一种能效优化的 MapReduce 资源比模型

宋 杰[1)] 刘雪冰[1)] 朱志良[1)] 李甜甜[2)] 赵大哲[2)] 于 戈[2)]

[1)](东北大学软件学院 沈阳 110819)

[2)](东北大学信息科学与工程学院 沈阳 110819)

摘 要 随着云计算的快速发展,IT 资源规模的不断扩大导致能耗问题日益凸显.为降低 MapReduce 编程模型带来的高能耗,文中研究 Map/Reduce 任务的资源消费特征及该特征与能效的关系,旨在寻找一种能够指导资源分配和任务调度的资源模型,进而实现能效优化.文中提出任务的能效与任务被分配的资源量无关,而与其被分配的各种资源的资源量比例相关,且存在一个"最佳资源比"使得能效达到最高.基于此,文中首先提出了普适的资源和能效模型,从模型层面证明最佳资源比和能效之间的关系,量化空闲资源量和空闲能耗;随后分析 MapReduce 编程模型,将普适资源比模型变换到 MapReduce 下.通过抽象的数据的"生产者-消费者"模式,求解 Map/Reduce 任务的最佳资源比;最后,通过实验从任务能效和空闲能耗两个角度证明了最佳资源比的存在,并根据实验结果,对 MapReduce 执行过程进行划分,给出了部分 Map/Reduce 任务的最佳资源比.最佳资源比的提出和求解将有利于基于该最佳资源比的任务调度和资源分配算法的研究,进而实现 Map/Reduce 任务能效的提高.

关键词 云计算;能效;资源比;MapReduce

中图法分类号 TP311 **DOI 号** 10.3724/SP.J.1016.2015.00059

An Energy-Efficiency Optimized Resource Ratio Model for MapReduce

SONG Jie[1)] LIU Xue-Bing[1)] ZHU Zhi-Liang[1)] LI Tian-Tian[2)] ZHAO Da-Zhe[2)] YU Ge[2)]

[1)](*Software College*, *Northeastern University*, *Shenyang* 110819)

[2)](*School of Information Science and Engineering*, *Northeastern University*, *Shenyang* 110819)

Abstract With the rapid development of cloud computing, the explosion of IT resources brings serious high energy consumption problems. To optimize energy consumption of MapReduce applications, we analyze resource consumption characteristics of Map/Reduce task and its relationships with energy efficiency, and aim at finding a resource model that could provide guidance on how to optimize energy-efficiency by using more reasonable resource allocation and task scheduling approach. The paper proposes that the energy efficiency of task is irrelevant to the resource quantity allocated to it, but relevant to a certain ratio of multiple allocated resources, and there exists a best resource ratio that maximizes the energy efficiency. On the basis of these theories, firstly, a universal resource and energy efficiency model is proposed, which proves the relationship between best resource ratio and energy efficiency, then idle resource quantity and idle energy consumption is quantified; Secondly, the MapReduce programming model is analyzed, then, universal resource ratio model is adapted to MapReduce; afterwards, the paper derives the value of best resource ratio of MapReduce task through an abstract producer-consumer model of data processing; finally,

收稿日期:2013-10-15;最终修改稿收到日期:2014-09-05.本课题得到国家自然科学青年基金(61202088)、中国博士后科学基金面上项目(2013M540232)、中央高校基本科研业务费专项资金(N120817001)、教育部博士点基金(20120042110028)资助.**宋 杰**,男,1980 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为云计算、海量数据计算、高能效计算. E-mail: songjie@mail.neu.edu.cn. **刘雪冰**,男,1990 年生,硕士研究生,主要研究方向为高能效计算. **朱志良**,男,1962 年生,博士,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为复杂网络. **李甜甜**,女,1989 年生,博士研究生,中国计算机学会(CCF)会员,主要研究方向为高能效计算. **赵大哲**,女,1960 年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为软件工程. **于 戈**,男,1962 年生,博士,教授,博士生导师,主要研究领域为数据库理论.

through experiments, the existence of best resource ratio is proved from perspectives of energy efficiency and idle energy consumption. The MapReduce execution process is divided into phrases on the basis of experiment results, and best resource ratio is also derived. The proposed best resource ratio of task and its derivation could benefit the further study of resource allocation and task scheduling algorithms, which enables energy efficiency optimization of MapReduce applications.

Keywords cloud computing; energy efficiency; resource ratio; MapReduce

1 引 言

云计算已成为IT服务的主流技术[1],具有较高的成熟度,如亚马逊、Google、Salesforce等知名公司均掌握成熟的云计算技术①②[2],并将其进行推广.在众多云计算技术中,MapReduce[3]编程模型最为流行.MapReduce是分布式地并行完成大规模运算的一种有效技术,MapReduce框架完成输入数据的分割、任务调度、资源分配、节点容错、节点通信和结果数据保存等复杂操作,使得应用程序能够便捷地在大量普通配置的计算机上并行执行.MapReduce是当前研究的热点,现有研究或对MapReduce进行性能优化,或修改现有数据分析算法以适应MapReduce模型,或在MapReduce的基础上寻找更为高效的编程模型,而本文将主要从MapReduce作业能效优化的角度展开研究.

随着IT设施功率的逐渐增加和MapReduce集群规模的不断扩大,集群能耗过大已经成为云计算面临的一个难题.《纽约时报》估计全球互联网数据中心的用电功率可能达300亿瓦特,相当于30个核电站的供电功率.而数据中心的耗电量仅有6%~12%是被用于网站计算的,其余均在维持服务器工作状态时被无谓消耗③.此外,绿色和平组织预测,到2020年,全球主要IT运营商的能耗将达到2万亿千瓦时,超过德、法、加和巴西等4国的能耗总和④.从环境角度,数据中心在消耗大量电力的同时会产生惊人的碳排放.在美国,100 MW发电站会花费6000万到1亿美元并排放5000万吨CO_2,目前全球数据中心的CO_2排放量相当于阿根廷整个国家的CO_2排放量,全球IT产业的碳排放占温室气体总排放的2%[4].在此背景下,如何在数据中心里采用高能效的方式运行MapReduce作业,进而降低能耗实现绿色计算,成为当今研究的热点问题.

按照物理学观点,能效是指在能源利用中,发挥作用的与实际消耗的能源量之比.从消费角度看,能效是指为终端用户提供的服务与所消耗的总能源量之比.所谓提高能效,就是指用更少的能源投入带来同等的能源服务,可以认为能效是性能和能耗之比.本研究着重于MapReduce能效优化.之所以选择MapReduce是因为它是云计算环境中主流的编程模型,大多Internet服务商的核心业务均采用MapReduce实现.MapReduce模型有多种实现,如Google MapReduce、Apache Hadoop、Map-Reduce-Merge、多核和多处理器系统的MapReduce等.本文选择开源且被开发者和研究人员广泛接受的Apache Hadoop作为云计算环境.当前主流的能效优化思路为负载集中和关闭空闲节点,然而MapReduce集群中的节点不仅完成运算任务,还存储海量数据.节点需提供"Always-On"的数据服务,且由于集群自身特性使开关节点的代价很高,因此无法通过临时关闭节点的"ON-OFF"算法[5-6]实现节能.我们前期研究证明[7],MapReduce集群节点的资源使用率不高,产生这一情况的原因可以归结为节点资源等待,如CPU会因等待I/O操作而阻塞,或等待网络数据传输.因节点资源被动空闲而产生空闲资源,空闲资源越多,空闲能耗越大,能效就越低.我们认为,一个任务所分配的资源越多,执行性能越好,但能耗越高,因此能效未必有所提高;相反的,若资源分配不合理,部分资源过量,则空闲资源也会增加,空闲能耗增加,能效反而降低.我们提出一个假设,当任务的资源分配满足一定比例时,无论分配的资源量多少,任务的能效值不变,且存在一个最佳的比例,使能效最高,我们称这一最佳的资源分配比例为最佳资源比.本文从理论推导和实验验证这两个角度证明了最佳资源比的存在,并给出了

① Amazon EC2. http://aws.amazon.com/ec2, 2013
② Google App Engine. http://code.google.com/appengine/, 2013
③ James G. Power, Pollution and the Internet. The New York Times. http://www.nytimes.com/2012/09/23/technology/, 2012
④ GreenPeace. http://www.greenpeace.org/international/en/, 2013

求解 MapReduce 任务最佳资源比的过程. 最佳资源比的提出和求解将有利于进一步研究基于该最佳资源比的任务调度和资源分配算法，进而有效提高 MapReduce 任务的能效.

综上所述，本文研究 MapReduce 任务的资源消费特征，提出了任务的最佳资源比模型，定义任务达到能效最优时的资源占有情况. 该模型是普适的，当任务为单一资源密集型任务时，忽略不计的资源的比值被简化为无穷小的形式，以保证比值的有效性. 资源分配问题是一个经典问题，在不同的运算环境中有不同的设计目标，如性能、公平性和适应性等；本文在优化能效这一目标下，提出了采用资源量的比值而非资源量作为资源分配标准，最佳资源比是优化能效的新方法，尚未见类似研究.

本文首先定义一个针对任务而言的普适的资源模型，包括资源、任务、任务能效和任务最佳资源比；随后将普适模型运用到 MapReduce 中，对 MapReduce 任务进行进一步划分；接着利用资源的“生产者-消费者”模式求解 MapReduce 最佳资源比. 最后，我们通过能效和空闲能耗两个角度证明最佳资源比的存在性和合理性，并在实验基础上给出了部分 MapReduce 任务的划分方法和最佳资源比. 本文第 2 节介绍相关工作；第 3 节和第 4 节分别介绍普适资源比模型以及其在 MapReduce 中的应用；第 5 节从理论上求解 MapReduce 的最佳资源比；第 6 节通过实验验证了最佳资源比的存在性、合理性以及在部分用例下的具体值；第 7 节总结全文并提出下一步工作.

