其核心思想是:通过匹配两次的输入来分析其输出的相似性,并通过重用计算结果以及合理的调度,达到节能的效果.实验数据显示:该算法在保证算法精确性的基础上,可以有效降低数据中心机器学习算法的冗余计算,从而达到节能的效果.对于 *k*-means 算法,在将误差范围保证在 1.64E−4 之内的前提下,可以最大节约 23%的能耗;对于 PageRank 算法,在将误差值保证在 8.92E−5 之内的前提下,最大可以节约 17%的能耗.为了进一步验证算法的有效性,我们还随机选取了 2 000 个点(共 530K 个点)对 *k*-means 算法进行测试,该系统只会对 0.2%的点的归类产生影响.

本文第 1 节介绍相关工作.第 2 节分析数据中心能耗特点和数据中心机器学习算法的特点.第 3 节介绍本文的节能算法.第 4 节给出实验数据和相关分析.

1 相关工作

随着能源问题在各国国计民生中地位的进一步凸显,如何设计和实现数据中心的节能技术,已成为研究的热点之一.本节将从如下两个方面介绍相关工作:首先,介绍目前已有的面向数据中心的节能技术;然后,介绍针对 MapReduce 框架目前的节能方法.

早在 21 世纪初,研究人员就已经开始研究面向网络服务器的节能技术.Elnozahy 等人[13]在 2003 年提出通过使用动态调整电压(dynamic voltage scaling)和批处理网络请求来降低能耗的方法,该方法的主要思想是:当系统负载较低时,可以将网络请求缓存起来,此时,通过 DVS 技术使得处理器处于低能耗状态;当请求数目超过一定阈值时再提高处理器的供电量,并将这些请求发送给处理器进行处理.该技术在一定程度上牺牲了系统实时

性,在此条件下,降低了处理器的能耗.由于 Web 服务一般是多层应用(包含了交互层、逻辑层和存储层),因此,Horvath 等人提出面向运行多层应用的服务器的节能技术[14].该系统提供了一套完整的算法和优化框架,使得在降低能耗的同时并不影响服务请求的响应时间.实验结果表明:在不影响响应时间的情况下,该系统达到了30%的节能效果.该课题组同时还提出通过提供优先级的方式对客户端归类划分[15],对于不同优先级的客户端使用不同的节能策略.实验结果表明:该技术在降低系统能耗的同时,还提高了系统的吞吐量.Yuan 等人[16]通过对请求的跟踪识别,对不同请求进行归类,并利用 DVS 技术对系统能耗进行控制.随着虚拟化技术的广泛应用,通过 DVS/DVFS 来控制系统的能耗变得困难.Wang 等人[17]设计并实现了面向虚拟化环境节能的 Partic 系统,Partic 系统是一个基于控制理论设计的两层控制系统:主控制器通过采用多输入多输出控制技术为所有虚拟机提供负载均衡,辅助控制器通过动态调整处理器频率以降低功耗.同时,他们还提出了 Co-Con 系统[18],该系统仍然基于两层架构,并保证每个虚拟机在低能耗的状态下仍然可以达到需要的性能.除了通过 DVS 改变处理器频率降低系统能耗之外,还可以通过关闭集群中零负载的物理机以达到节能目的.Chase 等人[19]提出以物理机为单位进行动态资源的分配和调度.Heath 等人[20,21]解决了异构集群中,如何以物理机为单位进行资源调度和分配的问题.为了使得更多机器可以关闭,Chen 等人[22]和 Kaushik 等人[23]提出在不影响整个系统吞吐量的情况下,尽量将服务请求转到某一部分服务器,使得剩下的服务器处于零负载状态,因而可以被移出集群的方法.Lin 等人[24]和 Lu 等人[25]提出根据负载情况动态调整数据中心服务器的在线算法.Kliazovich 等人[26]设计并实现了调度系统 DENS,在计算中心的能耗、单个工作效率和网络通信需求中取得了平衡.

MapReduce 计算框架是目前在数据中心广泛被使用的编程模型,因此它对数据中心的能耗至关重要.目前,已有很多工作关注于研究 MapReduce 模型的能耗以及具体的控制方法.MapReduce 计算框架的实现一般分为两个部分:一个是分布式存储,另一个是分布式计算.GreenHDFS[23]设计并实现了绿色分布式存储的方法,它将系统的存储节点分为两个部分:一部分是热点区域,另一部分是非热点区域.不同类型的数据会存储在不同的区域中,与此同时,非热点区域会长时间处于低能耗状态.Chen 等人[27]研究了 MapReduce 中不同的配置参数对能耗的影响,他们还提供了针对 MapReduce 测量能耗的企业级基准[28].Leverich 等人[29]通过在一定程度上牺牲性能来降低能耗.Lang 等人[30]发现:与计算时只用一部分计算节点并关掉其他计算节点相比,运行一个计算任务应该使用所有的计算节点,当任务完成之后再关掉所有的计算节点,因为这样会达到更好的节能效果.Cardosa 等人[31]发现:可以通过调整或者控制虚拟机的物理位置来达到降低能耗的目的.Chen 等人[32]通过数据压缩的方式降低了系统的能耗.Wirtz 等人[33]使用 DVFS 技术面向计算密集型的 MapReduce 应用进行能耗控制.Li 等人[34]以及 Hartog 等人[35]均提出了不同的面向异构集群能耗的调度机制,使得可以在不严重影响系统吞吐量的情况下达到低功耗的效果.

2 分布式机器学习的特点和能耗问题

本节将以经典的 *k*-means 算法和 PageRank 算法为例,详细分析、介绍分布式机器学习的特点.我们首先对其基本算法进行介绍,然后对 Mahout 中两种算法的实现进行评测,并通过对评测结果的分析和观察,总结出机器学习的基本特点和能耗问题.

2.1 算法描述

2.1.1 *K*-means——聚类算法

K-means 聚类算法在 1957 年被提出,迄今为止仍然被广泛使用.该算法的核心思想是:以迭代计算的方式找出 *K* 个中心点,使得每个点到其所在聚类的中心点距离之和最小,其具体运算过程如下:

(1) 初始化指定 *K* 个中心点.一般可以通过随机的方式选取,也可以通过伞聚类的方法初始化中心点以减少迭代次数;

(2) 计算每一个数据点到各个中心点的距离,将数据点分配到距离最近的那个中心点所在的类;

(3) 通过计算每一个聚类中所有数据点的平均值得到新的中心点;

(4) 如果所有的中心点坐标都保持不变,则意味着结果收敛,停止计算;否则,返回第(2)步继续计算.

这里介绍 Mahout 库中基于 MapReduce 实现的 *k*-means 算法.每个点以向量的形式进行存储和计算,所有点被存储在分布式文件系统的不同数据块中.Map 函数接收一个数据点和当前 *K* 个中心点作为输入,计算出距离该数据点最近的中心点,并将该数据点和距离最近的中心点作为运算结果输出.Combiner 函数接收本地所有 Map 函数的计算输出,并将属于同一类的点相加,同时计算该类中相加的点的数目.Reduce 函数接收所有节点 Map 任务的计算结果,通过计算属于每个类的所有点的平均值来得到新的中心点,并通过原中心点和新中心点的坐标差是否小于某一个阈值来判断该中心点是否收敛:若所有中心点已经收敛,则结束本次计算.

