DOI:10.7544/issn1000-1239.2016.20148443
53(9):2107-2131,2016

MapReduce 能耗建模及优化分析

廖 彬[1,4] 张 陶[2,3] 于 炯[2,4] 尹路通[4] 郭 刚[4] 国冰磊[2,4]
[1](新疆财经大学统计与信息学院 乌鲁木齐 830012)
[2](新疆大学信息科学与工程学院 乌鲁木齐 830046)
[3](新疆医科大学医学工程技术学院 乌鲁木齐 830011)
[4](新疆大学软件学院 乌鲁木齐 830008)
(liaobin665@163.com)

Energy Consumption Modeling and Optimization Analysis for MapReduce

Liao Bin[1,4], Zhang Tao[2,3], Yu Jiong[2,4], Yin Lutong[4], Guo Gang[4], and Guo Binglei[2,4]
[1](*School of Statistics and Information, Xinjiang University of Finance and Economics, Urumqi* 830012)
[2](*School of Information Science and Engineering, Xinjiang University, Urumqi* 830046)
[3](*College of Medical Engineering and Technology, Xinjiang Medical University, Urumqi* 830011)
[4](*School of Software, Xinjiang University, Urumqi* 830008)

Abstract The continuous expansion of the cloud computing centers scale and neglect of energy consumption factors exposed the problem of high energy consumption and low efficiency. To improve the MapReduce framework utilization of energy consumption, we build an energy consumption model for MapReduce framework. First, we propose a task energy consumption model which is based on CPU utilization estimation, energy consumption accumulation of main components and the average energy consumption estimation as well as the job energy consumption model of MapReduce. Specifically, after analyzing the energy optimization under energy consumption model, we come up with three directions to optimize energy consumption of MapReduce: optimize MapReduce energy consumption of job execution, reduce MapReduce energy consumption of task waiting and improve the energy utilization rate of MapReduce cluster. We further propose the data placement policy to decrease energy consumption of task waiting under heterogeneous environment and the minimum resource allocation algorithms to improve energy utilization rate of MapReduce jobs by the deadline constraints. A large number of experiments and data analysis of energy consumption demonstrate the effectiveness of energy consumption model and optimum policy of energy consumption.

Key words green computing; task scheduling; energy consumption modeling; energy analysis; data layout

摘 要 云计算中心规模的不断扩大以及设计时对能耗因素的忽略,使其日益暴露出高能耗低效率的问题.为提高 MapReduce 框架能耗利用率,首先对 MapReduce 任务进行了能耗建模,提出基于 CPU 利用率估算、主要部件能耗累加及平均功耗估算的任务能耗模型,并在此基础上建立了 MapReduce 作业能耗模型.其次,基于能耗模型对能耗优化进行了分析,提出从优化 MapReduce 作业执行能耗、减少 MapReduce 任务等待能耗与提高 MapReduce 集群能源利用效率 3 个方向对 MapReduce 进行能耗

收稿日期:2014-12-30;**修回日期**:2016-02-16
基金项目:国家自然科学基金项目(61562078,61262088,71261025);新疆财经大学博士科研启动基金项目(2015BS007)
This work was supported by the National Natural Science Foundation of China (61562078, 61262088, 71261025) and the PhD Research Startup Foundation of Xinjiang University of Finance and Economics (2015BS007).

优化.再次,提出异构环境下的数据放置策略减小 MapReduce 任务等待能耗,提出截止时间约束下的最小资源分配方法提高 MapReduce 作业能耗利用效率.通过大量的实验及能耗数据分析,验证了能耗模型及能耗优化方法的有效性.

关键词 绿色计算;任务调度;能耗建模;节能分析;数据布局

中图法分类号 TP393.09

数据的产生过程在经历被动和主动 2 种产生过程后,发展到了自动产生阶段,预示着大数据时代的来临.数据从简单的处理对象开始转变为一种基础性资源,如何更好地管理和利用大数据已经成为普遍关注的话题,大数据的规模效应给数据存储、管理以及数据分析带来了极大的挑战[1].据文献[2]统计,2007 年全球数据量达到 281 EB,而 2007—2011 年这 5 年时间内全球数据量增长了 10 倍.数据量的高速增长伴随而来的是存储与处理系统规模不断扩大,这使得运营成本不断提高,其成本不仅包括硬件、机房、冷却设备等固定成本,还包括 IT 设备与冷却设备的电能消耗等其他开销,并且,系统的高能耗将导致过量温室气体的排放并引发环境问题.事实上,在能源价格上涨、数据中心存储规模不断扩大的今天,高能耗已逐渐成为制约云计算与大数据快速发展的一个主要瓶颈[1].据文献[3]统计,目前 IT 领域的 CO_2 排放量占人类总排放量的 2%,而到 2020 年这一比例将翻番.2008 年路由器、交换机、服务器、冷却设备、数据中心等互联网设备总共消耗 8 680 亿 kW·h,占全球总耗电量的 5.3%.纽约时报与麦肯锡经过一年的联合调查,最终在《纽约时报》上发表了"Power, pollution and the Internet"[4],调查显示 Google 数据中心的平均功耗为 3 亿 W,Facebook 则达到了 0.6 亿 W,但巨大的能耗中却只有 6%~12%的能耗被用于处理相应用户的请求.与此同时,Barroso 等人在文献[5]中对 Google 内部 5 000 多台服务器进行长达半年的调查统计结果表明:服务器在大部分时间里利用率都在 10%~50%之间,服务器在负载很低(小于 10%)的情况下电能消耗也超过了峰值能耗的 50%.

Hadoop[6]作为新的分布式存储与计算架构,参考 Google 的分布式存储系统 GFS[7]实现了分布式文件存储 HDFS;参考 MapReduce[8]计算模型实现了自己的分布式计算框架;参考 BigTable 实现了分布式数据库 HBase.由于能够部署在通用平台上,并且具有可扩展性(scalable)、低成本(economical)、高效性(efficient)与可靠性(reliable)等优点,使其在分布式计算领域得到了广泛运用,并且已逐渐成为工业与学术届事实上的海量数据并行处理标准[9].虽然 Hadoop 拥有诸多优点,但是和 Google 服务器一样,Hadoop 集群内部服务器同样存在严重的高能耗低利用率问题[10].Hadoop 主要由分布式存储系统 HDFS 与分布式任务执行框架 MapReduce 两部分组成,现有研究大多从分布式存储系统 HDFS 入手解决 Hadoop 的能耗问题,针对 MapReduce 框架能耗优化方面的研究则相对较少.大量研究围绕通过对存储资源的有效调度,在不影响系统性能的前提条件下将部分存储节点调整到低能耗模式,以达到节能的目的.为提高 MapReduce 的能耗利用效率,本文做了如下工作:

1) 对 MapReduce 系统模型进行了分析,提出 3 种任务级的能耗模型:基于 CPU 利用率估算的能耗模型、基于主要部件能耗累加的能耗模型及基于平均功耗估算的能耗模型,并在这 3 种模型基础上得到 MapReduce 作业能耗模型.

2) 对 MapReduce 作业能耗模型优化分析的基础上,对优化 MapReduce 作业执行能耗、减少 MapReduce 任务等待能耗与提高 MapReduce 集群能源利用效率 3 个方面的能耗优化方法进行了讨论.

3) 提出异构环境下的数据放置策略以减少异构环境下的 MapReduce 任务等待能耗;推导出截止时间约束下的最小资源槽 *slot* 分配计算公式,计算适应当前作业的最小资源数量,提高 MapReduce 作业能源利用效率.

4) 大量的实验及能耗数据分析表明本文所提能耗模型及能耗优化方法的有效性.

1 相关工作

传统的 IT 系统一方面通过超额资源供给与冗余设计以保障 QoS 与系统可靠性[11],另一方面负载均衡算法专注于将用户请求平均分发给集群中的所有服务器以提高系统的可用性,这些设计原则都

没有考虑到系统的能耗因素,这使得 IT 系统的能量利用日益暴露出高能耗低效率的问题[12].学术与工业界分别从硬件[13-15]、操作系统[16-18]、虚拟机[19-26]、数据中心[27-33]4 个层次去解决 IT 系统的能耗问题.针对分布式计算系统能耗问题的研究,通常以 Hadoop 作为研究对象,并且大多从分布式存储系统 HDFS 入手解决其存在的能耗问题,针对任务执行框架 MapReduce 能耗优化方面的研究则相对较少.