2　相关研究

提高能效就是指用更少的能源消耗来获得同等的服务，我们进行能效优化必须考虑性能和能耗之间的权衡. 国内外研究在这一方面持有两种不同的观点，文献[8-10]声称能耗和性能是两种不同的优化目标，所以他们之间存在着某种折中关系，提高性能必然导致高能耗，相反，降低能耗必然损失性能. 文献[11-12]进一步权衡了了电力消耗、SLA 需求和碳排放预算这三要素，通过 Lyapunov 优化技术设计了能耗感知的控制模型；文献[13]则声称能耗和性能的优化是一致的，最节能的系统往往也是性能最好的系统. 概念上，前者符合硬件设计思路，运算越快越耗电，后者符合软件设计思路，任务完成越快，计算机工作时间就越少，自然越省电. 这两种截然相反的观点的产生是因为采用了不同的假设和性能评价方法. 前者并没有考虑空闲能耗，也没有考虑“同一任务不同实现算法”的复杂度不同，性能评价粒度过细；后者则没有考虑计算机功率的动态性. 我们实验用计算机的有功功率在 50～100 瓦特之间浮动，某任务在 50 瓦下运行 2 秒和在 100 瓦下运行 1 秒消耗的能量相等，而性能则显然不同. 然而，在云环境下空闲能耗对任务能效的影响要大于节点有功功率的浮动，我们旨在通过“减少空闲能耗”来提高能效，即通过能耗优化方法来实现能效优化.

目前，新能源（如风能、太阳能）正得到广泛使用，其生成和预测的相关研究是数据中心能耗优化的重要手段；除此之外，跨数据中心的负载均衡的研究也使得能源得到充分利用[4]. 本文的研究旨在指导数据中心内部，即集群的能耗优化方法. 云环境下的能耗优化问题可以从硬件和软件两个角度考虑，硬件层面包括组件开关[5-6]，即观察集群中各个组件（如芯片）的工作负载，将处于空闲状态的组件关闭；以及如被广泛使用的 DVFS[14-15] 等性能调整技术，根据硬件的能耗情况动态地调节硬件性能，以得到低功耗的工作状态. 软件层面的能耗优化方法按其作用范围可以分为单机环境和集群环境，其中单机环境下有操作系统[16]和虚拟机能耗优化方法[17]. 我们主要从软件层面研究集群中的能耗优化技术，通过分析集群中任务的资源使用情况，寻找一种资源模型来指导资源分配和任务调度，降低空闲能耗，最终实现能效优化.

一些能耗优化方法不采用“以负载集中为目的”的任务调度，而是根据任务或任务队列特性，通过提高资源利用率，减少能源损失，进而优化能耗. 针对 MapReduce 框架：文献[18]设计了 MapReduce 调度过程的数学模型，分析影响数据本地化的系统配置值，设计了旨在减少网络传输的调度算法；文献[19]提出了一种考虑用户能耗需求的能耗可感知 MapReduce 应用模型；文献[20]基于遗传算法提出了旨在提高资源利用率的任务调度算法；文献[21]提出了一种动态槽机制，通过提高 CPU 资源的使用率来节约能耗并提高性能，但是并没有进行实验验证. 上述方法均通过提高资源利用率进行能效优化，这些方法主要考虑的是节点的某一种或几种特定资源的利用率，而本研究则着重考虑任务的多种资源使用的比率，与上述方法有本质不同.

对于一个任务，其最佳的资源占有率并非是最大资源占有率，而是一个适当的比例[22]，多种资源

占有率和性能之间的关系会是一个类似 U 型的曲面.本文则力求寻找这一最佳的资源占有率,并作为资源调度的目标.与该思路相关的现有研究包括:文献[23]按照任务的最佳资源请求、平均收益、失败率等排序,然后通过启发算法解决资源分配问题,其中最佳资源请求是预定义的,文中没有给出其定义方式;文献[24]将 Map/Reduce 任务继续划分为 CPU 密集型和 I/O 密集型阶段,采取称为"资源使用管道"的技术让细分后的不同任务阶段同时运行,从而提高了 CPU 和 I/O 的使用率,其本质上考虑了任务对不同资源的不同要求,但没有抽象出资源比的概念,而仅仅将任务划分成不同的资源密集类型;文献[25]提出了一种称为绿色调度的调度策略,通过适当地牺牲 MapReduce 公平调度算法的公平性来让资源使用情况互补的任务重叠运行,如增加 CPU 密集型任务和 IO 密集型任务同时执行的概率,但没有进行量化分析.本研究的最佳资源比模型则量化的定义的任务对资源的需求特征.

云环境中的资源分配是一项关键技术,大多数云平台(如 Hadoop MapReduce)任务调度算法的重心都放在调度器上①[26],主要考虑的是公平性和负载均衡,而较少的从任务资源需求的角度展开研究,这就可能导致任务分配后,某些节点出现过载,而其他节点却出现空闲状态的情况.因此,需要一个资源模型对任务所需的资源进行定义,进而提升集群的资源利用率.本文提出的最佳资源比模型则是从资源使用率的比值角度来表征一个任务的特点,并作为资源分配和任务调度的基础.

3 普适的资源和能效模型

本节首先定义了普适的资源模型和能效模型,然后在后续章节加以细化,本节中提及的任务是一种泛化的概念,但同样适用于 MapReduce 任务.一方面,资源和能效模型是针对任务的,而非针对集群或节点的,本研究着重研究任务占有的资源特征,而按照该特征的资源分配是进一步研究,因此我们研究一个任务的 n 种资源,以及该任务的能效.另一方面,任务占有的资源来源于某节点,等待能耗也是由节点产生的,若集群是由虚拟化技术构建的,每个虚拟节点拥有的资源为分配给该虚拟节点的物理资源或虚拟资源.由于虚拟节点的隔离性,绝大部分资源都互不影响,因此可以将虚拟节点当做物理节点处理.

定义 1. 计算资源(Computing Resource).计算机系统中的资源主要包括硬件资源、软件资源、网络资源和数据(内容)资源.其中,产生能耗的主要为硬件资源和网络资源,如中央处理器(CPU)、内存储器、外存储器、输入/输出设备、网卡等.计算资源为与这些硬件资源的抽象,如 CPU 资源、内存资源、磁盘资源、网络资源等.

定义 2. 资源量(Resource Quantity).资源量是对某任务分配或消费的节点计算资源大小的一种度量.设任务资源分配和消费均是动态的,随时间变化而变化.给定一个任务,n 为资源种类数,$r_i(t)$ 表示任务在 t 时刻消费第 i 个资源的资源量,$r'_i(t)$ 表示任务在 t 时刻分配的第 i 个资源的资源量,$|r_i|$ 表示第 i 个资源的资源总量.在 t 时刻,任务消费资源量 $R(t)$ 可用一个 n 维向量来表示,其每个分量是一个关于时间的函数.同理分配资源量 $R'(t)$ 和资源总量 $|R|$:

$$\begin{aligned} R(t)&=\langle r_1(t),r_2(t),\cdots,r_n(t)\rangle\\ R'(t)&=\langle r'_1(t),r'_2(t),\cdots,r'_n(t)\rangle\\ |R|&=\langle |r_1|,|r_2|,\cdots,|r_n|\rangle \end{aligned} \tag{1}$$

资源量需可度量、可计算,并统一不同种类资源的量纲.以消费资源量 $R(t)$ 为例,对于任意资源,需要确定其资源量的表示方法,如可以使用 CPU 频率、CPU 使用率、内存使用率、内存读写速率、硬盘 I/O 使用率、硬盘读写速度、网络吞吐量、带宽占用率等一项或多项指标分别来描述 CPU、内存、磁盘和网络的使用情况.资源量的定义要考虑节点的异构性,采用绝对值而非相对值,资源量的量纲需一致以计算资源比.下文求解中主要讨论 CPU、磁盘和网络 3 种资源.某一时刻 t 的磁盘资源量 $r_d(t)$,网络资源量 $r_n(t)$ 可以分别用该时刻的磁盘读写速率 $v_d(t)$,网络读写速率 $v_n(t)$ 来表示,两种资源量的量纲一致,都为 MB/s.若以 CPU 的使用率 $\omega(t)$ 作为 CPU 消费资源量 $r_c(t)$,则忽视了节点 CPU 的异构性,如 2.0 GHz CPU 的 50% 使用率下的运算能力相当于 1.0 GHz CPU 的 100% 使用率下的运算能力.而且该方法使 CPU 资源量的量纲与其他资源量不统一.根据我们前期的研究[27],本文用定义 3 中的 CPU 资源量 $r_c(t)$ 来描述 CPU 资源分配和消费情况,并将 CPU 资源量的量纲统一到了 MB/s,和网络资源量、磁盘资源量度量一致.

① Capacity Scheduler. http://hadoop.apache.org/docs/r1.0.4/capacity_ scheduler. html

定义 3.　CPU资源量(CPU Resource Quantity). 任务在 t 时刻 CPU 消费资源量 $r_c(t)$可以由该时刻 CPU 执行浮点运算的速率表示，同理分配资源量 $r'_c(t)$和资源总量$|r_c|$. 若用 MB 度量浮点运算数量(大小)，CPU 资源量单位为 MB/s. 设：

$$\begin{aligned} r_c(t)&=C\times f(t)\times\omega(t)\\ r'_c(t)&=C\times f'(t)\times\omega'(t)\\ C&=\frac{C_c\times C_{fn}\times C_{mw}\times C_{fz}}{2^{23}\times K\times P_c}\end{aligned} \tag{2}$$

其中，$f(t)$为 t 时刻的 CPU 频率，可以等价为 1 秒内时钟周期数目，单位 Hz；$\omega(t)$为 t 时刻的 CPU 使用率；C 为硬件相关的常量，是若干 CPU 参数的运算结果(见式(2))，其中 P_c为一个机器周期包括的时钟周期数；C_c为 CPU 核心数；C_{fn}为每机器周期浮点运算次数，C_{mw}为机器字长，单位比特；C_{fz}为浮点运算对象与机器字长的倍数关系，K 为数量级调节系数. CPU 资源量的定义仅考虑了 CPU 主要参数，虽然没有反映 CPU 所有参数特性，如缓存大小、总线速度、总线宽度等，但这些参数的对资源量定义的影响较小. 后文需要推导的最佳资源比并非是一个精确的比值，目前的 CPU 资源量定义可以满足最佳资源比的定义和后续的按资源比的资源分配研究.