2.1.2 PageRank 基本算法

PageRank 算法由 Google 创始人拉里·佩奇和谢尔盖·布林于 1999 年提出,主要用于表示网页等级的重要性.其中心思想是:每一个页面都使用 *rank* 值来表示它的重要程度,若一个页面的入链程度越多,则这个页面越重要;与此同时,若指向这个页面的入链权重越高,则这个页面所获得的权重也就越高.具体计算步骤如下:

(1) 每一个网页抽象成一个点,根据连接关系构建一个有向图;对每一个点分配一个相同的 *rank* 值,并对于每一个点进行以下两步操作;

(2) 每一个点将其部分 *rank* 值平均分配到其指向的点上;

(3) 每个点将其通过所有入链获得的权重加和以及保留的 *rank* 值得到本点新的 *rank* 值;

(4) 当所有的点收敛时,结束本次运算;否则,继续执行第(2)步.

在 Mahout 库中,PageRank 应用会使用一个分布式矩阵来表示这个有向图.如果第 *i* 行第 *j* 列的值为 1/*n*,则表示点 *j* 有 *n* 条出边,其中一条出边指向点 *i*.同时,程序中会使用一个向量存储所有点的权重,而矩阵和向量的乘积便是所有点获得的最新的权重.所以,PageRank 计算事实上是矩阵和向量不断相乘的过程.Map 函数会传入两个参数:一个是分布式矩阵的某一行,另一个参数是所有点权重的向量.Map 函数会将两个向量相乘,并将最终结果传给 Reduce 函数,Reduce 函数负责将所有 MapTask 传过来的数值组成一个新的权重向量.

2.2 特性分析

为了根据数据中心机器学习算法的特点对能耗进行优化,我们首先收集了 CPU 利用率和网络传输情况,并在此基础上分析了这些算法的能耗特性和计算冗余特性.

2.2.1 CPU 利用率和网络传输情况的分析

首先,我们测量了运行两个不同程序时处理器利用率和网络数据传输情况.为了隔离不同任务之间的影响,我们将输入的大小配置成 HDFS 一个基本块的大小(64MB),因此,计算过程中只会运行一个 MapTask 和一个 ReduceTask.我们让任务在同一个处理器上顺序执行.图 1 和图 2 显示了在 *k*-means 和 PageRank 中该处理器在整个计算过程中的利用率.我们可以看到:*k*-means 和 PageRank 在整个 MapTask 计算阶段的 CPU 利用率均处于较高状态,其中,*k*-means 的 CPU 利用率平均值就达到 60%左右.在网络传输方面,当输入数据为 64MB 时,*k*-means 应用的 MapTask 只会产生 47KB 的数据传输给 ReduceTask;PageRank 应用的 MapTask 也只会产生 1.5MB 左右的数据传输给 ReduceTask.由此可见,这两种典型应用均属于计算密集型任务.

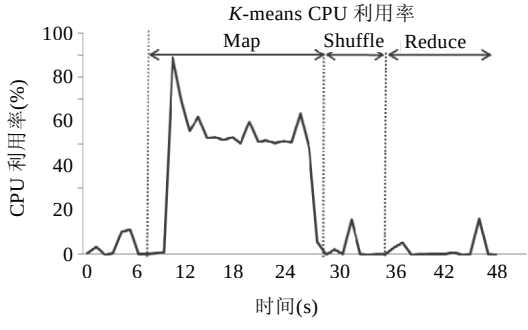

Fig.1 CPU usage (*K*-means)

图 1 CPU 利用率(*K*-means)

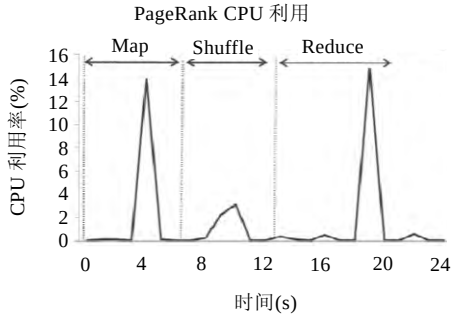

Fig.2 CPU usage (PageRank)

图 2 CPU 利用率(PageRank)

2.2.2 能耗特性

对于应用能耗计算,我们使用功率插座对单个计算节点的实时功率进行检测.图 3 和图 4 展示出 *k*-means 和 PageRank 执行多次迭代计算任务中一次迭代的测试结果,其中,横坐标是单次迭代计算的执行时间,纵坐标是该时间点的单节点功率.在 *k*-means 中,处于空闲状态时,该机器的功率处在 39W 左右的低功耗状态;而当机器的 CPU 利用率提升后,其实时功率可以达到 60W 左右的高功耗状态.从图中也可以看出:在 MapTask 计算过程中,系统大部分时间处于高能耗状态.在 PageRank 中,Map 阶段 CPU 利用率低于 *k*-means 中的 CPU 利用率,所以 Map 阶段的功率略低于 *k*-means 中 60W 左右的功率峰值.与此同时,其曲线与上一节的 CPU 利用率曲线也非常匹配.也就是说,这里的能源消耗主要是用于执行处理器的计算.因此,面向计算密集型的 MapReduce 任务,通过减少处理器的计算可以有效降低系统能耗.

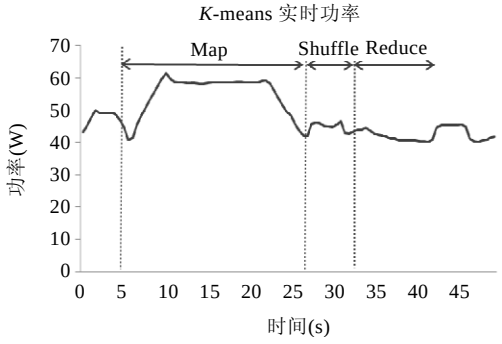

Fig.3 Realtime power (*K*-means)

图 3 实时功率(*K*-means)

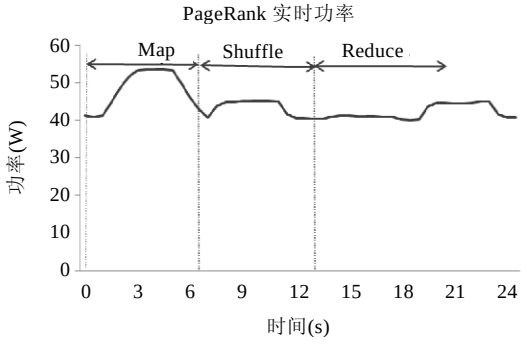

Fig.4 Realtime power (PageRank)

图 4 实时功率(PageRank)

2.2.3 计算冗余性的分析

我们测试了 *k*-means 随着迭代次数的增加,不同聚类的收敛情况.测试将 530K 个点分为 40 个聚类,最大迭代次数为 20 次.从图 5 中可以看出:当第 1 次迭代计算完成时,已有 1 个聚类的中心点处于收敛状态;在第 2 次完成时,已经有 29 个聚类的中心点收敛.然而,由于在计算过程中有一些数据点会从一个聚类被移到另一个聚类,因此我们从图中可以看到:在第 4 次迭代结束时,一部分已经收敛的点会重新变为不收敛点;直到 20 次迭代计算结束,仍然存在 11 个聚类的中心点处于不收敛状态.图 6 和图 7 展示了每一次迭代,每一个数据块中属于收敛的聚类的点的比例.我们分别测试了数据块大小为 16M 和 64M 的情况,从图 7 中可以看出:当数据块为 64M 时,第 3 次迭代之后,在 50%的数据块中,有超过 44%的点属于收敛的聚类;而当数据块为 16M 时,第 3 次迭代之后, 50%的数据块中有超过 97%的点属于收敛聚类.可以看出:减小数据块的大小,能更容易地在数据块的粒度上发现应用中存在的冗余计算.