研究分布式存储系统 HDFS 节能方面,根据软硬件角度进行划分,可分为硬件节能与软件节能 2 个方面[34].硬件节能主要通过低能耗高效率的硬件设备或体系结构,对现有的高能耗存储设备进行替换,从而达到节能的目的.硬件节能方法效果立竿见影,且不需要复杂的能耗管理组件;但是对于已经部署的大规模应用系统,大批量的硬件替换面临成本过高的问题.软件节能通过对存储资源的有效调度,在不影响系统性能的前提条件下将部分存储节点调整到低能耗模式,以达到节能的目的.由于不需要对现有硬件体系进行改变,软件节能是目前云存储节能技术的研究热点.软件节能研究主要集中在基于节点管理与数据管理 2 方面.节点管理主要研究如何选择存储系统中的部分节点或磁盘为上层应用提供数据服务,并让其他节点进入低能耗模式以达到降低能耗的目的.节点管理中被关闭节点的选择与数据管理技术紧密相关,而目前已有的数据管理技术主要有基于静态数据放置、动态数据放置与缓存预取 3 种.其中,基于静态数据放置的数据管理[35-39]根据固定的数据放置策略将数据存储到系统中各节点上,之后将不再改变其存储结构;基于动态数据放置的数据管理[40-43]根据数据访问频度动态调整数据存放的位置,将访问频度高与频度低的数据迁移到不同磁盘上,对存储低频度数据的磁盘进行节能处理以降低系统能耗;基于缓存预取的数据管理[44]借鉴内存中的数据缓存思想,将磁盘中的数据取到内存或其他低能耗辅助存储设备并使原磁盘进入低能耗模式以此达到节能的目的.Chen 等人[45]针对实际系统的研究发现数据访问具有很强的周期性,通过关闭空闲时段大量节点以达到节能的目的.文献[46]提出了 covering subset 的概念,将数据块与其副本中的至少一个数据块放入该子集中以保证所有数据块的可访问性的前提下,通过关闭与该数据块子集无交集的 DataNode 节点以达到节能的目的.文献[47-48]通过对 Yahoo 公司 HDFS 集群内部数据块访问规律的研究发现数据块的访问具有较强的规律性,根据数据块的访问规律将其放在 Cold-Zone 与 Hot-Zone 两个 DataNode 节点区域中,通过将 Cold-Zone 中 DataNode 节点进行节能处理从而达到整个集群节能的目的.

在分布式任务执行框架 MapReduce 能耗优化方面,少有的研究通过选择部分节点执行任务[46]、任务完成后关闭节点[49]、配置参数优化[45]、DVFS 调度[50]、作业调度[51]、虚拟机放置策略[52]及数据压缩[53]等方法达到提高 MapReduce 能耗利用率的目的.与 covering subset 思想相反,文献[49]提出 All-In Strategy(AIS)策略,即将整个 MapReduce 集群作为整体用于任务的执行,当任务结束后将整个集群做节能处理(关闭节点)达到节能的目的.Leverich 等人[46]发现 MapReduce 框架的参数配置对 MapReduce 能耗的利用具有较大影响,通过大量的实验得到优化 MapReduce 能耗的配置参数,对提高 MapReduce 集群系统的能耗利用率具有指导意义.文献[50]中利用 DVFS(dynamic voltage and frequency scaling)技术,通过动态地调整 CPU 频率以适应当前的 MapReduce 任务负载状态达到优化能耗利用的目的.文献[51]提出 Hadoop 节能适应性框架 GreenHadoop,通过合理的作业调度,在满足作业截止时间约束的前提下通过配置与当前作业量相匹配的作业处理能力(活动节点数量),达到最小化 Hadoop 集群能耗的目的,实验证明 GreenHadoop 与 Hadoop 相比提高了 MapReduce 的能耗利用率.宋杰等人[54]对云数据管理系统(包括基于 MapReduce 的系统)的能耗进行了基准测试,并定义了能耗的度量模型与能耗测试方法,证明了不同系统在能耗方面存在着较大差异,需要进一步对系统进行能耗优化.

本文与已有工作的不同包括:

1) 已有工作缺少对 MapReduce 能耗建模的研究.本文提出 3 种任务级的能耗模型:基于 CPU 利用率估算的能耗模型、基于主要部件能耗累加的能耗模型及基于平均功耗估算的能耗模型,并在这 3 种模型基础上得到 MapReduce 作业能耗模型.能耗模型是研究 MapReduce 能耗优化的理论基础.

2) 已有工作往往采用单一的方法优化 MapReduce 能耗.本文在 MapReduce 作业能耗模型的基础上,归纳了从 2 种不同的层面(作业的执行能耗与集群能源效率)、3 种不同的方向(优化 MapReduce

作业执行能耗、减少 MapReduce 任务等待能耗与提高 MapReduce 集群能源利用效率)优化 MapReduce 能耗.

3) 提出 MapReduce 任务等待能耗的概念,并提出异构环境下数据的放置策略减小 MapReduce 任务等待能耗.推导截止时间约束下的最小资源槽 *slot* 分配计算公式,计算适应当前作业的最小资源数量,提高 MapReduce 作业能源利用效率.

2 相关模型与定义

2.1 MapReduce 系统模型

MapReduce 运行环境通常由多个机架 RACK 组成,而一个 RACK 内部又由多个节点服务器组成.通常情况下,MapReduce 集群由 2 个 NameNode 管理节点(主管理节点与从管理节点)与多个 DataNode 节点构成.实际应用环境中,考虑到 DataNode 节点数量远远大于 NameNode 节点,DataNode 节点能耗之和占系统总能耗的 95%以上,并且 NameNode 负责整个系统的管理与调度工作,对 NameNode 进行节能处理将对系统的稳定性造成影响.所以本文的能耗模型主要针对 DataNode 节点、不对 NameNode 节点进行能耗建模.

定义 1. MapReduce 集群 DataNode 节点模型.将 MapReduce 集群中 n 个 DataNode 节点用集合 DN 表示:

$$DN=\{dn_0,dn_1,\cdots,dn_{n-1}\},\tag{1}$$

其中,$dn_i\in DN(0\leqslant i\leqslant n-1)$表示 MapReduce 集群中的 DataNode 节点.

定义 2. 作业的任务分解模型.如图 1 所示,MapReduce 框架将作业 *job* 分解为多个 Map 与 Reduce 任务并行地在集群中执行,可将这个过程定义为 $f(job)\mapsto M\cup R$,其中 f 为映射函数,集合 M 与 R 分别表示 *job* 分解后的 Map 与 Reduce 任务,其映射模型可由式(2)表示:

$$\begin{cases}f(job)\mapsto M\cup R,\\ M=\{mt_0,mt_1,\cdots,mt_{m-1}\},\\ R=\{rt_0,rt_1,\cdots,rt_{r-1}\}.\end{cases}\tag{2}$$

式(2)表示作业 *job* 被分解为 m 个 Map 任务与 r 个 Reduce 任务.其中,$mt_i\in M(0\leqslant i\leqslant m-1)$表示任意 Map 任务,$rt_i\in R(0\leqslant i\leqslant r-1)$表示任意 Reduce 任务.

定义 3. 任务资源映射模型.作业 *job* 被分解为 Map 与 Reduce 任务后,将由作业调度系统将任务映射到具有空闲资源槽的 DataNode 节点 dn_i 上执行.任务与资源映射过程可定义为映射 $f(M\cup R)\mapsto DN$,其中 f 为映射函数,M 与 R 分别表示 Map 与 Reduce 任务的集合,DN 表示 MapReduce 集群中 DataNode 节点的集合.具体而言,映射模型可由式(3)具体描述:

$$\begin{cases}f(M\cup R)\mapsto DN,\\ M=\{mt_0,mt_1,\cdots,mt_{m-1}\},\\ R=\{rt_0,rt_1,\cdots,rt_{r-1}\},\\ DN=\{dn_0,dn_1,\cdots,dn_{n-1}\}.\end{cases}\tag{3}$$

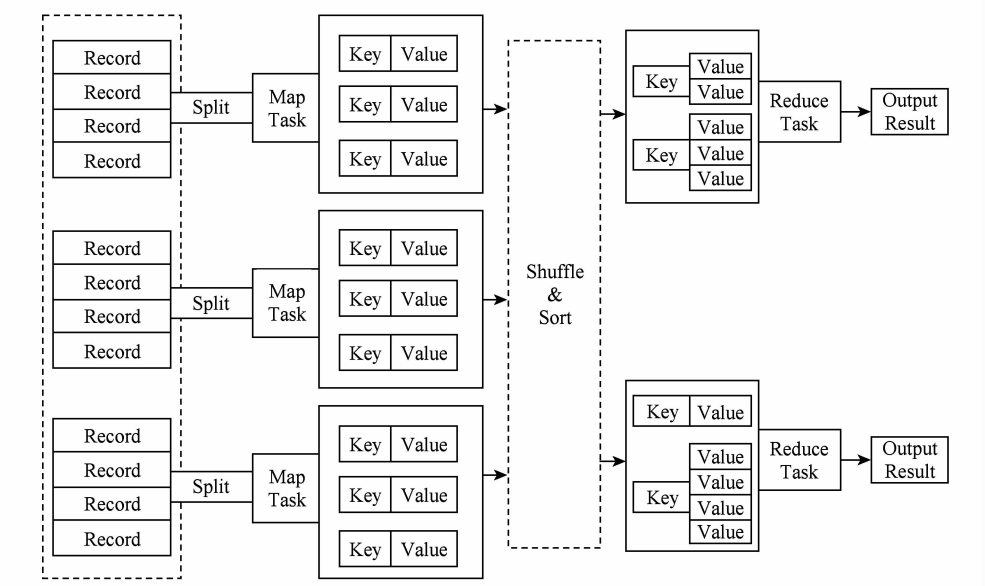


Fig. 1 MapReduce job decomposed into tasks.