定义 4.　吞吐量(Throughput). 吞吐量是对某任务在某时刻在给定资源量的条件下处理数据速度的一种度量. 任务在 t 时刻的吞吐量 $V(t)$是一个由多个分量组成的向量，吞吐量 $V(t)$在第 i 个资源上的分量记为 $v_i(t)$，是 t 时刻分配资源量 $r'_i(t)$下进行数据处理时达到数据处理速度，单位 MB/s.

$$V(t)=\langle v_1(t),v_2(t),\cdots,v_n(t)\rangle \tag{3}$$

根据资源量的定义可知：由于均采用 MB/s 作为单位，t 时刻，磁盘吞吐量等于磁盘消费资源量 $v_d(t)=r_d(t)$，网络吞吐量等于网络消费资源量 $v_n(t)=r_n(t)$. 唯独 $v_c(t)$和 $r_c(t)$无法对应，因为 CPU 处理数据的速度和算法处理数据的速度不同，还与算法复杂度有关，一条数据可能会在多个 CPU 周期反复处理.

定义 5.　CPU 吞吐量(CPU Throughput). 任务在 t 时刻的 CPU 吞吐量 $v_c(t)$是吞吐量 $V(t)$的 CPU 资源分量. CPU 资源具有特殊性，单位数据会被多个时钟周期处理，因此，$v_c(t)$需要从算法角度进行定义，设 t 为一个足够小的时间段，$\chi(t)$为一个比例，表示在 t 时间内从 CPU 处理的指令级数据量和 CPU 处理的算法级数据之间的比例. 则

$$v_c(t)=\chi(t)\cdot r_c(t) \tag{4}$$

$\chi(t)$与算法时间复杂度 $O(n)$相关. 本研究采用数据量单位 MB 来衡量 CPU 的运算能力，CPU 的运算能力是常量，即单位时间处理的指令级数据量大小为一定的，而单位时间处理的算法级数据量则与算法的复杂程度相关. 本研究采用一种一般化的单处理器、随机存取计算模型(RAM)来作为研究基础. 在 RAM 模型中，指令一条接一条地执行，没有并发操作. RAM 模型包含了真实计算机中常见的指令. 设 $D(t)$为任务在 t 时间内处理的算法级数据量，该值可以通过实验获得；$F(t)$为任务 t 时间内的算法复杂度，定义为处理一条数据所需的 CPU 指令(脉冲)个数，该值可通过对算法进行白盒分析获得，最简单的方式是通过实验，即在 CPU 使用率 $\omega(t)=1$ 时将处理一条数据的执行时间与 CPU 频率 $f(t)$相乘；$N(t)$为任务在 t 时间内处理的数据条数，可通过实验获得；C 为常量(见式(2)). 则

$$\chi(t)=\frac{D(t)}{F(t)\cdot N(t)\cdot C} \tag{5}$$

由式(4)和式(5)可得

$$v_c(t)=\chi(t)\cdot r_c(t)=\frac{D(t)\cdot r_c(t)}{F(t)\cdot N(t)\cdot C} \tag{6}$$

能效是单位时间内系统的运算量和耗电量的比值，节点能耗是节点资源消耗的能量，任务能效是任务自身的“运算量”比上任务“分配资源”的耗电量，采用分配资源量而非与消费资源量是因为一旦资源分配给某任务，即使没有被消费，也无法被别的任务使用. 任务能效的瞬时值中，运算量可以看做任务的吞吐量 $V(t)$，即处理数据的速度；耗电量则可以看作为节点功率 $P(t)$与资源占有率(%)之间的乘积.

定义 6.　任务能效(Energy Efficiency of Task). 任务在 t 时刻的能效 $\eta(t)$为任务吞吐量 $V(t)$与其占有资源 $R'(t)$消耗的能量 $E(t)$的比值：

$$\eta(t)=\frac{V(t)}{E(t)},\ E(t)\neq 0 \tag{7}$$

假设设备的功率稳定为 P：

$$E(t)=P\times\frac{R'(t)}{|R|} \tag{8}$$

$$\eta(t)=\frac{V(t)}{E(t)}=\frac{|R|\times V(t)}{P\times R'(t)}=\lambda\frac{V(t)}{R'(t)},\ \lambda=\frac{|R|}{P} \tag{9}$$

能效 $\eta(t)$的单位为 MB/Joule. 其中 λ 为常量，若考虑具体资源 $\forall i\in[1,n]$，结合式(6).

$$\eta_i(t)=\lambda\frac{V_i(t)}{R'_i(t)}=\lambda\times\chi(t)^{\gamma}\times\frac{r_i(t)}{r'_i(t)},\ \gamma=\begin{cases}1, & r_i=r_c\\ 0, & r_i\neq r_c\end{cases} \tag{10}$$

$r_i(t)/r'_i(t)$的最大值为1，当 $r_i(t)/r'_i(t)=1$ 时，$\eta_i(t)$达到最大值. 由此可见，若考虑计算机组件的功率为常量，则对于特定任务，能效仅与资源使用率(消费资源和分配资源的比值)有关. 某任务在时刻 t，若 $r_i(t)=r'_i(t)$，则该资源的利用率为100%，没有资源空闲. 若 $r_i(t)<r'_i(t)$，则资源 r_i 没有得到充分利用，资源量 $r'_i(t)-r_i(t)$ 处于空闲状态.

定义7. 空闲资源量(Idle Resource Quantity). 给定一个任务，设 n 为资源种类数，$\Delta r_i(t)$ 表示任务在 t 时刻空闲的第 i 个资源的资源量. 在 t 时刻，任务第 i 个资源的空闲资源量 $\Delta r_i(t)$ 和任务的总空闲资源量 $\Delta R(t)$ 为

$$\Delta r_i(t)=r'_i(t)-r_i(t),\ \Delta R(t)=\sum_{i=0}^{n}\Delta r_i(t) \tag{11}$$

$\forall i\in[1,n]$，任务在 T 时间段内的第 i 个资源的空闲资源量 Δr_i 以及总空闲资源量 ΔR 为

$$\Delta r_i=\int_0^T (r'_i(t)-r_i(t))\mathrm{d}t,\ \Delta R=\sum_{i=0}^{n}\Delta r_i \tag{12}$$

对于多个资源，当某一资源等待其他资源的运算结果时，该资源空闲，这种空闲并非真正意义上的空闲，而是一种被动空闲. 若任务分配的资源不合理，各个资源之间会产生等待，如CPU会因等待I/O操作而阻塞，产生空闲资源. 本研究针对因资源间等待产生的空闲资源，由定义6可知，空闲资源会降低任务的能效. 由于资源是被动空闲，因此对于任意时刻 t，$\exists i\in[1,n]$，$r_i(t)=r'_i(t)$，$\Delta r_i(t)=0$.

我们认为，对于某任务，当其分配资源量中的各类资源分量达到某一适当比例时，可以使得不同资源间没有因吞吐量不同而造成等待，进而没有因资源等待造成空闲资源，从而实现任务的能效优化. 实验一证明在MapReduce环境中，某任务运行时存在一个恰当资源比，使得不同资源间不存在因吞吐量或处理速度不同而造成等待，这个资源比定义为任务的最佳资源比. 例如，仅考虑CPU和网络这两种资源，若执行某任务时CPU处理数据的速度是网络接收该数据速度的两倍，该任务最佳资源比应为"1∶2"(CPU和网络资源的比值). 在多任务环境中，我们可以通过任务调度和资源分配来调整资源比以达到最佳，如该任务分发到某节点且该节点分配该任务CPU资源时，应该同时分配两倍的网络资源. 若某节点仅剩余较多CPU资源和极少网络资源，则该任务不适于分发给该节点. 可以认为，当节点运行一个I/O密集型任务时，CPU会较为空闲，这时可以调度一个CPU密集型任务给该节点. 本研究认为动态的资源分配和任务调度使任务的分配资源量满足最佳资源比，是一种有效的能效优化方法. 本文仅定义和推导最佳资源比.

定义8. 最佳资源比(Best Resource Ratio). 给定一个任务，其 t 时刻的资源比是各分配资源量之间的比值. 当有 n 种资源时，资源比表示形式如下：

$$r'_1(t):r'_2(t):\cdots:r'_n(t) \tag{13}$$

理想情况下，任务 t 时刻的最佳资源比 $\Omega(t)$ 是指按照该比值分配资源后，$\forall i\in[1,n]$，$r_i(t)=r'_i(t)$，$\Delta R(t)=0$，$\eta(t)$ 达到最大值. 因此，$\Omega(t)=r_1(t):r_2(t):\cdots:r_n(t)$. 特殊的，当某种资源可以忽略或不参与运算时，我们定义该资源的最佳资源比分量为 ∞^{-1}，∞^{-1} 表示无穷小. 例如，假设某任务仅仅需要CPU和本地磁盘，而不需要网络读写，其最佳资源比类似于 $m:n:\infty^{-1}$(CPU∶Disk∶Network).