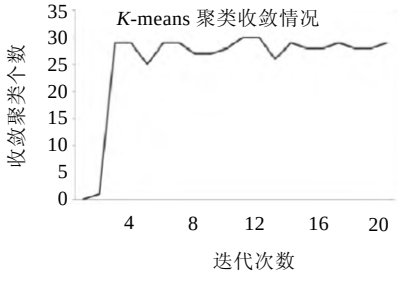

Fig.5 Number of converged clusters in *k*-means

图 5 *K*-means 中聚类收敛情况

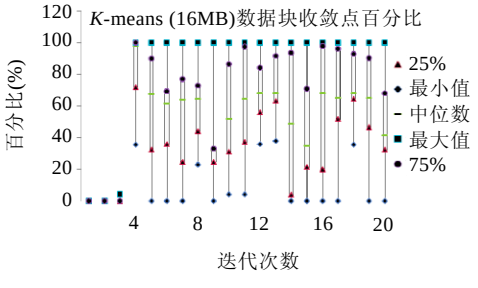

Fig.6 Percentage of converged points in *k*-means (16MB)

图 6 *K*-means 数据块中收敛点百分比(16MB)

对于 PageRank,我们测试了随着迭代次数的增加,每一个点的收敛情况.图 8 显示了收敛点的比例随着迭代次数的变化情况,从图 8 中可以看出:当迭代次数超过 4 时,超过 40%的点都处于收敛状态.由于 Mahout 随机生成的数据,不同的数据块上收敛的数据分布非常均匀,当迭代次数超过 5 时,所有数据块上超过 99%的点收敛;而在第 6 次迭代时,所有的点均已处于收敛状态.

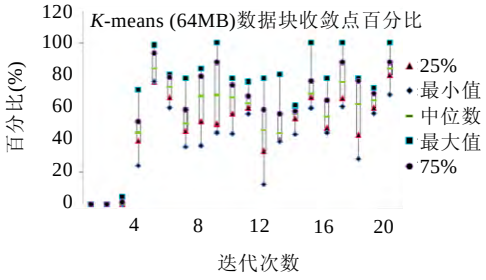

Fig.7 Percentage of converged points in *k*-means (64MB)

图 7 *K*-means(64MB)数据块收敛点百分比

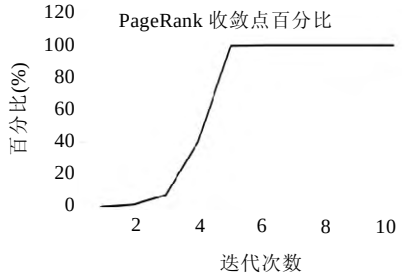

Fig.8 Percentage of converged points in PageRank

图 8 PageRank 收敛点百分比

从以上的结果中可以看出:对于 *k*-means 和 PageRank 应用,在整个任务执行过程中,大部分数据的计算结果在较少的迭代次数后就已经进入收敛状态(即,计算结果的变化在可忽略的范围内),只有少数的数据需要重复计算得到准确的结果.因此,对于已经属于收敛的聚类或者点进行重复计算在一定程度上属于冗余计算.在接下来的章节将介绍我们所提出的系统如何通过匹配计算输入的相似度来去除可能存在的冗余计算,从而达到节能的目的.

3 系统的设计实现

本节介绍了我们的系统的设计和具体实现:首先,介绍该系统的核心算法;然后,介绍总体架构以及系统中每一个功能模块的具体实现;最后,以 *k*-means 和 PageRank 为例介绍特定的应用如何在该系统上运行,以达到节能的目的.

3.1 核心思想

在研究过程中我们发现,面向机器学习的分布式计算对计算结果的精准度并没有苛刻的要求.如聚类算法中各聚类的划分、推荐算法中不同用户和不同商品的相关度以及 PageRank 中每一个点最后计算得到的排名,这些均不需要非常精确的计算结果,用户允许这些计算结果存在一定范围内的误差.由于存在随机初始化等特性,这些算法对于同一数据集合的两次运行结果也不完全相同.事实上,有些程序计算过程中并不能保证计算结果的误差范围.如 *k*-means,PageRank 和推荐算法等程序在计算时,用户倾向于指定一个收敛值,当前后两次计算结果小于该值时,则认为计算收敛.同时,用户需要指定程序运行时的最大迭代次数,当计算次数超过最大迭代次数时,即使计算结果不收敛也会终止计算并向用户输出最终计算结果.由此看来,对于面向机器学习的计算,用户对计算结果的误差有很大的容忍度.

基于以上观察,本文提出通过比较计算的输入数据来判断是否可以重用之前的计算结果,以达到去除冗余计算,节省能耗的目的,与此同时,保证计算结果的误差被控制在一定范围之内.本文的核心技术在于如何判断本次计算可能为冗余计算,为此,首先提出输入数据的相匹配度、输入/输出相关度等概念.

- 相匹配度

在本文中,两次输入数据相匹配度是指使用这两次数据进行输入、输出得到的结果相似程度.如果两次计算结果完全一致,则称这两次输入完全匹配.

- 输入与输出的相关性

我们使用相关性来描述输入数据的变化对计算输出的影响.在研究过程中我们发现:机器学习中的输入数

据可以被分为多个独立的输入,它们与计算结果的相关度可以单独表示.例如,在 *k*-means 中,每一个中心点的变化对本次 MapTask 计算结果的影响可以单独使用一个数值来进行描述,而这些相关度最终组成了一个相关度向量.相关度向量是本文中的核心数据结构,主要用于描述系统的相关性.

在了解了以上基本概念之后,接下来,本文将通过对输入数据本身、不同输入数据的差异度以及输入对输出的影响进行建模,以介绍基本的输入匹配算法.

- 输入向量

我们发现:对于大部分应用的输入,可以用多维向量$\gamma=(r_1,r_2,r_3,\ldots,r_n)$来表示,向量的每一个维度代表一块独立的输入数据.如在 *k*-means 中,r_i 表示每次输入中第 i 个聚类的中心点的位置.

- 相关度向量

我们使用相关度向量$\alpha=(a_1,a_2,a_3,\ldots,a_n)$来描述输入数据和输出数据的相关度,其维度与输入向量相同.其中,a_i 为 0~1 的一个数值,表示数据 r_i 的改变对计算结果影响的可能性.如当 a_i 为 0 时,表示输入数据 r_i 的改变不会影响计算结果;当 a_i 为 0.9 时,则表示 r_i 的改变有 90%的可能性对输出产生影响.