图 1 MapReduce 作业的任务分解

2.2　MapReduce 任务能耗模型

设某个任务 $task\in M\cup R$ 被映射到具有空闲资源槽的 DataNode 节点 dn_i 上执行.任务 $task$ 从时间点 t_s 开始,运行到时间点 t_e 结束,即作业运行时间区间为$[t_s,t_e]$.那么,运行任务 $task$ 所需要的能耗 E_{task} 可由式(4)得到:

$$E_{task}=\int_{t_s}^{t_e}P_{dn_i}(u_i(t))\mathrm{d}t, \tag{4}$$

其中,$u_i(t)$表示在时刻 $t\in[t_s,t_e]$节点 dn_i 的系统利用率(也可分别表示 CPU、内存、磁盘、网卡等部件的利用率),并且存在 $u_i(t)\in[0,1]$;$P_{dn_i}(u_i(t))$则表示在时刻 t 系统利用率为 $u_i(t)$条件下 dn_i 节点的功耗.

节点 dn_i 的电能消耗是由 CPU、内存、磁盘、网卡等设备的电能消耗共同组成的[55].节点瞬时功率值为这些硬件部件的瞬时功率之和.由于 CPU、内存、磁盘、网卡等部件的电器特性之间存在着很大的差异,导致不同部件能耗计算方法不同.使用累加所有部件能耗(或瞬时功耗)来计算节点的能耗值的方法较为复杂.所以,考虑到能耗模型的简便性与完备性,本文提出 3 种能耗模型:第 1 种能耗模型考虑到系统整体能耗通常与 CPU 利用率成正比的特性,提出基于 CPU 利用率估算的能耗模型;第 2 种对系统主要的部件能耗模型进行分别建模,从而累加得到系统的整体能耗,该模型是一种完备的能耗模型,但具有较高的复杂度;第 3 种基于平均功耗估算的能耗模型是基于历史数据分析的能耗估算模型,主要用于作业运行前的执行能耗预测.

2.2.1　基于 CPU 利用率估算的能耗模型

由于 DataNode 节点功耗由静态功耗与动态功耗 2 部分组成[56],$P_{dn_i}(u_i(t))$可进一步细化为 $P_{dn_i}^{\mathrm{idle}}(u_i(t))$与 $P_{dn_i}^{\mathrm{dynamic}}(u_i(t))$两部分,即:

$$P_{dn_i}(u_i(t))=P_{dn_i}^{\mathrm{idle}}(u_i(t))+P_{dn_i}^{\mathrm{dynamic}}(u_i(t)). \tag{5}$$

CPU 是最主要的能耗部件,系统整体能耗通常与 CPU 利用率成正比[57].同时,随着节能技术在处理器应用上的不断推广,比如 Intel 的 Speedstep 与 AMD 的 PowerNow 技术,使得处理器能够根据负载动态调节性能,从而使得能耗与负载之间具有较好的比例性[55].那么,可将 $P_{dn_i}^{\mathrm{idle}}(u_i(t))$与 $P_{dn_i}^{\mathrm{dynamic}}(u_i(t))$分别用式(6)进行表示:

$$\begin{cases}P_{dn_i}^{\mathrm{idle}}(u_i(t))=aP_{dn_i}^{\max},\ a\in[0,1],\\ P_{dn_i}^{\mathrm{dynamic}}(u_i(t))=(1-a)P_{dn_i}^{\max}u_i(t),\\ \quad u_i(t)\in[0,1],\end{cases} \tag{6}$$

其中,$P_{dn_i}^{\max}$为 DataNode 节点 dn_i 满负荷(CPU 利用率 100%)时系统的功耗;$a\in[0,1]$为调节 $P_{dn_i}^{\mathrm{idle}}(u_i(t))$与 $P_{dn_i}^{\mathrm{dynamic}}(u_i(t))$之间比例的参数,其值由具体的计算机硬件体系结构决定.

由式(4)~(6)可以得到:

$$E_{task}=\int_{t_s}^{t_e}(aP_{dn_i}^{\max}+(1-a)P_{dn_i}^{\max}u_i(t))\mathrm{d}t. \tag{7}$$

定理 1. 据微积分的思想,将时间区间$[t_s,t_e]$等分为 k 份时($k\to+\infty$),即将$[t_s,t_e]$等分为$[t_0,t_1,\cdots,t_{k-1}]$,且 $\forall[t_j,t_{j+1}]$,$j\in[0,1,\cdots,k-2]$,满足 $u_i(t\in[t_j,t_{j+1}])\equiv u_{ij}$,其中 u_{ij} 为在时间区间$[t_j,t_{j+1}]$内不随时间变化的常量,并且同样存在 $u_{ij}\in[0,1]$.那么可将在节点 dn_i 运行任务 $task_j$ 所需要的能耗进一步表达为式(8):

$$\begin{cases}E_{task}=aP_{dn_i}^{\max}(t_e-t_s)+(1-a)\times\\ \quad P_{dn_i}^{\max}\sum_{j=0}^{k-1}(u_{ij}(t_{j+1}-t_j)),\\ t_{j+1}-t_j=\dfrac{t_e-t_s}{k},j\in[0,1,\cdots,k-2],\\ k\to+\infty.\end{cases} \tag{8}$$

证明. 由于运行任务 $task$ 所需要的能耗 E_{task} 可由式(7)进行计算,即:

$$E_{task}=\int_{t_s}^{t_e}(aP_{dn_i}^{\max}+(1-a)P_{dn_i}^{\max}u_i(t))\mathrm{d}t=$$
$$\int_{t_s}^{t_e}aP_{dn_i}^{\max}\mathrm{d}t+\int_{t_s}^{t_e}(1-a)P_{dn_i}^{\max}u_i(t)\mathrm{d}t=$$
$$aP_{dn_i}^{\max}(t_e-t_s)+\int_{t_s}^{t_e}(1-a)P_{dn_i}^{\max}u_i(t)\mathrm{d}t.$$

当将时间区间$[t_s,t_e]$等分为 $k(k\to+\infty)$份时,即将$[t_s,t_e]$等分为$[t_0,t_1,\cdots,t_{k-1}]$,且 $\forall[t_j,t_{j+1}]$,$j\in[0,1,\cdots,k-2]$,满足 $u_i(t\in[t_j,t_{j+1}])\equiv u_{ij}$,其中 u_{ij} 为在时间区间$[t_j,t_{j+1}]$内不随时间变化的常量,且存在 $u_{ij}\in[0,1]$,那么:

$$E_{task}=aP_{dn_i}^{\max}(t_e-t_s)+\int_{t_s}^{t_e}(1-a)P_{dn_i}^{\max}u_i(t)\mathrm{d}t=$$
$$aP_{dn_i}^{\max}(t_e-t_s)+\int_{t_0}^{t_1}(1-a)P_{dn_i}^{\max}u_{i_1}\mathrm{d}t+$$
$$\int_{t_1}^{t_2}(1-a)P_{dn_i}^{\max}u_{i_2}\mathrm{d}t+\cdots+\int_{t_{k-2}}^{t_{k-1}}(1-a)P_{dn_i}^{\max}u_{i_{k-1}}\mathrm{d}t=$$
$$aP_{dn_i}^{\max}(t_e-t_s)+(1-a)P_{dn_i}^{\max}u_{i_1}(t_1-t_0)+$$
$$(1-a)P_{dn_i}^{\max}u_{i_2}(t_2-t_1)+\cdots+$$
$$(1-a)P_{dn_i}^{\max}u_{i_{k-1}}(t_{k-1}-t_{k-2})=$$
$$aP_{dn_i}^{\max}(t_e-t_s)+(1-a)P_{dn_i}^{\max}\sum_{j=0}^{k-1}u_{ij}(t_{j+1}-t_j).$$

又因为将时间区间等分为了 k 份,可以得到等分区间的大小 $t_{j+1}-t_j$ 为

$$t_{j+1}-t_j=\frac{t_e-t_s}{k},$$

并且存在 $j\in[0,1,\cdots,k-2]$且 $k\to+\infty$.　　证毕.