4 MapReduce资源比模型

前一节定义了普适任务模型下的资源模型、能效模型和最佳资源比模型，本节将分析MapReduce的执行流程和资源消费特征，以将上述模型应用到MapReduce中. MapReduce作业(MapReduce Job)是客户需要执行的一个工作单元. Hadoop将MapReduce作业分成若干任务(Task)来执行，其中包括两类任务：Map任务和Reduce任务，简称Map/Reduce任务. Map/Reduce任务是调度的最小单元，但其执行过程复杂，因此将Map/Reduce任务作为研究其资源分配和消费特征的最小单位，粒度过大. 我们将Map/Reduce任务进行分解. 首先，从资源消费的角度定义操作(Operation)的概念，Map/Reduce任务可以分解为若干反复执行的操作，操作有固定的算法特征和资源消费特征. 其次，从资源分配的角度，若以操作作为资源分配的最小单元，则粒度过小，因此定义阶段的概念，一组连续的且资源消费特征相似的操作可以看作一个阶段. 综上所述，Map/Reduce任务的执行过程是阶段的集合，阶段由重复出现的操作组成. 例如，A，B，C，D 和 E 是5个操作，若将任务则视为一个序列"*AB-AB-CDE-CDE-AD*"，其中子序列"*AB*"反复出现了2次，"*CDE*"反复出现了2次，"*AD*"只出现了1次，阶段是该序列的一种划分，上述子序列均为一个阶段. 基于此，本节给出如下定义.

定义9. 操作(Operation). 任务中包含的重复出现的算法单元称为操作，任务是连续操作组成

的序列. 操作具有特定的资源消费特征，从资源消费角度，设某操作的执行时间为$[t_1,t_2]$，$\forall t_x,t_y\in[t_1,t_2],i\in[1,n],t_x\neq t_y$，则 $r_i(t_x)=r_i(t_y)$. 操作是消费资源的原子单元.

定义 10.　阶段(Phrase). 任务中一组连续的且资源消费特征相似的操作可以看作一个阶段，阶段中任意两个时刻分配资源量相同，消费资源量相似. 设某阶段的执行时间为$[t_1,t_2]$，$\forall t_x,t_y\in[t_1,t_2],i\in[1,n],t_x\neq t_y$，则 $r_i'(t_x)=r_i'(t_y),r_i(t_x)\approx r_i(t_y)$. 阶段是资源分配的原子单元.

由此定义 10 可知，阶段是资源分配的最小单元，且包含多个操作，操作是资源消费的最小单元. 我们将重新定义第 2 节中出现的分配资源量、消费资源量、空闲资源量以及吞吐量，使其与时间无关. 我们用阶段取代普适模型中的任务概念，用操作取代任务的 t 时刻. 具体定义见表 1.

表 1　普适定义与 MapReduce 定义的映射关系

定义序号	模型名称	普适对象	普适表达	MapReduce 对象	MapReduce 表达
定义 2	消费资源量	任务 t 时刻	$R(t)=\langle r_1(t),r_2(t),\cdots,r_n(t)\rangle$	阶段的第 k 个操作	$R^k=\langle r_1^k,r_2^k,\cdots,r_n^k\rangle$
定义 2	分配资源量	任务 t 时刻	$R'(t)=\langle r_1'(t),r_2'(t),\cdots,r_n'(t)\rangle$	阶段	$R'=\langle r_1',r_2',\cdots,r_n'\rangle$
定义 3	CPU 资源量	频率和使用率	$f(t),f'(t),\omega(t),\omega'(t)$	阶段及其第 k 个操作	f^k,f',ω^k,ω'
定义 4	吞吐量	任务 t 时刻	$V(t)=\langle v_1(t),v_2(t),\cdots,v_n(t)\rangle$	阶段的第 k 个操作	$V^k=\langle v_1^k,v_2^k,\cdots,v_n^k\rangle$
定义 5	CPU 吞吐量	任务 t 时刻	$v_c(t)=\chi(t)\cdot r_c(t)$	阶段的第 k 个操作	$v_c^k=\chi^k\cdot r_c^k$
定义 6	任务能效	任务 t 时刻	$\eta_i(t)=\lambda\times\chi(t)^\gamma\times\frac{r_i(t)}{r_i'(t)}$	阶段的第 k 个操作	$\eta_i^k=\lambda\times(\chi^k)^\gamma\times\frac{r_i^k}{r_i'}$
			$\Delta r_i(t)=r_i'(t)-r_i(t)$		$\Delta r_i^k=r_i'-r_i^k$
			$\Delta R(t)=\sum_{i=0}^{n}\Delta r_i(t)$		$\Delta R^k=\sum_{i=0}^{n}\Delta r_i^k$
定义 7	空闲资源量	任务 t 时刻	$\Delta r_i=\int_0^T(r_i'(t)-r_i(t))\mathrm{d}t$	阶段及其第 k 个操作(s 为阶段的操作个数)	$\Delta r_i=\sum_{k=0}^{s}(r_i'-r_i^k)$
			$\Delta R=\sum_{i=0}^{n}\Delta r_i$		$\Delta R=\sum_{i=0}^{n}\Delta r_i$

由表 1 可知，阶段资源量是对某阶段分配或消费的计算资源大小的一种度量. 给定一个阶段，设 n 为可分配的资源种类数，s 为该阶段包含的操作数量. r_i'表示第 i 个资源的分配资源量，r_i^k 表示第 k 个操作消费第 i 个资源的资源量. 阶段资源比则表示阶段各分配资源量之间的比值. 对于某阶段，包含 s 个操作，$\forall k\in[1,s]$，$\forall i\in[1,n]$，若 $r_i^k=r_i$，则该操作资源的利用率为 100%，没有资源空闲，若 $r_i^k<r_i$，则资源 r_i 没有得到充分利用，产生空闲资源. 由于资源是被动空闲，因此 $\forall k\in[1,s]$，$\exists i\in[1,n]$，$r_i^k=r_i$. 因此，我们重新定义 Map/Reduce 任务的最佳资源比.

定义 11.　Map/Reduce 任务最佳资源比(Best Resource Ratio of Map/Reduce Task). 设 Map/Reduce 任务由多个阶段构成，每阶段包含多个操作. 任务的最佳资源比是由其包含的阶段的最佳资源比组成的向量. 阶段的最佳资源比由其包含的操作的最佳资源比聚合而得. 操作的最佳资源比为该操作消费资源量的比值. 若每个任务的每个阶段分配的资源满足最佳资源比，则整个任务也满足最佳资源比.

阶段是资源分配单元，尚无法根据操作或任务分配资源，因此，需要将操作的最佳资源比合并. 设 Map/Reduce 任务包含 m 个阶段，第 j 个阶段($j\in[1,m]$)有 s_j 个操作组成. 后文仅研究 3 种资源的最佳资源比，即 CPU、磁盘和网络，且因为每个操作都有 CPU 资源的参与，CPU 是必须被分配的资源，因此，本研究简化最佳资源比为 $1:y:z$ 的形式，其中 y 表示磁盘资源，z 表示网络资源. 设 Ω 为任务的资源比，Ω_j 为任务第 j 个阶段的最佳资源比，Ω_j^k 是第 j 个阶段的第 k 个操作的最佳资源比.

$$\Omega=\langle\Omega_1,\Omega_2,\cdots,\Omega_m\rangle$$
$$\Omega_j=r_c:r_d:r_n=1:y:z=agg(\Omega_j^1,\Omega_j^2,\cdots,\Omega_j^{sj})\quad(14)$$
$$\Omega_j^k=r_c^k:r_d^k:r_n^k=1:y^k:z^k$$

聚合函数 $agg()$将在第 5 节定义，由式(14)可知，求解一个 Map/Reduce 任务的最佳资源比只需要确定其包含的每个操作的最佳资源比，本文第 5 节将给出操作最佳资源比推导过程，所用符号均针对第 k 个操作，因此为简化表述，省略上标 k.

5　最佳资源比推导

本节研究操作最佳资源比的确定方法，以及式(14)中的 $agg()$函数的数学表达. 由于操作是消费资源的最小单元，我们认为对于一个操作，在其执行时间内资源的消费特征是一致的. 操作中，CPU、

硬盘和网络均通过内存交换数据，当 3 种资源通过内存交换数据的速度(吞吐量)一致时，没有空闲资源.则操作 k 的使用资源量 r_i 之间的比值就是该操作的最佳资源比.从资源访问角度，操作是两种资源通过内存交换数据的过程；从吞吐量角度，操作是具有稳定资源吞吐量的执行时间片段，每个操作内不同时刻的资源吞吐量相等.总而言之，操作是资源消费特征的归纳.我们考虑操作的 CPU 资源 r_c、磁盘资源 r_d 和网络资源 r_n 这 3 种资源消费特征，将其划分为 4 种类型.

定义 12. 操作类型(Operation Type).操作由一种资源生产数据(数据生产者)，一种资源消费数据(数据消费者)构成，两者之间通过内存进行数据的交换和缓冲，我们称该数据处理过程为“生产者-消费者”模式.由于内存的存在，数据缓冲允许数据生产速度(吞吐量)与数据消费速度(吞吐量)不一致.我们将内存的使用情况作为划分操作的依据，在一个操作中，数据生产者生产的数据流入内存，数据消费者消费的数据从内存流出，两种资源吞吐量是稳定的，有特征的.对于任意操作，存在四种可能的“数据生产和消费”方式，我们将其定义为操作类型，分别为 cd，cn，dc 和 nc(见图 1).