- 差异度向量

我们使用向量$\delta=(d_1,d_2,d_3,\ldots,d_n)$来表示两个输入向量$\gamma$和$\gamma'$的差异度,即:

$$\delta=DIFF(\gamma,\gamma')=(d_1,d_2,d_3,\ldots,d_n) \tag{1}$$

其中,d_i 表示对应的 r_i 和 r_i' 的差异,其具体计算公式如下(其中,Diff 函数由用户根据程序的语义进行实现):

$$d_i = Diff\,(r_i,r_i'),i\in\{1,n\} \tag{2}$$

- 输入差异度

两个输入的差异度 m 可以由两个输入向量的差异度向量与相关度向量进行矢量内积得出,其公式为

$$m=\alpha\cdot\delta \tag{1}$$

当系统得到两个输入的差异度时,可以通过比较该差异度是否小于某一个阈值(该阈值由用户指定)来判断两次计算是否相似.

3.2 总体结构

图 9 展示了基于 Hadoop 设计并实现的系统总体结构.

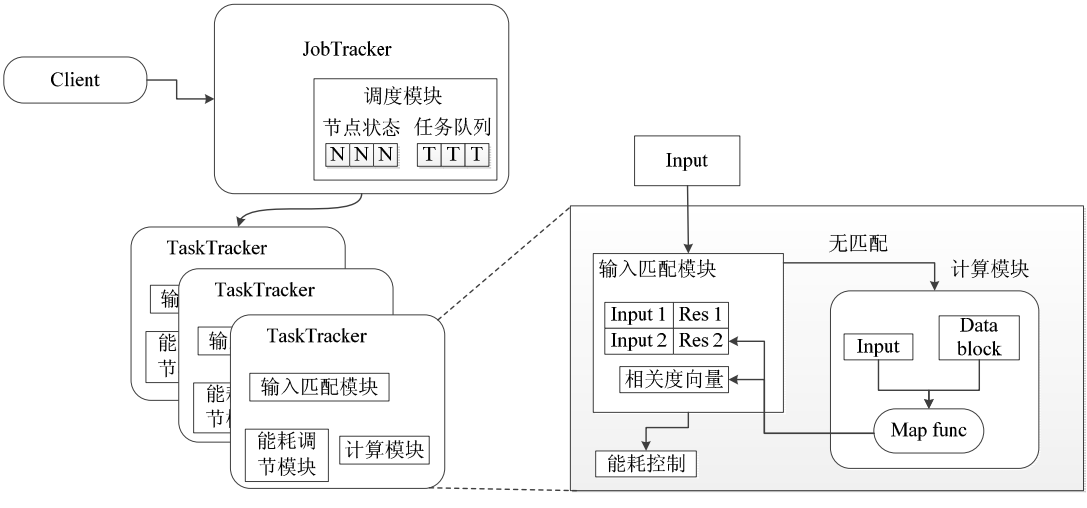

Fig.9 System architecture

图 9 系统结构图

对 JobTracker 节点,我们增加了面向能耗的调度模块,在该模块中会保存所有节点当前的能耗状态,并根据

当前节点的能耗情况进行任务分配.在 TaskTracker 中包含了输入匹配模块、能耗调节模块以及计算模块.当 TaskTracker 接收计算的输入数据时,首先将输入数据传给输入匹配模块,如果找到匹配的输入,则直接将存储的计算结果输出,并通过能耗调节状态将本节点设置为低能耗状态;如果没有相匹配的输入,则调用计算模块进行 Map 或者 Reduce 计算.下面分别介绍每个模块的具体设计和实现.

3.3 输入匹配模块

该模块主要用于计算本次 Task 与之前 Task 的计算输入数据的相匹配度.该模块中保存了之前计算结果以及对应的输入值,该模块还保存了相关度向量.

3.3.1 任务的输入

所有需要进行输入匹配的输入数据均需抽象为输入向量,即 vector⟨InputObj⟩向量类型.如:基于矩阵的机器学习算法大部分输入的都是矩阵,那么矩阵的每一行就看成向量的一个维度.然而与传统应用不同的是,机器学习在计算输入数据的格式和数目上存在多样化,如在 *k*-means 应用中,MapTask 有两个输入,分别为本地所有数据点和上次计算的聚类中心点坐标.而推荐算法中输入一般是两个特征矩阵.PageRank 的输入分别为所有点之间的连接关系和每个点对应的评分值.因此在本系统中,对输入向量中每一个元素的类型提供了较高层的抽象(InputObj)接口,用户可以根据需要,在实现某一类型输入时实现该接口.如在 *k*-means 中,本地数据块中的数据不会有变化,因此在输入匹配的过程中,我们只需存储并比较聚类中心点坐标.因此,输入向量应为所有聚类中心点的集合,并且聚类中心点需要实现 InputObj 接口.而对于类似于推荐算法等数据量较大或者多个输入都存在变化的应用,用户应根据程序的语义,使用提取特征值或者计算哈希等方法对输入数据进行合理的抽象,以减少数据存储缓存量和输入匹配时的计算量.

3.3.2 计算结果的保存

为了去除冗余计算,我们有必要将之前的计算结果在本地进行缓存.事实上,在 Hadoop 中,TaskTracker 会将 ReduceTask 计算的数据存储在分布式文件系统中,因此仅需考虑 MapTask 计算结果的缓存问题.在 Hadoop 系统中,TaskTracker 会将 MapTask 的计算结果存储在本地目录中,ReduceTask 会在 Shuffle 阶段从 MapTask 阶段拉取数据.当整个计算任务结束后,系统会将 MapTask 本地缓存的计算结果删除.在本系统中,当 MapTask 所在的计算任务结束后,并不会马上删除所有 MapTask 的计算结果.事实上,每一次 MapTask 的输入和计算结果的输出路径会保存在 TaskTracker 维护的数据结果中,当后面的输入与已存储的某一次输入相匹配时, TaskTracker 会将该次输入对应的计算结果的路径直接发给 JobTracker.用户也可以手动设置清除点以清除缓存的中间计算结果.

3.3.3 相关度向量

如上所述,相关度向量主要用来表示输入数据的改变对计算结果的影响程度.具体实现时,相关度向量是一个 double 类型的向量 vector⟨double⟩.向量中每一个数值分别表示输入数据不同部分的变化对最终计算结果产生影响的可能性,其中,每一项数值应该在 0~1 之间,0 表示输入的变化不会导致计算结果的变化(无关),1 则表示输入的变化一定会导致输出的变化.

由于相关度向量的计算逻辑依赖于程序的语义.本系统仅提供 updateRelVector 接口用于更新相关度变量的某一个维度.相关度变量对于用户并不可见,用户可以根据程序语义灵活地更新相关度向量.如在 *k*-means 中,用户可以选择在 combine 函数结束后对相关度向量进行计算和更新;在 PageRank 中,应该在每个 Map 函数计算过程中对相关度向量进行更新.相关度向量所有维度默认值为 1.这意味着输入数据的任何改变都会影响最终的输出.