利用基于 CPU 利用率估算能耗模型对 MapReduce 作业进行能耗计算的唯一前提条件是取得所有任务运行过程中的 CPU 利用率变化序列. CPU 利用率变化序列可通过资源利用监控软件取得,或对现有的 MapReduce 框架进行扩展,在作业运行报告中添加任务执行节点的 CPU 负载信息. 基于 CPU 利用率估算的能耗模型意义在于作业运行完毕后对作业能耗的计算,是将来开发 Hadoop 能耗监控及优化软件的理论基础.

2.2.2 基于主要部件能耗累加的能耗模型

由于节点的电能消耗是由 CPU、内存、磁盘、主板、风扇、网卡等设备的电能消耗共同组成的,本节首先建立各计算机主要部件的能耗模型,继而推导出基于主要部件能耗累加的能耗模型(本节主要考虑 CPU、内存、磁盘、网卡的能耗模型,其他部件如主板、风扇等可根据其能耗模型进行扩展).

CPU 的能耗模型受到处理器的活动状态、指令执行情况、高速 cache 使用情况、当前执行频率及制造工艺等因素的影响. 设 P_{CPU} 表示 CPU 利用率为 u_{CPU} 时的功率,根据 Kansal 等人在文献[58]中提出的能耗模型,P_{CPU} 与 u_{CPU} 的关系可表示如下:

$$P_{\mathrm{CPU}}=a_{\mathrm{CPU}}u_{\mathrm{CPU}}+\lambda_{\mathrm{CPU}}, \tag{9}$$

其中,a_{CPU} 与 λ_{CPU} 为 CPU 能耗模型的常数,不同的 CPU 之间存在着差异,其值可通过大量的训练获得.

已有大量工作针对内存的能耗模型进行了研究,发现影响内存能耗的主要因素是内存读写数据的吞吐量[59]. 记录内存最后一层 Cache(last level cache, LLA)的缺失次数,是一种轻量级的内存吞吐量评估方法. 根据吞吐量指标,设 $E_{\mathrm{mem}}(T)$ 表示内存在时间区间 T 内的能耗,N 表示最后一层 Cache 的缺失次数,内存模型表达如下[58]:

$$E_{\mathrm{mem}}(T)=a_{\mathrm{mem}}\times N+\lambda_{\mathrm{mem}}, \tag{10}$$

其中,a_{mem} 与 λ_{mem} 为内存能耗模型的常数.

磁盘能耗与读写数据量密切相关,设 $E_{\mathrm{disk}}(T)$ 表示磁盘在时间区间 T 内的能耗,r 与 w 分别表示在时间区间 T 内磁盘的读写数据量,磁盘能耗模型可表达如下[60]:

$$E_{\mathrm{disk}}(T)=a_{\mathrm{read}}\times r+a_{\mathrm{write}}\times w+\lambda_{\mathrm{disk}}, \tag{11}$$

其中,a_{read} 与 a_{write} 分别表示磁盘读写参数,λ_{disk} 表示磁盘的静态能耗,3 个参数的值可通过大量的能耗数据分析与训练获得.

与磁盘能耗模型类似,网卡的能耗与发送、接收到的数据量成正比. 设 $E_{\mathrm{net}}(T)$ 表示网卡在时间区间 T 内的能耗,s 与 v 分别表示在 T 时间区间内网卡发送与接收到的数据量,网卡能耗模型可表达如下:

$$E_{\mathrm{net}}(T)=a_{\mathrm{send}}\times s+a_{\mathrm{receive}}\times v+\lambda_{\mathrm{net}}, \tag{12}$$

其中,a_{send} 与 a_{receive} 分别表示网卡在发送、接收数据时的能耗参数,λ_{net} 为网卡的静态能耗参数.

节点的电能消耗是由 CPU、内存、磁盘、网卡等设备的电能消耗的总和. 当任务 $task\in M\cup R$ 在节点 dn_i 上执行,运行时间区为 $T=[t_s,t_e]$ 时运行任务 $task$ 所需要的能耗 E_{task} 可表示如下:

$$\begin{aligned}E_{task}&=E_{\mathrm{CPU}}+E_{\mathrm{mem}}+E_{\mathrm{disk}}+E_{\mathrm{net}}=\\&\int_{t_s}^{t_e}(a_{\mathrm{CPU}}\times u_{\mathrm{CPU}}+\lambda_{\mathrm{CPU}})\mathrm{d}t+\int_{t_s}^{t_e}(a_{\mathrm{mem}}\times N+\\&\lambda_{\mathrm{mem}})\mathrm{d}t+\int_{t_s}^{t_e}(a_{\mathrm{read}}\times r+a_{\mathrm{write}}\times w+\lambda_{\mathrm{disk}})\mathrm{d}t+\\&\int_{t_s}^{t_e}(a_{\mathrm{send}}\times s+a_{\mathrm{receive}}\times v+\lambda_{\mathrm{net}})\mathrm{d}t.\end{aligned} \tag{13}$$

基于主要部件能耗累加的能耗模型相比基于 CPU 利用率估算能耗模型不仅考虑了 CPU 能耗,还考虑了内存、磁盘及网络等部件的能耗使用,准确地表达了任务在执行过程各部件的能耗,理论上比基于 CPU 利用率估算的能耗模型更为完毕、精确. 由于 MapReduce 任务主要分为 CPU 密集型、网络 I/O 密集型、磁盘 I/O 密集型、Map 阶段 CPU 与 Reduce 阶段 I/O 密集型 5 类,基于主要部件能耗累加的能耗模型更能够清晰地对不同 MapReduce 任务类型的能耗行为进行描述. 但是,由于参数偏多且参数值的取得较为困难,加之资源在任务执行过程中的动态性使得基于主要部件能耗累加的能耗模型计算更为复杂. 所以,本文的研究重点是基于 CPU 利用率估算的能耗模型与基于平均功耗估算的能耗模型. 但是,由于具有更为精确的特点,基于主要部件能耗累加的能耗模型将是将来研究工作的重点.

2.2.3 基于平均功耗估算的能耗模型

作业的任务分解及任务(定义 1)与资源之间的映射关系(定义 2)由作业调度系统确定. 当某任务 $task\in M\cup R$,数据量大小为 D_i,调度到节点 dn_i 上执行时,其运行时间 T_i 可通过式(21)进行预测(节点 dn_i 对任务 $task_i$ 的计算能力可通过大量数据训练获得). 设任务 $task_i$ 在节点 dn_i 上运行的时间区间 T_i 内,节点 dn_i 的平均功耗为 $\overline{p_i}$. 同样,$\overline{p_i}$ 的值可通过大量的实验及数据分析获得. 当 $task_i$ 的运行时间 T_i 及平均功耗 $\overline{p_i}$ 确定时,$task_i$ 的能耗可由式(14)估算:

$$E_{task}=\overline{p_i}\times T_i=\overline{p_i}\times\frac{D_i}{C_i}. \tag{14}$$

利用基于平均功耗估算的能耗模型对 MapReduce 作业进行能耗计算的唯一前提条件是取得作业的分解与调度结果. 作业的分解与调度结果由 MapReduce 的作业调度系统确定,现有的调度系统在考虑集群 CPU、内存、磁盘及网络等资源状态的基础上,基于作业优先级、截止时间、作业量、作业类型等因素对作业进行调度. 现有调度系统并没有将能耗作为一种资源进行考虑,而基于平均功耗估算的能耗模型主要功能是在作业运行前对作业的执行能耗进行预测,所以基于平均功耗估算的能耗模型是将来研究节能的 MapReduce 作业调度系统的基础.

2.3　MapReduce 作业能耗模型

当 MapReduce 接收到一个作业 *job* 后,MapReduce 框架首先将按照定义 2(作业的任务分解模型)将 *job* 分解为多个 Map 与 Reduce 任务;当 *job* 进入运行状态后,MapReduce 框架按照定义 3 中的资源映射模型将所有的任务绑定到一个空闲资源槽执行.