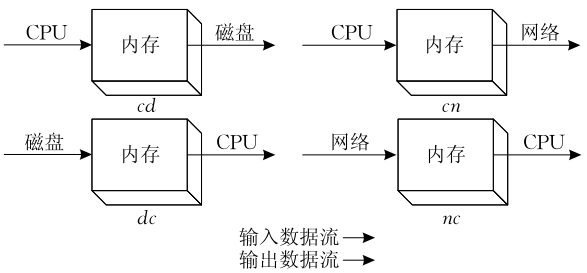


图 1 4 种操作类型

“生产者-消费者”模式满足通用计算系统的 IPO 模型(Input-Processing-Output Model)①，是一种抽象的数据处理模式，基于该模式的最佳资源比推导过程具有通用性，理由如下：首先，“生产者-消费者”模式在调用方式、内存缓冲区类型、角色数目和并行性等方面存在多种特征.例如，内存缓冲区可以具有多种数据结构，如队列、栈等；可以同时存在多个消费者和生产者；并且内存缓冲区的读写规则可以是同步或异步的.对这些特征进行组合，可以构建更加复杂的模式.由此可见，复杂模式均可以简化为“生产者-消费者”模式[28].一些经典的问题，如“哲学家就餐”和“读者-写者”问题，也都可以归纳、分解、转换或简化为“生产者-消费者”模式.其次，最佳资源比并非需要一个精确的比例，近似值即可表征任务的资源使用特性，因此将复杂运算模式简化为“生产者-消费者”模式是可行的.

操作类型和操作是“模式”和“实例”之间的关系.定义操作类型是为了更加方便的求解操作的最佳资源比.由定义 9 可知，操作 p 是 MapReduce 编程模型中最细粒度的资源消费单元，是两种资源通过内存交换数据的过程.本节所用符号均针对第 k 个操作，因此为简化表述，省略上标 k.某操作 p 可以用序偶来表示：

$$p=\langle \vec{r},\tilde{r}\rangle \tag{15}$$

$\vec{r}$ 为消费资源量，该资源是数据生产者，且生产的数据按 $\vec{v}$ 流入内存；$\tilde{r}$ 为消费资源量，该资源是数据消费者，且消费的数据按 $\tilde{v}$ 从内存流出，两个资源以内存为中介以各自稳定的吞吐量处理数据.操作内资源的吞吐量是稳定的.

定义 13. τ 函数(τ function).给定某阶段的第 k 个操作 $p=\langle \vec{r},\tilde{r}\rangle$，$\tau$ 函数返回该操作的执行时间.

$$t=\tau(p)=\tau(\vec{r},\tilde{r}) \tag{16}$$

定义 14. 操作内存(Memory of Phrase).假设每个操作都分配已知大小的内存，操作对该内存独占操作，各个操作内存读写不互相影响.设第 k 个操作内存容量为 ε，且已经保存任务相关的数据量为 ε_0，对于任意操作 $\varepsilon_0\leq\varepsilon$.

从操作类型可以看出，每种类型的操作都存在 CPU 资源，且存在网络资源或磁盘资源，内存必然为吞吐量大的资源充当数据缓冲(读缓冲或写缓冲)，假定第 k 个操作处理的数据量为 M，单位为 MB，操作执行时间为未缓冲的资源处理 M 的用时：

$$t=\frac{M}{\min(\vec{v},\tilde{v})} \tag{17}$$

M 可以测量得到，结合吞吐量和消费资源量之间的关系，τ 函数为

$$\tau(\vec{r},\tilde{r})=\frac{M}{\chi^{\gamma}\times\vec{r}^{\alpha}\times\tilde{r}^{\beta}}\begin{cases}\vec{v}<\tilde{v},\alpha=1,\beta=0,\gamma=\begin{cases}1, & \vec{r}=r_c\\ 0, & \vec{r}\neq r_c\end{cases}\\ \vec{v}\geq\tilde{v},\alpha=0,\beta=1,\gamma=\begin{cases}0, & \vec{r}=r_c\\ 1, & \vec{r}\neq r_c\end{cases}\end{cases} \tag{18}$$

求解操作的最佳资源比的基本思想是通过数据生产者和消费者的吞吐量之间的关系，若不存在内存缓冲，则数据生产者和消费者吞吐量应该相等；若考虑内存，可以将内存看作为一大小固定，容量有限的存储区，在内存缓冲区被放满时，生产数据的资源

① IPO Model. http://en.wikipedia.org/wiki/IPO_Model

(数据流入内存)被阻塞;在内存缓冲区为空时,消费数据的资源(数据从内存流出)被阻塞.

设操作 $p=\langle \vec{r},\vec{r}\rangle$. 数据生产(流入内存)和消费(流出内存)速度(吞吐量)分别为 $\langle \vec{v},\vec{v}\rangle$,执行时间为 t,则满足式(19)时,资源不会被阻塞,空闲资源为零:

$$t\times(\vec{v}-\Theta\times\vec{v})\in[-\varepsilon_0,\varepsilon] \tag{19}$$

数据可能会被多次消费,如图 1 中操作 dc 和 nc. 因此式(19)中设置 Θ 为操作 p 的读取系数.

若 $t\times(\vec{v}-\Theta\times\vec{v})>\varepsilon$,生产数据(流入内存)的资源将被阻塞,产生空闲资源. 同理,若 $t\times(\vec{v}-\Theta\times\vec{v})<-\varepsilon_0$,消费数据(流入内存)的资源将被阻塞,产生空闲资源. 按 τ 函数的定义和资源吞吐量和消费资源量之间的关系把式(18)中的吞吐量转换成资源量,得出

$$\frac{M\times(\chi^{\gamma}\times\vec{r}^{k}-\chi^{\gamma-1}\times\Theta\times\vec{r})}{\chi^{\gamma}\times\vec{r}^{\alpha}\times\vec{r}^{\beta}}\in[-\varepsilon_0,\varepsilon] \tag{20}$$

可以看出最佳资源比是多个可能、且与内存大小相关的值. 为使最佳资源比成为一个确定值,本研究令式(20)的左项等于 $\frac{\varepsilon+\varepsilon_0}{2}$. 为简化推导,考虑临界情况,即当内存作为数据输出端缓冲区时,其中充满等待处理的数据,当内存作为数据输入端缓冲区时,内存为空. 则 $\varepsilon_0=\varepsilon$. 得出

$$\frac{M\times(\chi^{\gamma}\times r^{k}-\chi^{\gamma-1}\times\Theta\times\vec{r})}{\chi^{\gamma}\times\vec{r}^{\alpha}\times\vec{r}^{\beta}}=\varepsilon$$

$$\begin{cases}\vec{v}<\vec{v},\alpha=1,\beta=0,\gamma=\begin{cases}1, & \vec{r}=r_c\\ 0, & \vec{r}\neq r_c\end{cases}\\ \vec{v}\geq\vec{v},\alpha=0,\beta=1,\gamma=\begin{cases}0, & \vec{r}=r_c\\ 1, & \vec{r}\neq r_c\end{cases}\end{cases} \tag{21}$$

式(21)是一个形如 $\frac{a(bx-cy)}{x}=d$(或 $\frac{a(bx-cy)}{y}=d$)的方程,其中 a,b,c,d 均为常量,x 和 y 为变量,因此可以求得 $\frac{x}{y}=\frac{ac}{ab-d}$(或 $\frac{x}{y}=\frac{ac+d}{ab}$). 因此:

当 $\alpha=1,\beta=0$ 时:

$$\frac{M\times(\chi^{\gamma}\times r^{k}-\chi^{\gamma-1}\times\Theta\times\vec{r})}{\chi^{\gamma}\times\vec{r}}=\varepsilon$$

$$\Rightarrow\frac{\vec{r}}{\vec{r}}=\frac{M\times\chi^{1-2\gamma}\times\Theta}{M+\varepsilon},\ \gamma=\begin{cases}1, & \vec{r}=r_c\\ 0, & \vec{r}\neq r_c\end{cases} \tag{22}$$

当 $\alpha=0,\beta=1$ 时:

$$\frac{M\times(\chi^{\gamma}\times r^{k}-\chi^{\gamma-1}\times\Theta\times\vec{r})}{\chi^{\gamma}\times\vec{r}}=\varepsilon$$

$$\Rightarrow\frac{\vec{r}}{\vec{r}}=\frac{\chi^{1-2\gamma}\times\Theta+\varepsilon}{M},\ \gamma=\begin{cases}0, & \vec{r}=r_c\\ 1, & \vec{r}\neq r_c\end{cases} \tag{23}$$

由式(22)和(23)可以求出任务每个操作的资源比,其中参数均可以通过实验确定.

最后,将阶段的每个操作资源比聚合为阶段资源比. 设阶段的资源比为 $1:y:z$. 其中 y 表示磁盘资源,z 表示网络资源. 阶段的各个操作资源比为 $1:y^1:z^1,1:y^2:z^2,\cdots,1:y^s:z^s$. 由阶段的 4 种类型可知,每个操作仅包含两种资源,且必然包含 CPU 资源,因此,对于任意操作,y^k 和 z^k 必有一项为零. 我们采用四分位数聚合每个阶段的所有操作的最佳资源比.

$$\begin{aligned}y&=upper\_quartile\ (y^1,y^2,\cdots,y^s)\\ z&=upper\_quartile\ (z^1,z^2,\cdots,z^s)\end{aligned} \tag{24}$$

若 $y=0$,则 y 为最小的非零 y^k 的值,同理 z. 采用上四分位数聚合而非均值是考虑 y^k 或 z^k 会有较多的零值,且式(21)是式(20)的临界条件,应该尽量给吞吐量小的一方多分配资源,在大部分情况下基于 MapReduce 的运算是一种数据密集型运算,I/O 相对于 CPU 均是稀缺资源.

6　实验验证

本节通过实验验证最佳资源比模型的存在性和合理性,并通过实验列出 Map/Reduce 任务的操作和各个阶段以及任务的最佳资源比,本研究在真实的 MapReduce 环境下进行,实验环境如表 2.