3.3.4 输入数据的匹配

为了判断本次计算是否为冗余计算,系统需要调用 Match 函数来查看本次计算与之前计算的相似度.当 Match 函数返回为 true 时,表示本次计算可以认为是冗余计算.具体地,Match 函数实现如图 10 所示,该函数传入当前的输入向量、之前某一次计算的输入向量以及当前系统中的相关度向量.首先,系统通过调用用户定义的 Diff 函数构建差异度向量,因此,用户需要定义 Diff 函数来计算输入向量的不同;然后,通过差异度向量和相关度

向量进行矢量内积,最后得到两个输入的匹配度;最后,再将匹配度和用户设置的阈值进行比较,如果匹配度小于阈值则匹配成功.在匹配成功的情况下,TaskTracker 会直接将相匹配的输入所对应的保存的输出数据地址发送给 JobTracker,并通知能耗模块匹配成功.

bool **Match** (*vector*⟨*ObjInput*⟩v_1,*vector*⟨*ObjInput*⟩v_2,*vector*⟨*double*⟩*rel*)
Vector⟨*double*⟩*diff*=*rel.clone*(·)
For *i* in 1,…,*n*
    diff[*i*]=***Diff***(v_1[*i*],v_2[*i*])

double $d = \sum diff[i] * rel[i]$
return (*d*<*threhold*)

Fig.10 Basic algorithm of input matching
图 10 具体输入匹配算法的基本实现

3.4 能耗控制模块

当能耗控制模块接收到匹配成功后,会将系统设置为节能状态.对支持网络唤醒的机器,该模块会将该机器置为休眠状态.当整个运行任务进入 Shuffle 阶段后,如果 Reduce 需要从该节点获取数据,它首先会向 JobTracker 发送请求,当获取到具体数据的存储位置后,会向对应的 Map 节点发送网络请求.如果该节点已经处于休眠状态,会被该网络请求唤醒,唤醒后将数据传输给 ReduceTask,然后再次进入休眠状态.对于机器学习类型的处理任务,MapTask 往往会占整个计算任务比较大的时间比重.因此,休眠状态可以较大程度地节省能耗.但其缺点是有可能延迟 ReduceTask 计算任务获取数据的时间,从而延长整个任务计算的时间.

对于不支持网络唤醒机制的机器或者对任务实时性要求较高的机器,能耗模块会直接使用 DVFS 技术将本地处理器的频率降低.当本地再次执行计算时,再将处理器的频率调高.用户可以通过修改配置文件的对应项来选择不同的能耗控制机制.在该机器进入节能状态之前,能耗控制模块需要向 JobTracker 发送请求以更新本计算节点的状态信息.

3.5 调度模块

在 JobTracker 端,我们增加根据当前节点所处的能耗状态进行任务调度的模块.该模块中,有一个优先级队列保存了所有节点及其状态信息.该队列中的任务根据其优先级进行排列,优先级最大的排在队头,优先级最小的排在队尾.在正常运行状态,每个节点的优先级使用以下公式进行计算:

$$P = \frac{\#Task}{ExecutionTime} \tag{4}$$

即,单位时间内执行的任务的数目.这里,我们忽略了整个计算过程中任务的多样性,因此,若一个节点的优先级越高,就意味着其单位时间内执行的任务数目越多.因此,如果可以尽可能地将较多的任务发给它,那么就可以使得在不严重影响性能的情况下空闲出更多的物理节点.这些空闲的物理节点可以被置为低能耗状态,或者休眠状态,以节省能源.在一个物理节点被能耗控制模块置于低能耗状态(休眠或者低处理器频率)之前,它会首先向 JobTracker 发送请求以更新本计算节点的状态信息.调度模块接收到这个信息之后,会更新对应节点的能耗信息.同时,它会将当前节点的优先级除 2,即,$P=P/2$.然后,将该节点重新放入优先级队列中适当的位置.这样做的目的主要是为了避免低能耗的节点再次被分配新的任务,从而避免某一个节点频繁地在低能耗和高能耗之间切换.当系统需要分配一个新的任务到某一个节点时,它会从队头开始查找,直到找到某一个节点拥有空闲的可运行位置,它就会将该任务分配到该节点上.

3.6 应用示例

本节以 *k*-means 和 PageRank 为例来介绍如何使用相关接口,在不严重影响计算精准度的情况下达到节省计算能耗的目的.

3.6.1 *K*-means 算法

K-means 是机器学习中的经典算法,一般需要多个计算任务迭代运行,以达到数据收敛的效果.这里,我们以 *k*-means 为例介绍如何通过去除冗余计算来实现降低能耗的目的.由于 *k*-means 的计算主要在于 MapTask,我们仅去除 MapTask 所导致的冗余计算.

- 输入数据的格式

K-means 的 MapTask 主要负责对本地数据重新归类,因此需要两个输入:一个是需要归类的数据点,这个数据点存在于本地;另一个是上一次迭代计算出的新的中心点的集合.这里,输入向量表示所有中心点的集合,因此,描述中心点的类需要实现 ObjInput 接口.MapTask 初始化时,会将新计算的中心点加载到向量数组中,并从本地文件系统中加载需要归类的数据点.该数据点会作为 Map 函数的输入,在 Map 函数中分别计算每个点到哪个中心点最近,并作为结果输出.

- 相关度的计算

在 *K*-means 中,相关度向量的长度为聚类的数目.在相关度向量中,每一个值代表着该聚类中心点的变化对本地 MapTask 计算输出结果的影响.当本地所有数据均使用 Map 函数处理完之后,会使用一个 combine 函数对已有数据进行合并.事实上,combine 函数会计算本地数据中属于每一个聚类点的数目以及这些数据点坐标各维度数值之和.在 combine 函数计算结束时,用户可以使用如图 11 所示的代码通过调用 updateRelVector 接口对相关向量进行更新,*cs*[*i*].*pointsNumber* 为本地数据点中属于第 *i* 个聚类点的数目,*TotalPointsNumber* 为本地数据点总数目,它们的比值作为相关向量中对应维度的值.该计算的核心思想是,第 *i* 个中心点的相关度可理解为本地数据点中属于该聚类的比例.由于 MapTask 计算是对本地数据进行分类,当本地数据点属于某一聚类的比例较大时,该 MapTask 对该聚类中心点的计算结果影响也较大.也就是说,当本地数据点大部分属于中心点已收敛的聚类时,其他数据对不收敛聚类中心点的计算影响较小.在后面这种情况下,计算则可能为冗余计算.

```
For each cs[i] in Clusters
    rel=cs[i].pointsNumber/TotalPointsNumber
    updateRelVector(i,rel)
```

Fig.11 Relevance vector update algorithm of *k*-means

图 11 *K*-means 相关度向量更新算法

- 匹配方法

在 MapTask 初始化完成之后,系统会调用 Match 函数对本次 MapTask 的输入进行输入匹配.而在 Match 函数中,会调用用户实现的 Diff 函数来计算差异度向量.*K*-means 中 Diff 函数的具体实现如图 12 所示.由于输入向量的每一个维度都是聚类的中心点,因此 Diff 函数传入的是两个输入向量在相同维度上的中心点.通过函数的定义我们可以看出,中心点在具体实现过程中也是多维度的向量.该函数首先计算了两个中心点的距离平方,当该值小于 *conv* 时返回 0,表示两个点可以被认为相等;而当该值大于 *conv* 时,返回该值与 *conv* 的比值.*conv* 变量为用户在 *k*-means 应用中设置的收敛值,当中心点变化范围小于收敛值时,则认为该中心点所在的聚类收敛.

```
double Diff(Point p1,Point p2)
double  d =|∑(p1[i]* p1[i]− p2[i]* p2[i])|
if d<conv
    return 0
else
    return d/conv
```

Fig.12 Implementation of Diff interface in *k*-means

图 12 *K*-means 中,Diff 接口的实现

3.6.2 PageRank 算法

PageRank 和 *k*-means 都是机器学习中的经典算法,并且需要多次迭代计算才能达到数据收敛的效果.下面,

我们介绍使用本系统实现 PageRank 节能算法.