设$[ms_0,me_0]$与$[rs_0,re_0]$分别表示 Map 任务 mt_0 与 Reduce 任务 rt_0 执行的起始时间,那么可将任务集合 M 与 R 中所有任务起始时间用式(15)表示:

$$\begin{cases} MT=\{[ms_0,me_0],[ms_1,me_1],\cdots,[ms_{m-1},me_{m-1}]\},\\ RT=\{[rs_0,re_0],[rs_1,re_1],\cdots,[rs_{r-1},re_{r-1}]\}. \end{cases} \tag{15}$$

MapReduce 作业的执行能耗为所有子任务(Map 任务与 Reduce 任务)能耗的总和,设 E_{jobexe} 表示某作业的执行能耗,基于 CPU 利用率估算的能耗模型 E_{jobexe} 计算公式推导如式(16)所示:

$$\begin{aligned} E_{\text{jobexe}} &= (E_{mt_0}+E_{mt_1}+\cdots+E_{mt_m})+\\ &(E_{rt_0}+E_{rt_1}+\cdots+E_{rt_r})=\sum_{i=0}^{m}E_{mt_i}+\sum_{u=0}^{r}E_{rt_u}=\\ &\sum_{i=0}^{m}aP_{dn_i}^{\max}(ms_i-me_i)+(1-a)P_{dn_i}^{\max}\times\\ &\sum_{j=0}^{k-1}u_{ij}(t_{j+1}-t_j)+\sum_{u=0}^{r}aP_{dn_u}^{\max}(rs_u-re_u)+\\ &(1-a)P_{dn_u}^{\max}\times\sum_{j=0}^{k-1}u_{u_j}(t_{j+1}-t_j). \end{aligned} \tag{16}$$

基于主要部件能耗累加的能耗模型 E_{jobexe} 计算公式推导如式(17)所示:

$$\begin{aligned} E_{\text{jobexe}} &= (E_{mt_0}+E_{mt_1}+\cdots+E_{mt_m})+\\ &(E_{rt_0}+E_{rt_1}+\cdots+E_{rt_r})=\sum_{i=0}^{m}E_{mt_i}+\sum_{u=0}^{r}E_{rt_u}=\\ &\sum_{i=0}^{m}\int_{t_s}^{t_e}(a_{\text{CPU}}\times u_{\text{CPU}}+\lambda_{\text{CPU}})_i\,\mathrm{d}t+\\ &\sum_{i=0}^{m}\int_{t_s}^{t_e}(a_{\text{mem}}\times N+\lambda_{\text{mem}})_i\,\mathrm{d}t+\\ &\sum_{i=0}^{m}\int_{t_s}^{t_e}(a_{\text{read}}\times r+a_{\text{write}}\times w+\lambda_{\text{disk}})_i\,\mathrm{d}t+\\ &\sum_{i=0}^{m}\int_{t_s}^{t_e}(a_{\text{send}}\times s+a_{\text{receive}}\times v+\lambda_{\text{net}})_i\,\mathrm{d}t+\\ &\sum_{u=0}^{r}\int_{t_s}^{t_e}(a_{\text{CPU}}\times u_{\text{CPU}}+\lambda_{\text{CPU}})_u\,\mathrm{d}t+\\ &\sum_{u=0}^{r}\int_{t_s}^{t_e}(a_{\text{mem}}\times N+\lambda_{\text{mem}})_u\,\mathrm{d}t+\\ &\sum_{u=0}^{r}\int_{t_s}^{t_e}(a_{\text{read}}\times r+a_{\text{write}}\times w+\lambda_{\text{disk}})_u\,\mathrm{d}t+\\ &\sum_{u=0}^{r}\int_{t_s}^{t_e}(a_{\text{send}}\times s+a_{\text{receive}}\times v+\lambda_{\text{net}})_u\,\mathrm{d}t. \end{aligned} \tag{17}$$

基于平均功耗估算的能耗模型 E_{jobexe} 计算公式推导如式(18)所示:

$$\begin{aligned} E_{\text{jobexe}} &= (E_{mt_0}+E_{mt_1}+\cdots+E_{mt_m})+\\ &(E_{rt_0}+E_{rt_1}+\cdots+E_{rt_r})=\sum_{i=0}^{m}E_{mt_i}+\sum_{u=0}^{r}E_{rt_u}=\\ &\sum_{i=0}^{m}(\overline{p_i}\times\frac{D_i}{C_i})+\sum_{u=0}^{r}(\overline{p_u}\times\frac{D_u}{C_u}). \end{aligned} \tag{18}$$

3　能耗优化分析

3.1　能耗优化方法讨论

基于 MapReduce 作业能耗模型与 MapReduce 调度模型,本节对 MapReduce 作业的节能优化问题进行讨论. 下面分别从优化单个 MapReduce 作业的执行能耗与提高 MapReduce 集群能源效率(MPUE),即从作业级与系级 2 方面出发,提出以下 3 个方面对 MapReduce 作业能耗进行优化:

1) 优化 MapReduce 作业执行能耗

基于 2.3 节提出的 3 种作业能耗模型,优化单个 MapReduce 作业执行能耗即是在相同作业条件下求解式(16)~(18)的极小值,具体可分为以下 2 种方法:

① 采用低能耗高效率的硬件设备. 基于 CPU 利用率估算的能耗模型中(如式(16)所示),可通过减小 a 值或 $a\times P_{dn_i}^{\max}$ 值达到减小 MapReduce 作业的执行能耗的目的. 基于主要部件能耗累加的能耗模型中(如式(17)所示),可通过减小 a_{CPU},λ_{CPU},a_{mem},λ_{mem},a_{read},a_{write},λ_{disk},a_{send},a_{receive},λ_{net} 等参数的值达到减小 MapReduce 作业的执行能耗的目的. 基于平均功耗估算的能耗模型中(如式(18)所示),可通过减小

$\overline{p_i}$值或减小 D_i/C_i 值优化 MapReduce 作业的执行能耗.

事实上,方法①的节能实质是采用低能耗高效率的硬件设备或体系结构(a 与 λ 参数值较小),对现有的高能耗设备进行替换,从而达到节能的目的.已有大量工作[59,61-62]采用此方法,并取得了不错的节能效果.基于方法①,节能效果立竿见影,且不需要复杂的软件节能方法相匹配;但是对于已经部署的大规模 MapReduce 应用系统,大批量的硬件替换面临成本过高的问题.

② 寻找任务数量与任务执行时间总和之间的平衡点.根据式(16)与式(17)可知,当 a 与 λ 参数值固定时,可通过求解式(19)得到式(16)与式(17)的极小值:

$$\min\left(\sum_{i=0}^{m}(ms_i-me_i)+\sum_{u=0}^{r}(rs_u-re_u)\right). \quad (19)$$

式(19)中 Map 任务的数 m 主要由逻辑单元 Split 决定,多数情况下 Split 与 Block 呈一一对应关系,数据块 Block 与 Split 对应关系如图 2 所示;Reduce 任务数 r 可由任务调度算法进行指定.理想情况下,相同作业被分解后的 Map 任务数目与 Reduce 任务数目越大,任务完成时间越短;但是,实际情况则不是这样,如图 3 所示为 TeraSort 任务在不同任务(Map 任务数量固定,Reduce 数量增多)数条件下的完成时间比较.图 3 表明任务数目与任务完成时间之间并不呈线性关系,当任务数量增加到某临界点时,任务完成时间并不会因为任务数量的增加而减小.一方面由于 MapReduce 集群资源状态具有动态性;另一方面当作业类型不同时,其任务数目与任务完成时间之间关系也存在着较大差异.以上 2 点原因造成试图通过求解式(19)来优化 MapReduce 作业能耗面临诸多挑战.但是,一种可行的方案是针对单一的作业类型,可通过大量的实验取得基础数据,对基础数据进行分析得到式(19)的近似解.

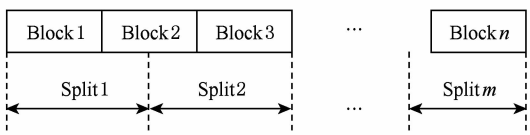


Fig. 2 The relationship between data Block and Split.

图 2 数据块 Block 与 Split 之间的对应关系

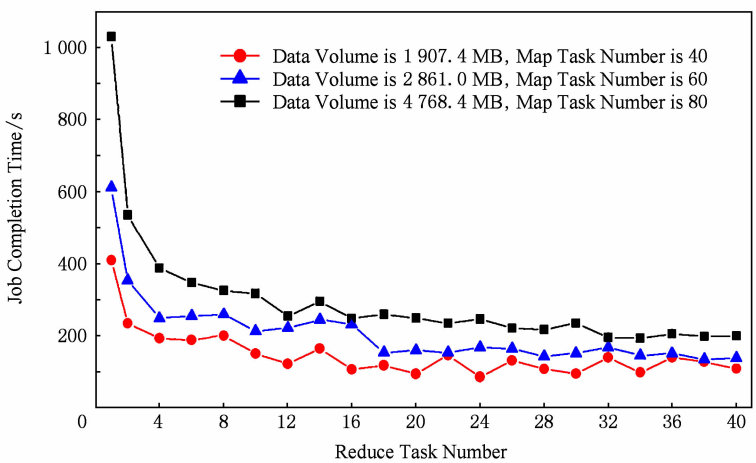


Fig. 3 The relationship between task number and job completion time (with 10 datanodes cluster).