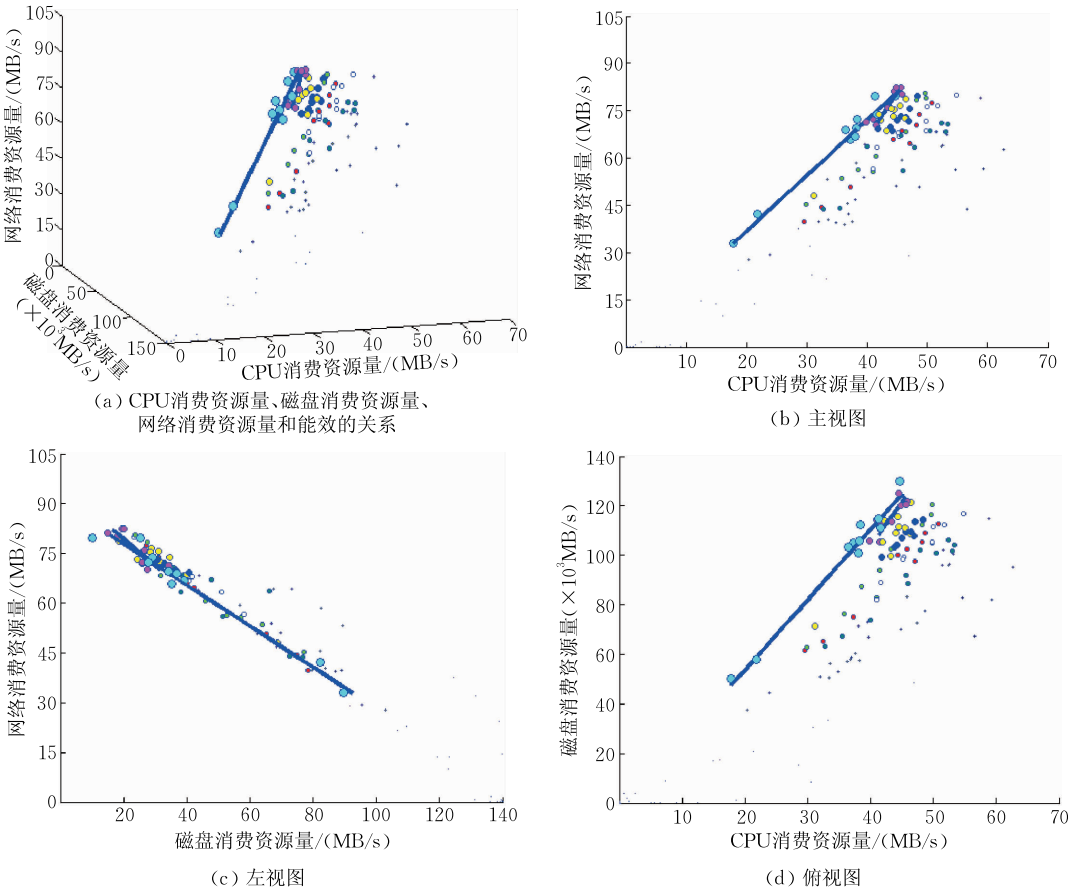
表 2　实验环境

项	描述
节点	1 个管理节点,11 个运算节点,同构的清华同方超翔 Z900PC,CPU Inter i5-2300 2.80GHz,8GB 内存,1TB 硬盘,主板集成声卡、显卡和网卡.
操作系统	CentOS 5.6, Linux 2.6.18 内核
MapReduce 版本	Hadoop 1.0.2
编程环境	JavaSE 6
功率测量方法	依据文献[29]
$f(t),\omega(t)$采集频率	1 秒采集 1 次数据($\Delta t=1$s)
测量单位	能效:兆比(MB)/焦耳(Joule); 功率:测量一律采用有功功率,单位瓦特(watt).
资源限制工具	*cpulimit* 限制 CPU 资源量,*cgroup* 限制磁盘和网络资源量.
MapReduce 用例	*MRBench*:交互式运算,即高并发地执行小作业,实验将 Hadoop *MRBench* 顺序启动多作业扩展为并发启动多作业,并发作业数为 10(10 线程总数据量与其他应用保持一致); *WordCount*:CPU 密集型计算,网络和磁盘 I/O 负载较轻; *Sort*:I/O 密集型计算, 上述 3 种用例分别采集了每节点为 250 MB,500 MB,750 MB,1000 MB,1250 MB,1500 MB,1750 MB 和 2000 MB 的数据量.

6.1 最佳资源比存在验证

本节通过实验数据证明“最佳资源比”的存在.在 Map/Reduce 任务执行时,某时刻任务将消费 CPU 资源量 $r_c(t)$、磁盘资源量 $r_d(t)$和网络资源量 $r_n(t)$,此时刻对应的能效为 $\eta(t)$.那么,在对于不同的任务,T 时间内会对应若干个$\langle r_c(t),r_d(t),r_n(t),\eta(t)\rangle$.我们将 t 时刻 $r_c(t)$、$r_d(t)$和 $r_n(t)$用三维空间的点表示,并用数据点的大小反映能效的大小,假如最佳资源比存在,在空间中就会存在一条直线,使得这条直线上的能效比其他能效高.我们不同负载对 *MRBench*、*WordCount* 和 *Sort* 的能效均做了上述空间散点图,发现能效极值点出现的位置具有一定的规律性,较高的能效点都在一个带状区域中,对于不同类型的计算,带状区域出现的位置也不同,初步证明能效与资源比相关.

为了找到最佳资源比,我们以 *WordCount* 为例进行了如下步骤的实验:令每个节点只同时执行一个任务,并通过工具限制任务使用的资源,使任务在各种分配资源量下执行.我们记录各节点上任务的实时能效(见定义 6),首先将 n 个实验数据点按照能效值所在区间等分成 k 类,每一类中所有的数据点的能效值都在同一个区间,且各个区间不相交;其次以 $r_c(t)$,$r_d(t)$,$r_n(t)$为坐标轴做出三维散点图(见图 2),用灰度和点的大小反映点所属的类别,点越大,能效越大,最后分别取能效值最大的两个分类中的点,用最小二乘法拟合出空间直线.



(a) CPU消费资源量、磁盘消费资源量、网络消费资源量和能效的关系

(b) 主视图

(c) 左视图

(d) 俯视图

图 2 CPU 资源量、磁盘资源量、网络资源量和能效的关系

如图 2 所示,在 $r_c(t)$,$r_d(t)$,$r_n(t)$构成的三维空间中可以发现,同一个能效分类中的点都处于一个带状区域,说明能效相同的点都在一条空间直线附近,而在一条空间直线上的所有点(x,y,z),都符合$\frac{x-x_0}{a}\approx\frac{y-y_0}{b}\approx\frac{z-z_0}{c}$,即符合$(x-x_0):(y-y_0):(z-z_0)\approx a:b:c(x_0,y_0,z_0,a,b,c$ 均为常数).由此可以推断,能效相同的点对应的资源比相同.以直线为参照,并从不同视角观察可知,能效值最大的点都在参照直线附近,可以认为该直线上的点具有相同的能效最大值,因此可推断最佳资源比的存在.在三维空间中,或许存在能效更高的点,但这样的点没有被采集到,可以认为,该能效点代表当

前测试用例在当前集群环境下不能达到的能效极大值.我们可以通过调整 CPU 的频率、改变磁盘速度或者改变网络带宽来达到能效更高的点,但这样做也改变了集群环境的特性.综上所述,我们认为,在给定的集群环境下,每一种任务都存在一个特定的资源比,使得在该资源比下能效值最大.

6.2　空闲资源耗能验证

本实验从另一个角度验证资源比的存在和其对能耗的影响,实验不采用能效而从能耗角度考虑空闲资源量.我们计算空闲资源的耗能情况,称为空闲能耗,空闲能耗的计算如式(25),其中 Δr_i.

$$E(T)=\int_0^T\left(\sum_{i=c,d,n}p_i\times\frac{r_i'(t)-r_i(t)}{r_i'(t)}\right)\mathrm{d}t \quad (25)$$

式(25)中,T 为任务的执行时间,p_c、p_d、p_n 分别为 CPU、硬盘、网卡的空载功率,在该实验中 $p_c=37\,\mathrm{W}$,$p_d=13\,\mathrm{W}$,$p_n=4\,\mathrm{W}$.$r_c'(t)$,$r_d'(t)$,$r_n'(t)$ 分别为时刻 t 分配的 CPU、磁盘、网络资源量,我们采用固定值代替瞬时值.实验环境中用的网络带宽为 1000 Mbps,理论最大峰值传输速率为 1000 MB/8s=125 MB/s,实验中测得的实际最大值为 95 MB/s,该值即为 $|r_n|$.节点硬盘接口类型为 Serial ATA 1.0a,数据传输率理论可达 150MB/s,实验中用 hdparm 测得磁盘的实际最大读写速度为 128 MB/s,该值即为 $|r_d|$.CPU 的最大使用率为 100%,最大支持频率 2.8 GHz,由式(2)计算可以得到,$K=10^{-3}$,$f=2.8\times10^9$ Hz,$\omega=1$,$P=12$,$C_c=4$,$C_{fn}=4$,$C_{mw}=64$ bit,$C_{fz}=2$,$|r_c|$ 近似计算为 64 MB/s.实验中将 CPU 频率固定为 2.8 GHz,则可用 CPU 使用率的分配代替 CPU 资源量的分配.通过给任务分配不同的 CPU、磁盘、网络资源量,观察任务的运行时间.本文选取如表 3 所示的 11 种资源分配策略进行研究.