- 数据的输入格式

PageRank 中,MapTask 有两个输入:一个是本地存储的点之间的连接关系,这个输入会以矩阵的形式存储在本地,且在整个计算过程中不会有任何变化;另一个是每一个点的权重值,该值为上一轮计算的结果.每一个点的权重值组成需要匹配的输入向量,因此,描述每一个点及其对应的权重值的类需要实现 ObjInput 接口.与 *k*-means 类似,当 MapTask 初始化时,会将上一轮所计算的所有点的权重加载到向量数组中,并使用 setInput 更新系统中保存的输入向量.

- 相关度向量的计算

就目前的实现来看,在 PageRank 应用中,相关度向量的长度为所有点的数目.相关度向量中只有两个数值 0 或 1,用于表示这个点是否属于本地数据块.与 *k*-means 不同,相关度向量与计算结果无关,只与本地点的关联关系有关.因此,在数据被存储到本地时,用户就可以计算相关度向量.具体算法如图 13 所示,当数据被加载到本地时,遍历数据中的所有点,将相关向量中该点对应的维度置为 1.因此,PageRank 中输入匹配的核心算法是:两次输入中,如果存储在本地的所有点的权重值相等,那么这两次输入可以认为相匹配.

```
Clear relVector
For each point_i in local data
    updateRelVector(i,1)
```

Fig.13 Relevance vector update algorithm of PageRank

图 13 PageRank 相关度向量更新算法

- 匹配方法

在 PageRank 中,当一个点的权重值变化范围小于用户设置的收敛度时,我们认为该点已经收敛.因此,比较两个点的 Diff 函数如图 14 所示.由于输入向量为所有点的权重值,因此,Diff 函数的两个参数为同一个点两次不同计算的权重值.当这两个权重值小于用户设置的收敛度时,则直接返回 0;否则,返回 1(表示两个点不匹配).由于输入向量和相关度向量的长度均等于所有点的数目,因此当点较多时,匹配计算的时间会比较长,进而影响系统的性能.我们通过设置匹配计算的频率来降低匹配计算对系统性能的影响,系统提供了 setSampleFreq 的接口,通过该接口,可以指定匹配计算调用的周期.在具体实验中,我们将 PageRank 的 *sample* 值设置为 5,也就是说,平均执行 5 个 MapReduce 计算会进行一次匹配计算.

```
double Diff(Rank r1,Rank r2)
d=|r1−r2|
if d<conv
    return 0
else
    return 1
```

Fig.14 Input matching algorithm of PageRank

图 14 PageRank 输入匹配算法

4 实 验

本节将具体介绍对本系统采用的评测方法以及相应的测试结果,具体内容包含实验环境、节能效果以及计算误差的分析.

4.1 实验环境

我们首先使用功率插座测试了配置 Intel i7-4770 处理器、32GB 内存和 2TB 硬盘的单节点各状态的能耗情况.评测根据一个 MapTask 分配在一个节点上的原则进行.对于 *k*-means 应用,我们使用 GraphLab 中的聚类数据生成程序产生 530K 个数据点(基于 40 个聚类)作为输入,每个数据点为 60 维的向量,最终将这些点归并到 40

个聚类中.最大迭代次数设置为 20 次,HDFS 的每一个数据块大小设置为 16MB,每一次 MapReduce 任务会分配 15 个 MapTask 和 1 个 ReduceTask.对于 PageRank 应用,由于受现有数据大小的限制,我们定义了一个随机的数据生成程序,具体过程如下:对于每个点,利用基于平均值的随机分布决定该点的连接数 n,然后从其余点中随机选取 n 个点作为连接的终点.我们使用自定义的数据生成程序产生 400K 个数据点(205MB),其中,每个点的平均连接数为 30,最大迭代次数设置为 10 次.首先,我们进行了应用能耗的评估测试.

4.2 节能效果

本节介绍了能耗的评估测试结果.对于 *k*-means 和 PageRank 应用,输入的差异度阈值被设置成 0.05,即,若差异度小于 5%,则认为两次输入匹配.

首先介绍 *k*-means 和 PageRank 应用中的电能计量方法和节能效果.对于原应用的消耗计算,我们通过功率插座获得单个节点执行 MapTask 的实时功率,从而获得该节点的消耗电能.目前,本系统针对的是同构数据中心,因此可以认为所有节点的功率情况相似,将所有节点的 MapTask 的电能累加,从而获得 Map 阶段消耗的电能.然后,利用功率插座获得执行 ReduceTask 节点的实时功率,即,Reduce 阶段消耗的电能.之后,通过累加即可获得单次迭代执行的总电能消耗.基于每次迭代计算的相似性,整个计算的总能耗可以由单次迭代消耗的电能乘以迭代次数得到.对于优化后的电能计算,我们分别考虑了两种处理方式.

- 一种是简单地将处理冗余的数据块的 MapTask 节点空闲,我们将其称为基于空闲的节能方法.在这种方式下,原来 MapTask 用来计算的时间都将变成空闲时间,其功率计算可以通过将原来的计算时间的功率替换成空闲状态下单节点的功率来完成.考虑到在 Shuffle 阶段需要处理的数据相同,其功率以及消耗的电能可以认为不变.这里主要是采用替换的方法来估计节省下来的电能,通过相同的方法计算应用的总电能消耗;
- 另一种是采用休眠的方式来处理原来 MapTask 的计算时间,即通过休眠的方式处理冗余计算,通过 ReduceTask 的数据请求或者 JobTracker 的任务分配请求唤醒该节点,我们将其称为基于休眠的节能方法.这种情况下,该节点的功率得到进一步降低,不过需要相应的硬件支持,总电能的计算方式与前一种方法相似,可以通过将原来的计算时间功率替换成休眠状态下的功率来估计电能消耗.

图 15 和图 16 中显示了 *k*-means 和 PageRank 应用中单次迭代的 MapTask 的实时功率对比情况.

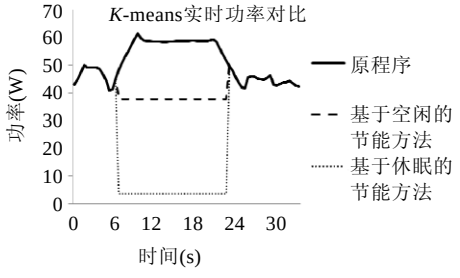

Fig.15 Realtime power of *k*-means

图 15 *K*-means 实时功率对比

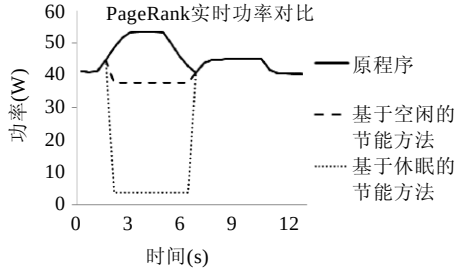

Fig.16 Realtime power of PageRank

图 16 PageRank 实时功率对比

最后,通过计算每次迭代所有的 MapTask 的电能消耗和 ReduceTask 的电能消耗,我们可以获得每次迭代的电能消耗.图 17 和图 18 显示了 *k*-means 和 PageRank 单次迭代执行的优化前后的电能对比情况,从中可以看出:在 *k*-means 应用中,通过简单的基于空闲的节能方法,我们也能获得 10%的电能节省,而通过基于休眠的节能方法,原程序中 27%的消耗电能可被节省;在 PageRank 应用中,基于空闲的节能方法能够节省 10%的电能,而通过基于休眠的节能方法能够节省 37%的电能.在 *k*-means 实际运行过程中,针对 20 次的迭代执行,从第 3 次迭代执行之后,在总共 15 个数据块中,9 个数据块拥有超过 95%的点属于已经收敛的聚类,所以之后的 17 次迭代执行中能够应用相应的节能方法;而在 PageRank 中,第 5 次迭代执行后,可以发现全部 7 个数据块的所有输入均相匹

配,因此均可被认为是冗余计算,所以后面的 5 次迭代都可进行节能优化.