图 3 任务数与作业完成时间之间的关系(10 节点集群环境下)

2) 减少 MapReduce 任务等待能耗

当前 Hadoop 中采用机架感知的数据存放策略,将数据文件切分为相同大小的数据块(block)随机存储到集群 DataNode 节点中.在同构环境中,这种数据的切分与存储方法能够满足系统可用性与负载分流的要求.但是,在异构环境中,由于集群中各节点的计算能力存在着差异,异构节点处理相同任务(任务数据集大小相同)的完成时间不同.由于只有当一个作业的 Map 任务成功完成的数量超过一定的阈值时,才能开始分配该作业的 Reduce 任务给某 TaskTracker 节点执行,所以对于计算机能力强的节点,机架感知的数据存放策略会造成大量的等待时延.MapReduce 任务执行过程中任务之间并不是按照完全并行的方式进行的,Map 与 Reduce 任务之间存在不同程度的执行顺序与数据调用的制约关系.当某任务处于等待其他任务的执行结果(或等待其他任务的执行完毕)才能继续往下执行而处于"被动空闲"状态时,"任务等待能耗"便产生了,下面对 MapReduce 任务等待能耗进行定义.

定义 4. MapReduce 任务等待能耗.在 Map-

Reduce 任务执行过程中,某些 Map 任务或 Reduce 任务由于等待其他任务的完结或执行结果,或者等待额外资源的分配才能继续往下执行,将这些任务所处的“被动空闲”状态所产生的能耗定义为 MapReduce 任务等待能耗.

同构环境中,由于各节点计算能力相同,各并行任务产生 MapReduce 任务等待能耗的概率较小.本文在 3.2 节提出异构环境下数据的放置策略,有效地缩短等待时延的同时达到减小“MapReduce 任务等待能耗”的目的.

3) 提高 MapReduce 集群能源利用效率

如图 4 所示为 WordCount 与 TeraSort 两种作业在 10 个节点的 MapReduce 集群环境中同时执行时的资源利用图.其中 WordCount 作业 Map 任务数为 10,Reduce 任务数为 2;TeraSort 作业 Map 任务数为 8,Reduce 任务数为 5.从图 4 可以看出,即使在 MapReduce 集群并行处理多个作业的情况下,Map 资源槽与 Reduce 资源槽出现空闲的现象(特别是 Reduce 资源槽).特别当 MapReduce 集群资源数目较多且作业较少的情况下,资源空闲的现象将更为严重.

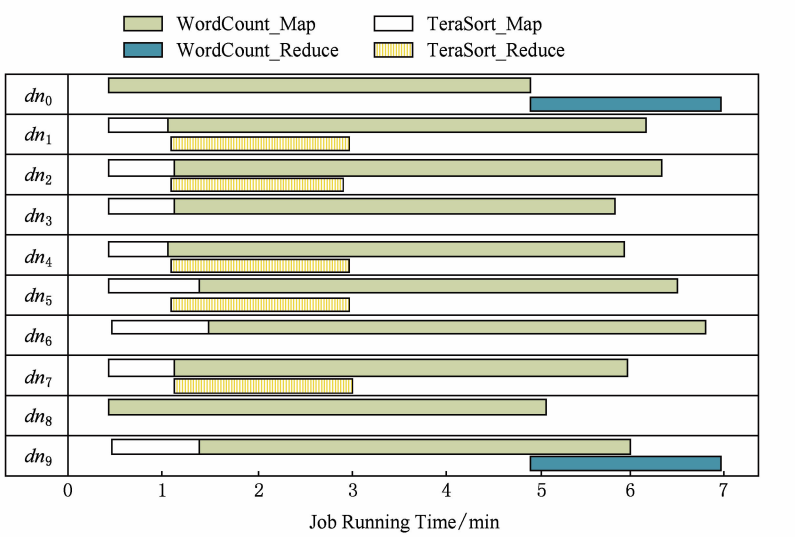


Fig. 4　The usage of slot by WordCount and TeraSort job.

图 4　WordCount 与 TeraSort 两种作业 slot 占用图示

为了提高集群能耗利用率,传统的方法是减小“空闲能耗”,此方法在 MapReduce 集群系统中同样适用.在解决数据块可用性的前提下(即关闭空闲节点不会影响 MapReduce 任务的执行),可通过减小“空闲能耗”达到优化 MapReduce 的能耗利用率.在我们的前期工作中,已经通过构造数据块可用性度量矩阵[63-65],从数据的存储层解决了数据可用性完全覆盖问题.所以,在解决数据可用性问题且满足作业的截止时间约束的前提条件下,将作业队列中的作业压缩(调度)到合适的资源集上执行,并关闭“空闲资源”以达到提高 MapReduce 集群能源利用率是一种切实可行的办法.本文 3.3 节提出截止时间约束下的最小资源槽 *slot* 分配方法,在满足作业任务截止时间约束的前提下,计算完成该任务所需最少资源;通过任务调度方法将作业调度到部分资源集上,由此可产生大量的“空闲资源”.为了统一 MapReduce 集群能源利用效率的度量标准,需要对 MapReduce 任务执行系统的能源使用效率的度量标准——MapReduce 集群能源效率(MapReduce power usage effectiveness, *MPUE*)进行定义.

定义 5. MapReduce 集群能源效率(*MPUE*).设在时间段 T 内,某 MapReduce 集群总共完成 $n(n\geqslant 0)$ 个作业,这些作业的执行能耗分别为 $E_i(0\leqslant i\leqslant n-1)$,并设时间段 T 内集群总能耗为 E_{cluster}.特定义在时间段 T 内,MapReduce 能源效率(*MPUE*)来衡量该 MapReduce 集群的能源利用效率.*MPUE* 可由如式(20)进行计算.

$$MPUE=\frac{\sum_{i=0}^{n}E_i}{E_{\text{cluster}}},\tag{20}$$

其中,$MPUE\in[0,1]$.*MPUE* 值越大,表明该集群 MapReduce 作业能源利用效率越高.当 $MPUE=0$,说明在时间段 T 内没有 MapReduce 作业执行;$MPUE=1$ 为理想情况,表明集群所有能耗都花费在了执行作业操作上.*MPUE* 表达的是集群层面的能源利用效率,也是衡量集群效能与 *slot* 资源利用率的重要标准.

3.2 异构环境下数据的放置策略

异构环境中的数据放置策略应当适应不同节点的计算能力,数据块的大小应与存储该数据块节点的数据处理能力成正比.这样的数据块切分原则可以有效地屏蔽异构集群中各个节点处理能力的差异性,保证多个并行数据处理任务尽量在相同的时间内完成,在有效地缩短等待时延的同时减小"MapReduce任务等待能耗".

设 C_{ij} 表示节点 dn_i 处理任务 $task_j$ 时的计算能力,那么 C_{ij} 可表示为:

$$C_{ij}=\frac{D_{ij}}{T_{ij}}, \tag{21}$$

其中,T_{ij} 表示节点 dn_i 完成任务 $task_j$ 所花的时间,D_{ij} 表示任务所处理数据量的大小.Hadoop集群中每个节点对于不同任务的计算能力可通过大量的训练与测试得到,本文中设 C_{ij} 值为已知.可将训练后不同节点处理不同任务时的计算能力用表1进行记录.

Table 1 Computing Capability of DataNodes for Each Task

表1 每个任务的节点计算能力记录表

Task	dn_1	dn_2	…	dn_i
$task_1$	C_{11}	C_{21}	…	C_{i1}
$task_2$	C_{12}	C_{22}	…	C_{i2}
⋮	⋮	⋮		⋮
$task_j$	C_{1j}	C_{2j}	…	C_{ij}

设某MapReduce作业 $J\in job$ 被分解为 n 个Map任务,每个Map任务映射到具有不同计算能力的节点资源 $slot$ 上.为使这 n 个Map任务完成时间均衡并缩短等待时延,数据的切分规则应满足式(22):

$$\begin{cases}\dfrac{C_1}{D_1}=\dfrac{C_2}{D_2}=\cdots=\dfrac{C_n}{D_n},\\ D=D_1+D_2+\cdots+D_n,\end{cases} \tag{22}$$

其中,$C_1,C_2,\cdots,C_n$ 与 $D_1,D_2,\cdots,D_n$ 分别表示对应节点的计算能力与被分配到的数据量;D 表示作业 J 需要处理的总数据量.

3.3 截止时间约束下的最小资源分配

在数据放置策略确定并且节点的计算能力已知的情况下,能够对任务的完成时间进行有效的预测.本节提出截止时间约束下的最小资源分配计算方法,在满足作业完成时间deadline约束前提下,计算完成该任务所需最少资源;通过任务调度方法将作业集中地调度到部分资源集上.一方面,该调度方法可产生大量的"空闲资源",通过将空闲资源进行休眠处理以达到节能的目的;另一方面,该调度策略也能提高单作业的能耗效率.