表 3　不同实验组的资源限制

编号	CPU 资源量/(MB/s) $\lvert r_c\rvert=64$ MB/s	磁盘资源量/(MB/s)	网络资源量/(MB/s)
1	$0.25\times\lvert r_c\rvert$	3.75	2.26
2	$0.05\times\lvert r_c\rvert$	7.50	4.52
3	$0.1\times\lvert r_c\rvert$	15	9.50
4	$0.2\times\lvert r_c\rvert$	30	19
5	$0.4\times\lvert r_c\rvert$	60	38
6	$0.8\times\lvert r_c\rvert$	120	76
7	$0.4\times\lvert r_c\rvert$	120	76
8	$0.2\times\lvert r_c\rvert$	120	76
9	$0.1\times\lvert r_c\rvert$	120	76
10	$0.05\times\lvert r_c\rvert$	120	76
11	$0.25\times\lvert r_c\rvert$	120	76

本实验中用到的测试用例为修改后的 *WordCount*.为了增大任务的 CPU 使用率,在 Map 函数中,每读取完一条数据,做一次复杂度为 $O(n^3)$ 的运算(两个 n 阶矩阵的乘法).按照式(25)计算得到的实验结果如图 3 所示.

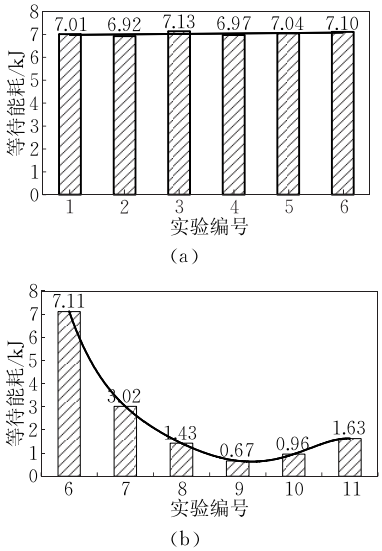


图 3　各组实验等待能耗对比

如图 3 所示,横轴为实验编号,纵轴为集群等待能耗,曲线为趋势线.从图 3(a)中可以看出,在资源比相同的情况下,空闲能耗并不随资源量的增大而变化,空闲能耗和资源量之间没有明显关系,而且在误差范围内,各组实验的空闲能耗相同.从图 3(b)中可以看出,当资源比发生变化时,空闲能耗也随之变化,而且并不是资源量越少,空闲能耗就越小.该任务平均消费 CPU 资源量在 $0.1\times|r_c|$ 左右.当 CPU 资源量多于 $0.1\times|r_c|$,CPU 因分配多过资源而空闲,当 CPU 资源量少于 $0.1\times|r_c|$,磁盘和网络资源因等待 CPU 而空闲.

6.3　MapReduce 阶段划分

本实验对 MapReduce 编程模型进行了操作和阶段的划分,Map/Reduce 任务的执行过程如图 4 所示.其中,虚线箭头表示节点内部的数据传输,实线箭头则表示节点之间的数据传输.

由前文可知,通过推导操作的最佳资源比即可知阶段的最佳资源比.通过对 Map/Reduce 任务的执行流程进行分析,可以得到一系列不同算法,不同类型的操作,每个操作在 Map/Reduce 任务执行流程中出现至少一次.Map 任务和 Reduce 任务各个操作的描述和所属类型见表 4.

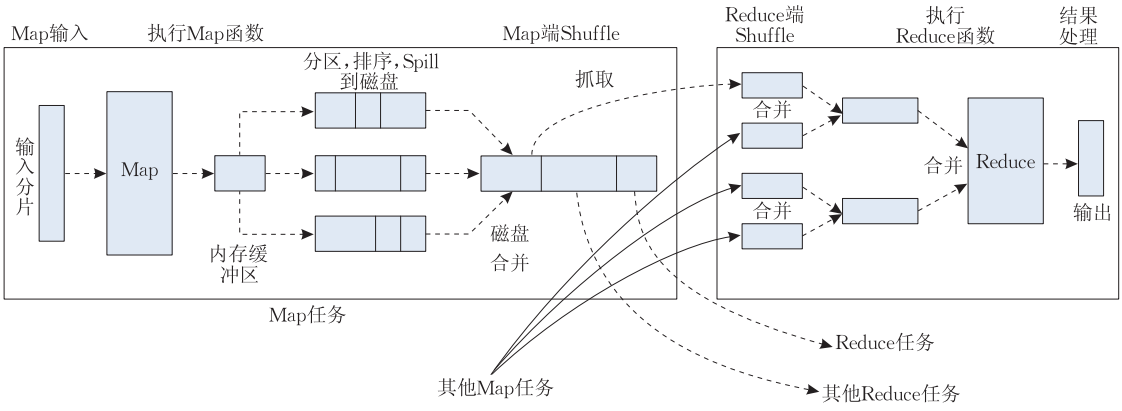


图 4 MapReduce 的执行过程

表 4 Map 和 Reduce 任务包含的操作

操作	名称	类型	操作描述
p^1	Map Remote Input	*nc*	Map 任务由网络获取 Map 函数远程输入,读到内存中
p^2	Key/Value Division	*cc*	将远程输入数据分解成键/值对
p^3	Map Local Input	*dc*	Map 函数读取本地输入数据,并将其分解成键/值对
p^4	Map Spill	*cd*	Map 函数处理数据的同时将生成结果写入到本地溢出写文件中
p^5	Map Merge-read	*dc*	在 Map 任务完成之前,溢出写文件被合并成一个已分区且以排序的输出文件
p^6	Map Merge-write	*cd*	合并之后将数据压缩后写入磁盘
p^7	Map Shuffle-read	*dc*	将生成数据读入 CPU
p^8	Map Shuffle-write	*cn*	将生成数据分别发送给对应 Reducer
p^9	Map Delete	*cd*	删除中间结果
p^{10}	Reduce Shuffle-read	*nc*	Reduce 端复制,将 Map 任务输出结果复制至本地
p^{11}	Reduce Shuffle-write	*cd*	Reduce 端复制,将 Map 任务输出结果复制至本地
p^{12}	Reduce-read	*dc*	从磁盘或内存读入数据并进行 Reduce 函数处理
p^{13}	Reduce-write	*cd*	Reduce 函数处理后将数据写入到磁盘中
p^{14}	Backup-read	*dc*	备份时将数据读入 CPU
p^{15}	Backup-write	*cn*	备份时将数据写入到网络位置

操作 p^2 的类型为 *cc*,即 CPU 先从内存中读取数据处理后再放回内存中,数据生产者和数据消费者都为 CPU 资源.该种操作类型只消费了 CPU 一种资源,因此 *cc* 类型阶段的最佳资源比定义为 $1:\infty^{-1}:\infty^{-1}$.其余操作类型的最佳资源比皆可通过第 5 节中提及的方法计算得到.

通过对表 4 中的操作进行聚合,可以得到 Map 任务和 Reduce 任务的阶段划分,每个阶段包含的操作和分配资源如表 5 所示.每个任务的各种资源(包括 CPU、网络、磁盘)的使用特征是相同且可分析的.多个阶段之间或会存在少量的并行执行,但是这种并行性较小,不影响每个阶段的最佳资源比计算.每个阶段在整个 Map/Reduce 任务执行过程中至多出现一次.除表 5 所列的阶段类型,通过实验分析发现,在 Map/Reduce 任务的执行过程中还有一些空闲阶段.在此阶段内 Map/Reduce 任务处于阻塞状态,对资源没有需求,空闲阶段的持续时间不定.

表 5 Map 和 Reduce 任务的阶段划分

阶段	名称	包含操作	分配资源
A	Map Remote Input	p^1,p^2	CPU,网络
B	Map Local Input	p^3	CPU,磁盘
C	Map Spill	p^4	CPU,磁盘
D	Map Merge	p^5,p^6	CPU,磁盘
E	Map Shuffle	p^7,p^8	CPU,磁盘,网络
F	Map Delete	p^9	CPU,磁盘
G	Reduce Shuffle	p^{10},p^{11}	CPU,磁盘,网络
H	Reduce	p^{12},p^{13}	CPU,磁盘
I	Reduce Save	p^{14},p^{15}	CPU,磁盘,网络

本研究对 *MRBench*,*WordCount* 和 *Sort* 的 $A\sim I$ 阶段的最佳资源比进行了分析,由于 *MRBench* 的执行时间很短,*WordCount* 的 Reduce 任务执行时间较短,*Sort* 的 Map 任务执行时间较短,不利于采集数据和计算最佳资源比.最终本研究成功计算了 *WordCount* 的 Map 任务的最佳资源比,以及 *Sort* 的 Reduce 任务最佳资源比.阶段的最佳资源比仅保留 1 位小数位,且用 ∞^{-1} 表示一个非常小的数,本实验列出 *Sort* 的 Reduce 任务各阶段的资源消费特征和最佳资源比,见表 6.

表 6　*Sort* 的 Reduce 任务各阶段最佳资源比

阶段	完成功能	分配资源	最佳资源比
等待	由于没有新的任务，节点处于空闲状态，但是会接收由 Master 节点发送来的数据. 此时的 CPU 是用来处理网络数据.	CPU 网络	$1:\infty^{-1}:1$
G	网络读取 Map 任务发送的中间结果，缓冲区存满后写入到本地磁盘中.	CPU 磁盘 网络	1:3.3:5.7
H	首先，从本地磁盘读取数据，然后对其进行处理，发送给存储节点.	CPU 磁盘 网络	1:4.2:9.6
I	将输出结果写入到本地作为第一份备份，其他备份写入到 HDFS 中.	CPU 磁盘 网络	1:15.2:8.4

7　结论和进一步工作

现有 MapReduce 集群环境中，不合理的资源分配会导致资源利用率降低，从而产生空闲资源和空闲能耗，最终导致能效低下. 本文从资源分配角度提高 MapReduce 能效，提出了“最佳资源比”模型，即对于一个任务，其各种资源的最佳分配比例并非是最大分配率，而是一个适当的比例，多种资源分配比例和能效之间的关系会是一个类 U 型曲面，分配率最大和最小都未必能实现能效最优，当分配资源满足最佳资源比时，资源均得到充分利用且不存在空闲，从而实现了能效优化. 基于此，本文进行了如下几方面的研究：

(1) 提出普适的资源和能效模型，该模型适用于任何运算任务，由于最佳资源比是一个比值，因此本研究将各种资源量统一量纲为 MB/s，并结合吞吐量的定义，从模型层面证明最佳资源比和能效之间的关系，量化空闲资源量和空闲能耗；

(2) 分析 MapReduce 编程模型，提出资源消费的最小单元“操作”和资源分配的最小单元“阶段”这两个概念；将普适的资源比模型变换到 MapReduce 下. 通过数据的“生产者-消费者”模式，求解 Map/Reduce 任务的最佳资源比；

(3) 实验分析典型 Map/Reduce 任务，从任务能效和空闲能耗两个角度证明了最佳资源比的存在，以及按资源比的资源分配对任务能效和空闲能耗的影响；

(4) 根据实验结果，对 MapReduce 执行过程进行划分，定义各个操作和阶段，并给出了部分 Map/Reduce 任务的最佳资源比，验证“任务-阶段-操作”模型的正确性. 同时，实验还发现了 Map/Reduce 任务执行过程中等待阶段的存在.