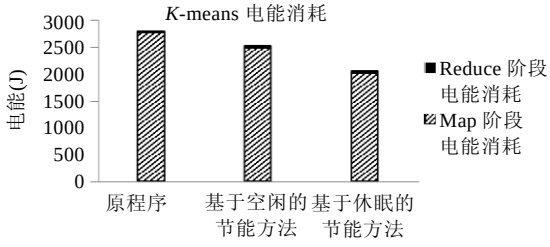

Fig.17 Energy consumption of *k*-means

图 17 *K*-means 电能消耗

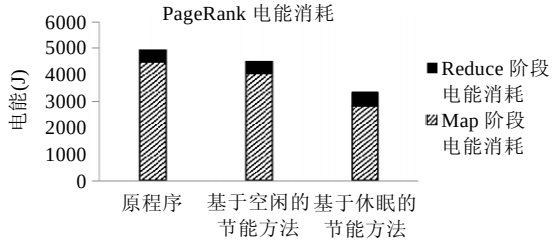

Fig.18 Energy consumption of PageRank

图 18 PageRank 电能消耗

图 19 展示出 *k*-means 应用 20 次迭代总电能消耗的情况,图 20 显示了 PageRank 应用 10 次迭代总电能消耗的情况.其中,基于空闲的节能方法能够为 *k*-means 应用节省 7%左右的电能消耗,而通过基于休眠的节能方法,整个 *k*-means 应用中将近 23%的电能消耗能够被节省;对于 PageRank 应用,基于空闲的节能方法能够节省 5%的总电能消耗,而通过基于休眠的节能方法,整个 PageRank 应用中将近 17%的电能消耗能够被节省.

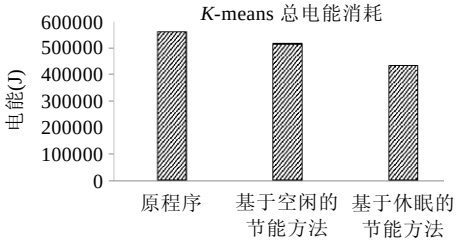

Fig.19 Total energy consumption of *k*-means

图 19 *K*-means 总电能消耗

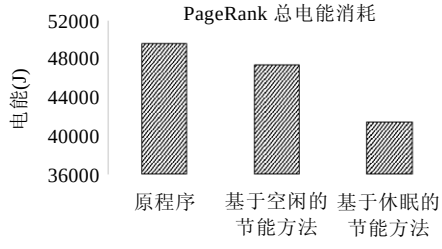

Fig.20 Total energy consumption of PageRank

图 20 PageRank 总电能消耗

4.3 计算误差

下面,我们分别对于两个应用进行计算误差方面的分析.

- 对于 *k*-means 应用

通过比较收敛的聚类中心点变化来表示去除冗余计算而导致的误差.对于计算后得到的每个聚类的中心点,具体计算公式如下:

$$deviation = \frac{|\alpha' - \alpha|}{mean\ distance} \tag{5}$$

其中,α'表示通过使用本系统在节能状态下计算出的聚类中心点,α表示原系统计算出的对应的聚类中心点,*mean distance* 是所有聚类中心点的平均距离.我们用这种方式来说明一个聚类在两次计算的稳定性:如果误差较小,说明两次计算得出的聚类具有较高的稳定性.具体误差值如图 21 所示,平均误差为 2.39E−5,其中,最大值为 1.64E−4,50%的点误差值小于 2E−5.为了查看对所有数据点的影响,随机选取 2 000 个点(输入为 420K),与 Mahout 相比,其中仅 0.2%的点在最终归类的结果上会有影响.

- 对于 PageRank 应用

通过比较去除冗余计算后所有点的结果与原程序计算所得结果来显示去除冗余计算所带来的误差,具体计算公式如下:

$$deviation = \frac{|R_i' - R_i|}{R_i} \tag{6}$$

其中, R_i' 表示其中第 i 个点使用本系统计算后的结果,R_i 表示第 i 个点的原系统的计算结果.对于 PageRank,我们设置最大迭代次数为 10 次,在本系统 5 次迭代后,所有 MapTask 均处于收敛状态.图 21 展示了其误差分布图,其中,最大误差值为 8.92E–5,50%的点的误差值小于 1.37E–31,平均误差为 1.27E–5.

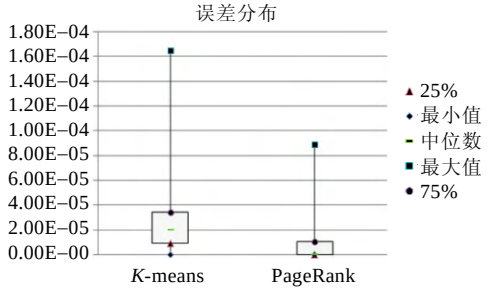

Fig.21 Error distribution of PageRank and *k*-means

图 21 PageRank 和 *k*-means 的误差分布

5 结束语

本文针对数据中心的能耗问题,利用机器学习算法对结果误差的容忍和计算自身的冗余性,对计算的输入进行建模,并通过匹配输入以及计算结果重用的方式去除不必要的冗余计算,从而达到节能的目的.本文从能耗和精确度两个方面对系统进行了分析,实验结果表明:在保证误差控制在一定范围内的前提下,本系统有效地节约了系统的能耗.下一步的工作将具体研究本系统在异构环境下的工作情况,以及对更多的数据中心应用进行测试.

致谢 在此,我们向对本文工作给予支持和建议的同行,尤其是上海交通大学陈榕老师表示感谢.

References:

[1] Krikorian R. Twitter by the numbers. 2010. http://www.slideshare.net/raffikrikorian/twitter-by-the-numbers

[2] Tam D. Facebook processes more than 500 TB of data daily. 2012. http://www.cnet.com/news/facebook-processes-more-than-500-tb-of-data-daily/

[3] Wikipedia, Petabyte. 2014. http://en.wikipedia.org/wiki/Petabyte

[4] Wikibon, a comprehensive list of big data statistics. 2012. http://wikibon.org/blog/big-data-statistics/

[5] 李洁.微软研究院展示大数据与机器学习的魅力.2013.http://datacenter.ctocio.com.cn/132/12765132.shtml

[6] Low Y, Gonzalez J, Kyrola A, Bickson D, Guestrin C, Hellerstein JM. Graphlab: A new framework for parallel machine learning. arXiv preprint arXiv:1006.4990. 2010.