设某MapReduce作业 J 被初始化为 N_M 个Map任务与 N_R 个Reduce任务,并且这些任务在拥有 R_M 个Map任务执行资源(Map slot)与 R_R 个Reduce任务执行资源(Reduce slot)的集群中运行.在数据量大小与节点计算能力确定的情况下,能够通过式(21)对Map任务完成时间进行预测,但由于MapReduce任务由Map阶段、shuffle&sort阶段与Reduce阶段组成,shuffle&sort阶段与Reduce阶段的任务完成时间无法进行直接计算.所以,下面通过任务采样的方法来预测作业 J 的完成时间,其主要步骤如下:

1) 从 N_M 个Map任务与 N_R 个Reduce任务中分别挑选出 N_M^S 个Map采样任务与 N_R^S 个Reduce采样任务.

2) 在集群中运行作业 J 的采样任务并记录每个采样Map任务 $i(1\leqslant i\leqslant N_M^S)$ 的完成时间 $T_M(i)$ 和任务输入数据大小 $D_M(i)$.利用得到的采样数据建立Map任务输入数据量与完成时间的函数关系:

$$T_M(i)=f(C,D_M(i)). \tag{23}$$

3) 记录每个采样任务 $j(1\leqslant j\leqslant N_R^S)$ 在shuffle&sort阶段的完成时间 $T_S(j)$ 与处理数据量大小 $D_S(j)$.利用得到的采样数据建立shuffle&sort阶段处理数据量与完成时间的函数关系:

$$T_S(j)=g(C,D_S(j)). \tag{24}$$

4) 记录每个采样Reduce任务 j 的完成时间 $T_R(j)$ 与处理数据量大小 $D_R(j)$.利用得到的采样数据建立Reduce任务输入数据量与完成时间的函数关系:

$$T_R(j)=h(C,D_R(j)). \tag{25}$$

其中,式(22)~(24)中的 C 表示任意节点 dn_{ij} 处理作业 J 时的计算能力.

5) 计算Map阶段的采样任务平均完成时间为

$$T_M^A=\frac{\sum_{i=1}^{N_M^S}f(C,D_M(i))}{N_M^S}. \tag{26}$$

6) 计算shuffle&sort阶段的采样任务平均完成时间为

$$T_S^A=\frac{\sum_{j=1}^{N_R^S}g(C,D_S(j))}{N_R^S}. \tag{27}$$

7) 计算 Reduce 阶段的采样任务平均完成时间为

$$T_{\mathrm{R}}^{\mathrm{A}}=\frac{\sum_{j=1}^{N_{\mathrm{R}}^{\mathrm{S}}}h(C,D_{\mathrm{R}}(j))}{N_{\mathrm{R}}^{\mathrm{S}}}. \tag{28}$$

基于采样任务的运行结果,可预测作业 J 在 Map 阶段的平均完成时间为

$$T_{\mathrm{M}}^{\mathrm{avg}}\approx\frac{N_{\mathrm{M}}}{R_{\mathrm{M}}}\times T_{\mathrm{M}}^{\mathrm{A}}\approx\frac{N_{\mathrm{M}}}{R_{\mathrm{M}}}\times\frac{\sum_{i=1}^{N_{\mathrm{M}}^{\mathrm{S}}}f(C,D_{\mathrm{M}}(i))}{N_{\mathrm{M}}^{\mathrm{S}}}. \tag{29}$$

同样的方法,可预测 Reduce 阶段(包含 shuffle&sort 阶段)任务的平均完成时间为

$$T_{\mathrm{R}}^{\mathrm{avg}}\approx\left(\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}-1\right)T_{\mathrm{S}}^{\mathrm{A}}+\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}\times T_{\mathrm{R}}^{\mathrm{A}}\approx\left(\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}-1\right)\times\frac{\sum_{j=1}^{N_{\mathrm{R}}^{\mathrm{S}}}g(C,D_{\mathrm{S}}(j))}{N_{\mathrm{R}}^{\mathrm{S}}}+\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}\times\frac{\sum_{j=1}^{N_{\mathrm{R}}^{\mathrm{S}}}h(C,D_{\mathrm{R}}(j))}{N_{\mathrm{R}}^{\mathrm{S}}}. \tag{30}$$

值得注意的是,由于作业 J 在集群中运行 shuffle&sort 的第 1 次操作与 Map 阶段有重叠部分,所以式(29)中减去了重叠部分的时间.

设 T_{avg} 表示作业 J 的总完成时间,在式(29)与式(30)的基础上,总完成时间为

$$T_{\mathrm{avg}}\approx T_{\mathrm{M}}^{\mathrm{avg}}+T_{\mathrm{R}}^{\mathrm{avg}}\approx\frac{N_{\mathrm{M}}}{R_{\mathrm{M}}}\times T_{\mathrm{M}}^{\mathrm{A}}+\left(\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}-1\right)T_{\mathrm{S}}^{\mathrm{A}}+\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}\times T_{\mathrm{R}}^{\mathrm{A}}. \tag{31}$$

对式(31)进行变换,可得到式(32):

$$T_{\mathrm{avg}}+T_{\mathrm{S}}^{\mathrm{A}}\approx\frac{N_{\mathrm{M}}\times T_{\mathrm{M}}^{\mathrm{A}}}{R_{\mathrm{M}}}+\frac{N_{\mathrm{R}}\times(T_{\mathrm{S}}^{\mathrm{A}}+T_{\mathrm{R}}^{\mathrm{A}})}{R_{\mathrm{R}}}. \tag{32}$$

在式(31)中,设 $A_J=N_{\mathrm{M}}\times T_{\mathrm{M}}^{\mathrm{A}}$,$B_J=N_{\mathrm{R}}\times(T_{\mathrm{S}}^{\mathrm{A}}+T_{\mathrm{R}}^{\mathrm{A}})$,$C_J=T_{\mathrm{avg}}+T_{\mathrm{S}}^{\mathrm{A}}$,可将式(32)简化为

$$C_J=\frac{A_J}{R_{\mathrm{M}}}+\frac{B_J}{R_{\mathrm{R}}}. \tag{33}$$

本文提出截止时间约束下的最小资源分配的计算方法,即是 R_{M} 与 R_{R} 值最小,通过最优化的办法建立最优化目标函数为

$$f(R_{\mathrm{M}},R_{\mathrm{R}})=R_{\mathrm{M}}+R_{\mathrm{R}},\ \text{s.t.}\ C_J=\frac{A_J}{R_{\mathrm{M}}}+\frac{B_J}{R_{\mathrm{R}}}. \tag{34}$$

利用拉格朗日乘数法求函数 $f(R_{\mathrm{M}},R_{\mathrm{R}})$ 最小值为

$$\begin{cases}R_{\mathrm{M}}=\dfrac{\sqrt{A_J}(\sqrt{A_J}+\sqrt{B_J})}{C_J},\\ R_{\mathrm{R}}=\dfrac{\sqrt{B_J}(\sqrt{A_J}+\sqrt{B_J})}{C_J}.\end{cases} \tag{35}$$

分别将 $A_J=N_{\mathrm{M}}\times T_{\mathrm{M}}^{\mathrm{A}}$,$B_J=N_{\mathrm{R}}\times(T_{\mathrm{S}}^{\mathrm{A}}+T_{\mathrm{R}}^{\mathrm{A}})$,$C_J=T_{\mathrm{avg}}+T_{\mathrm{S}}^{\mathrm{A}}$ 代入式(35),可得:

$$\begin{cases}R_{\mathrm{M}}=\left\{\sqrt{N_{\mathrm{M}}\times T_{\mathrm{M}}^{\mathrm{A}}}\left(\sqrt{N_{\mathrm{M}}\times T_{\mathrm{M}}^{\mathrm{A}}}+\sqrt{N_{\mathrm{R}}\times(T_{\mathrm{S}}^{\mathrm{A}}+T_{\mathrm{R}}^{\mathrm{A}})}\right)\right\}\Big/\left\{T_{\mathrm{avg}}+T_{\mathrm{S}}^{\mathrm{A}}\right\},\\ R_{\mathrm{R}}=\left\{\sqrt{N_{\mathrm{R}}\times(T_{\mathrm{S}}^{\mathrm{A}}+T_{\mathrm{R}}^{\mathrm{A}})}\left(\sqrt{N_{\mathrm{M}}\times T_{\mathrm{M}}^{\mathrm{A}}}+\sqrt{N_{\mathrm{R}}\times(T_{\mathrm{S}}^{\mathrm{A}}+T_{\mathrm{R}}^{\mathrm{A}})}\right)\right\}\Big/\left\{T_{\mathrm{avg}}+T_{\mathrm{S}}^{\mathrm{A}}\right\}.\end{cases} \tag{36}$$

由于 MapReduce 作业的类型是有限的,相同类型的作业 Map,shuffle&sort,Reduce 的数据量与完成时间的函数(式(26)~(28))相同. 所以对于相同类型的 MapReduce 作业,函数式(26)~(28)具有可重用性. 即通过大量的训练与测试,可确定不同类型作业的处理数据量与完成时间之间的函数关系. 再则,通过 3.2 节的方法可确定任意节点 dn_i 处理作业 J 时的计算能力,所以,式(36)可解. 同样,当参数 R_{M} 与 R_{R} 值已知时,可利用式(36)确定参数 N_{M} 与 N_{R} 的值,即当可用 Map 资源 *slot* 及 Reduce 资源 *slot* 已知的情况下,可对满足作业完成时间 deadline 约束条件下的最小 Map 及 Reduce 任务数进行计算.