本文仅提出了最佳资源比模型，并论证了最佳资源比对能效的影响，但尚未基于最佳资源比为 Map/Reduce 任务进行资源分配，也未对 Map/Reduce 任务调度机制做出改进，因此下一步工作将提出一种基于最佳资源比的资源分配和任务调度算法以提高 Map/Reduce 集群能效. 此外，为了使本文中的理论能够得到实际应用，我们还需要简化最佳资源比的计算过程，如采用计算复杂度来替换公式中大量参数.

参　考　文　献

[1] Napper J, Bientinesi P. Can cloud computing reach the TOP500?//Proceedings of the Combined Workshops on UnConventional High Performance Computing Workshop Plus Memory Access Workshop. 2009: 17-20

[2] Milojicic D, Wolski R E. Delivering a private cloud. Computer, 2011, 44(4): 102-104

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

[4] Deng W, Liu F, Jin H, et al. Harnessing renewable energy in cloud datacenters: opportunities and challenges. IEEE Network, 2014, 28(1): 48-55

[5] Chen G, He W, Liu J, et al. Energy-aware server provisioning and load dispatching for connection-intensive internet services//Proceedings of the USENIX Symposium on Networked System Design and Implementation (NSDI). Boston, USA, 2008: 337-350

[6] Heath T, Diniz B, Carrera E V, et al. Energy conservation in heterogeneous server clusters//Proceedings of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Chicago, USA, 2005: 186-195

[7] Song Jie, Li Tian-Tian, Zhu Zhi-Liang, Bao Yu-Bin, Yu Ge. Benchmarking and analyzing the energy consumption of cloud data management system. Chinese Journal of Computers, 2013, 36(7): 1485-1499(in Chinese)
(宋杰，李甜甜，朱志良，鲍玉斌，于戈. 云数据管理系统能耗基准测试与分析. 计算机学报，2013，36(7)：1485-1499)

[8] Brown R. Report to congress on server and data center energy efficiency: Public law 109-431. Lawrence Berkeley National Laboratory, 2008

[9] Zhou Zhi, Liu Fang-Ming, Jin Hai, et al. On arbitrating the power-performance tradeoff in SaaS clouds//Proceedings of the International Conference on Computer Communications (INFOCOM). Turin, Italy, 2013: 872-880

[10] Harizopoulos S, Shah M, Meza J, et al. Energy efficiency: The new holy grail of data management systems research. arXiv preprint arXiv: 0909.1784, 2009

[11] Zhou Z, Liu F, Xu Y, et al. Carbon-aware load balancing for geo-distributed cloud services//Proceedings of the Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS). San Francisco, USA, 2013: 232-241

[12] Deng W, Liu F, Jin H, et al. Smartdpss: Cost-minimizing multi-source power supply for datacenters with arbitrary demand//Proceedings of the Distributed Computing Systems (ICDCS). Philadelphia, USA, 2013: 420-429

[13] Tsirogiannis D, Harizopoulos S, Shah M A. Analyzing the energy efficiency of a database server//Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. New York, USA, 2010: 231-242

[14] Hsu C, Feng W. A power-aware run-time system for high-performance computing//Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. Seattle, Washington, USA, 2005: 1

[15] Wang L, Von Laszewski G, Dayal J, et al. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS//Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid). Melbourne, Australia, 2010: 368-377

[16] Zeng H, Ellis C S, Lebeck A R. Experiences in managing energy with ecosystem. IEEE Pervasive Computing, 2005, 4(1): 62-68

[17] Stoess J, Lang C, Bellosa F. Energy management for hypervisor-based virtual machines//Proceedings of the USENIX Annual Technical Conference. Santa Clara, USA, 2007: 1-14

[18] Guo Z, Fox G, Zhou M. Investigation of data locality in MapReduce//Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012). Ottawa, Canada, 2012: 419-426

[19] Li Y, Zhang H, Kim K H. A power-aware scheduling of MapReduce applications in the cloud//Proceedings of the 2011 IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC). Sydney, Australia, 2011: 613-620

[20] Wang X, Wang Y, Zhu H. Energy-efficient task scheduling model based on MapReduce for cloud computing using genetic algorithm. Journal of Computers, 2012, 7(12): 2962-2970

[21] Yong M, Garegrat N, Mohan S. Towards a resource aware scheduler in hadoop//Proceedings of the 2009 IEEE International Conference on Web Services. Los Angeles, USA, 2009: 102-109

[22] Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing//Proceedings of the Workshop on Power Aware Computing Systems. San Diego, USA, 2008: 115-121

[23] Stillwell M, Schanzenbach D, Vivien F, et al. Resource allocation using virtual clusters//Proceedings of the International Symposium on Cluster Computing and the Grid, Shanghai. China, 2009: 260-267

[24] Chen Y P, Archana G. To compress or not to compress-Compute vs. IO tradeoffs for MapReduce energy efficiency//Proceedings of the ACM SIGCOMM Workshop on Green Networking. New Delhi, India, 2010: 23-28

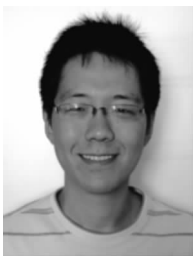
[25] Zhu Tao, Shu Chengchun, Yu Haiyan. Green scheduling: A scheduling policy for improving the energy efficiency of fair scheduler//Proceedings of the 2011 IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). Gwangju, South Korea, 2011: 319-326

[26] Zaharia M, Borthakur D, Sarma J S, et al. Job scheduling for multi-user MapReduce clusters. EECS Department, University of California, Berkeley: Technique Report UCB/EECS-2009-55, 2009

[27] Song Jie, Hou Hong-Ying, Wang Zhi, Zhu Zhi-Liang. Improved energy-efficiency measurement model for cloud computing. Journal of Zhejiang University (Engineering Science), 2013, 47(1): 44-52(in Chinese)
(宋杰, 侯泓颖, 王智, 朱志良. 云计算环境下改进的能效度量模型. 浙江大学学报(工学版), 2013, 47(1): 44-52)

[28] Silberschatz A, Galvin P B, Gagne G, et al. Operating system concepts. State of New Jersey, USA, Addison-Wesley, 1998

[29] Song Jie, Li Tian-Tian, Yan Zhen-Xing, et al. Energy-efficiency model and measuring approach for cloud computing. Journal of Software, 2012, 23(2): 200-214(in Chinese)
(宋杰, 李甜甜, 闫振兴等. 一种云计算环境下的能效模型和度量方法. 软件学报, 2012, 23(2): 200-214)



SONG Jie, born in 1980, Ph. D., associate professor. His research interests include cloud computing, massive data computing, and energy-efficient computing.

LIU Xue-Bing, born in 1989, M. S. candidate. His current research focuses on energy-efficient computing.

ZHU Zhi-Liang, born in 1962, Ph. D., professor, Ph. D. supervisor. His current research focuses on complex network and cloud computing.

LI Tian-Tian, born in 1989, Ph. D. candidate. Her current research focuses on energy-efficient computing.

ZHAO Da-Zhe, born in 1960, Ph. D., professor. Her research focuses on software engineering.

YU Ge, born in 1962, Ph. D., professor, Ph. D. supervisor. His research interest is database theory.

Background

Cloud computing (CC) is a new emerged and infrastructure shared computing method, in which, huge system pools are connected together to provide computational resources (such as data and software) on-demand via a computer network. Many factors contribute to the demand for this type of environment. However, the coming of CC era has also brought a serious problem of computer power consumption. CC is considered to be a green data management mode, but it does not provide mature solutions to evaluate and reduce energy consumption. Therefore, we still need an energy-efficient way to achieve real "green computing".

At present, there have been some studies on the optimization of energy efficiency, but these studies are not completely effective for CC because of its distinctive features (such as Shared-Nothing, migration of operations rather than migration of data, MapReduce, high availability and on-demand services). Therefore, it is necessary to introduce an energy efficiency optimization solution special for CC. This paper is a work of such topic; it focuses on the MapReduce cluster and applications in CC. We propose, derive and prove the best resource ratio of MapReduce task which could provide the guidance to energy efficiency optimization based on resource allocation and task scheduling.

This work is mainly supported by the National Natural Science Foundation of China (No. 61202088), named as "Research on Energy-Consumption Optimization Approaches for Cloud Database Systems". The project aims to study an optimization approach of "reducing idle power consumption by reducing node's waiting time" in cloud database system; this paper focuses on the "energy-consumption optimized data placement" part of the project. This work is also supported by Fundamental Research Funds for the Central Universities of China (N100704001). Our group has been working in the area of massive data computing and energy-efficient computing for four years and has published many papers.