[7] The apache software foundation, what is apache mahout. 2014. http://mahout.apache.org/

[8] Qian ZP, Chen XW, Kang NX, Chen MC, Yu Y, Moscibroda T, Zhang Z. MadLINQ: Large-Scale distributed matrix computation for the cloud. In: Proc. of the 7th ACM European Conf. on Computer Systems. ACM Press, 2012. 197–210. [doi: 10.1145/2168836.2168857]

[9] Williams C. What is a green data center. 2011. http://www.mnn.com/green-tech/computers/stories/what-is-a-green-data-center

[10] Koomey J. Growth in data center electricity use 2005 to 2010. 2011. http://www.analyticspress.com/datacenters.html

[11] 宿艺.专家称2011年我国数据中心耗电量达700亿千瓦时.2012. http://news.cnw.com.cn/news-china/htm2012/20120316_243401.shtml

[12] ARLnow.com. Google’s energy usage, compared to Arlington. 2011. http://www.arlnow.com/2011/09/12/googles-energy-usage-compared-to-arlington/

[13] Elnozahy M, Kistler M, Rajamony R. Energy conservation policies for Web servers. In: Proc. of the 4th USENIX Symp. on Internet Technologies and Systems, Vol.4. 2003. 8. http://dl.acm.org/citation.cfm?id=1251468

[14] Horvath T, Abdelzaher T, Skadron K, Liu X. Dynamic voltage scaling in multitier Web servers with end-to-end delay control. IEEE Trans. on Computers, 2007,56(4):444−458. [doi: 10.1109/TC.2007.1003]

[15] Horvath T, Skadron K, Abdelzaher T. Enhancing energy efficiency in multi-tier Web server clusters via prioritization. In: Proc. of the 2007 IEEE Parallel and Distributed Processing Symp. (IPDPS 2007). 2007. 1−6. [doi: 10.1109/IPDPS.2007.370509]

[16] Yuan L, Zhan JF, Sang B, Wang L, Wang HN. PowerTracer: Tracing requests in multi-tier services to save cluster power consumption. Technical Report, Institute of Computing Technoly, Chinese Academy of Sciences, 2010.

[17] Wang YF, Wang XR, Chen M, Zhu XY. Partic: Power-Aware response time control for virtualized web servers. IEEE Trans. on Parallel and Distributed Systems, 2011,22(2):323−336. [doi: 10.1109/TPDS.2010.79]

[18] Wang XR, Wang YF. Coordinating power control and performance management for virtualized server clusters. IEEE Trans. on Parallel and Distributed Systems, 2011,22(2):245−259. [doi: 10.1109/TPDS.2010.91]

[19] Chase JS, Anderson DC, Thackar PN, Vahdat AM, Doyle RP. Managing energy and server resources in hosting centers. In: Proc. of the 18th Symp. on Operating Systems Principles. 2001. [doi: 10.1145/502034.502045]

[20] Heath T, Diniz B, Carrera EV, Jr. Meira W, Bianchini R. Self-Configuring heterogeneous server clusters. In: Proc. of the Workshop on Compilers and Operating Systems for Low Power. 2003. http://www.cs.rutgers.edu/~ricardob/papers/colp03.ps.gz

[21] Heath T, Diniz B, Carrera EV, Jr. Meira W, Bianchini R. Energy conservation in heterogeneous server clusters. In: Proc. of the ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP). 2005. 186−195. http://dl.acm.org/citation.cfm?id=1065969

[22] Chen G, He WB, Liu J, Nath S, Rigas L, Xiao L, Zhao F. Energy-Aware server provisioning and load dispatching for connectionintensive Internet services. In: Proc. of the NSDI. 2008. 337−350. http://dl.acm.org/citation.cfm?id=1387613

[23] Kaushik RT, Bhandarkar M. GreenHDFS: Towards an energy-conserving storage-efficient, hybrid hadoop compute cluster. In: Proc. of the USENIX Annual Technical Conf. 2010. http://dl.acm.org/citation.cfm?id=1924927

[24] Lin MH, Wierman A, Andrew LL, Thereska E. Dynamic right-sizing for power-proportional data centers. IEEE/ACM Trans. on Networking, 2013,21(5):1378−1391. [doi: 10.1109/TNET.2012.2226216]

[25] Lu T, Chen MH, Andrew LL. Simple and effective dynamic provisioning for power-proportional data centers. IEEE Trans. on Parallel and Distributed Systems, 2013,24(6):1161−1171. [doi: 10.1109/TPDS.2012.241]

[26] Kliazovich D, Bouvry P, Khan SU. DENS: Data center energy-efficient network-aware scheduling. Cluster Computing, 2013,16(1): 65−75. [doi: 10.1007/s10586-011-0177-4]

[27] Schurgers C, Srivastava MB. Energy efficient routing in wireless sensor networks. In: Proc. of the Military Communications Conf. (MILCOM 2001). Communications for Network-Centric Operations: Creating the Information Force, Vol.1. IEEE, 2001. 357−361. [doi: 10.1109/MILCOM.2001.985819]

[28] Chen YP, Ganapathi AS, Fox A, Katz RH, Patterson DA. Statistical workloads for energy efficient mapreduce. Technical Report, No. UCB/EECS-2010-6, University of California at Berkeley, 2010.

[29] Leverich J, Kozyrakis C. On the energy (in) efficiency of hadoop clusters. ACM SIGOPS Operating Systems Review, 2010,44(1): 61−65. [doi: 10.1145/1740390.1740405]

[30] Lang W, Patel JM. Energy management for mapreduce clusters. Proc. of the VLDB Endowment, 2010,3(1-2):129−139.

[31] Cardosa M, Singh A, Pucha H, Chandra A. Exploiting spatio-temporal tradeoffs for energy efficient MapReduce in the cloud. Technical Report, No.10-008, Department of CSE, University of Minnesota, 2010.

[32] Chen YP, Ganapathi A, Katz RH. To compress or not to compress-compute vs. IO tradeoffs for mapreduce energy efficiency. In: Proc. of the 1st ACM SIGCOMM Workshop on Green Networking. ACM Press, 2010. 23−28. [doi: 10.1145/1851290.1851296]

[33] Wirtz T, Ge R. Improving MapReduce energy efficiency for computation intensive workloads. In: Proc. of the Int'l Green Computing Conf. and Workshops (IGCC). IEEE, 2011. 1−8. [doi: 10.1109/IGCC.2011.6008564]

[34] Li Y, Zhang HL, Kim KH. A power-aware scheduling of MapReduce applications in the cloud. In: Proc. of the Int'l Conf. on Dependable, Autonomic and Secure Computing (DASC). IEEE, 2011. 613−620. [doi: 10.1109/DASC.2011.111]

[35] Hartog J, Fadika Z, Dede E, Govindaraju M. Configuring a MapReduce framework for dynamic and efficient energy adaptation. In: Proc. of the 2012 IEEE 5th Int'l Conf. on Cloud Computing (CLOUD). IEEE, 2012. 914–921. [doi: 10.1109/CLOUD.2012.137]

王肇国(1986－),男,吉林长春人,博士生,主要研究领域为多核内存数据库,全系统模拟器.
E-mail: zgwang@fudan.edu.cn

易涵(1990－),男,硕士生,主要研究领域为分布式计算,多核内存数据库.
E-mail: ken.yihan1990@gmail.com

张为华(1974－),男,博士,副教授,CCF 会员,主要研究领域为计算机体系结构,软件纠错,编译器优化.
E-mail: zhangweihua@fudan.edu.cn