从图 3 可知,MapReduce 任务数目与任务完成时间之间并不呈线性关系,当任务数量增加到某临界点时,任务完成时间并不会因为任务数量的增加而减小. 通过式(36)可计算出任意作业在满足截止时间约束前提下的最小资源分配 R_{M} 与 R_{R}. 根据 MapReduce 作业执行原理,R_{M} 值可指导数据的分块策略(或 Split 切分原则);R_{R} 值确定了 Reduce 任务的数量. 由于截止时间约束下的最小资源分配策略减少了执行相同作业所需的资源,从而减小了完成单个 MapReduce 作业所需能耗,优化了单个 MapReduce 作业对能耗的利用效率.

从整个集群的角度看,如图 5 所示为 MapReduce 作业的节能调度模型,当所有的 MapReduce 作业切分任务数量减少时,完成相同作业所需的资源数量将减少,即截止时间约束下的最小资源分配策略将更容易产生空闲节点. 特别当系统空闲时段,截止时间约束下的最小资源分配策略可将任务集中分配在少量的节点上,从而产生大量的空闲节点. 将空闲节点进行休眠处理,将大大提高 *MPUE* 值. 特别地,利用休眠空闲节点方法进行节能时需要考虑数据的可用性需求,该问题属于数据存储层的问题,可参考

文献[46,63]. 由于提高 MapReduce 集群的能源利用效率(*MPUE*)主要涉及 MapReduce 作业及任务的调度问题,涉及的研究内容较多,不作为本文的研究重点,是下一步研究的主要内容.

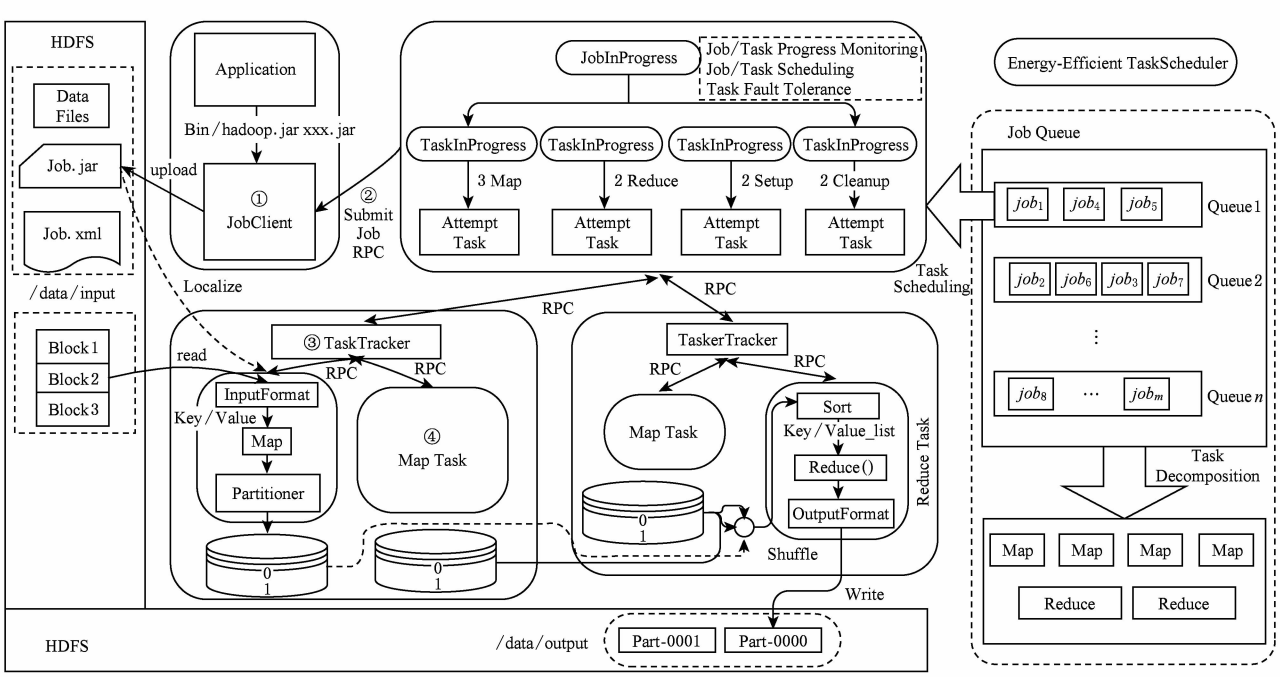


Fig. 5 Energy-Effcient scheduling model for MapReduce jobs.

图 5 MapReduce 作业的节能调度模型

4 能耗模型实验与结果分析

本节对 3 种能耗模型及作业运行时的资源行为进行了实验分析,讨论了 3 种模型的不同特点及适用的场景;对基于 CPU 利用率估算与基于平均功耗估算的能耗模型进行了运用及误差分析;对 3.1 节中讨论的能耗优化方法进行了实验.

4.1 实验环境及参数配置

为了对 MapReduce 能耗模型进行实验分析,项目组搭建了拥有 52 个节点的 Hadoop 集群;其中 NameNode 与 SecondNameNode 分别独立为 1 个节

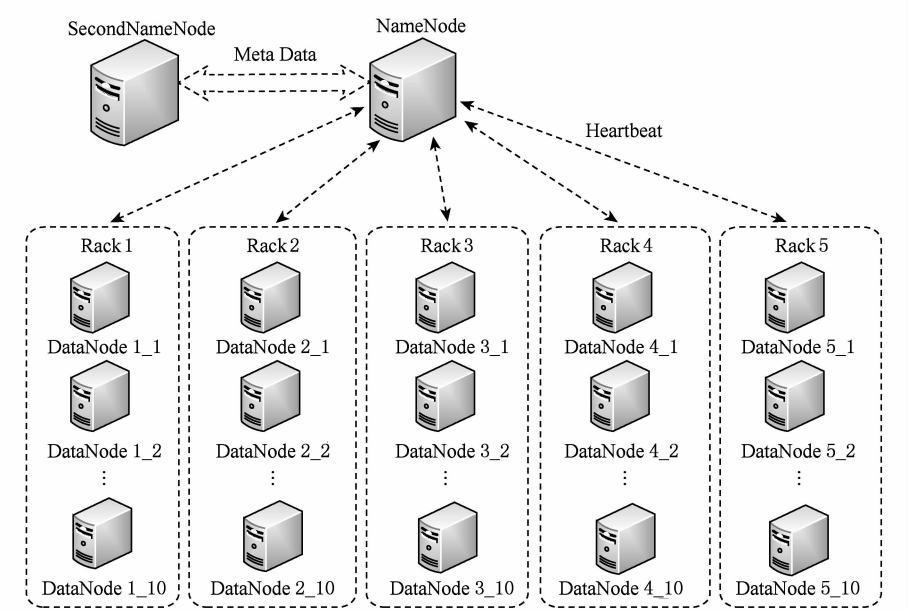


Fig. 6 Topology diagram of experimental environment.

图 6 实验环境拓扑结构图

点,其余 50 个节点为 DataNode(5Rack×10DataNode).实验拓扑结构如图 6 所示.

为了控制实验过程中的 Map 任务数量,达到控制实验数据的统计与计算量的目的,特将数据块分块大小配置为 512 MB,即 *dfs.block.size*=512MB.单个 DataNode 节点上 Map 与 Reduce 任务 *slot* 资源槽数设置为 1,即配置项 *mapred.tasktracker.map.tasks.maximum*=1 与 *mapred.tasktracker.reduce.tasks.maximum*=1.能耗数据测量方面,实验采用北电电力监测仪(USB 智能版),数据采样频率设置为每次 1 s,各节点能耗数据(包括瞬时功率、电流值、电压值、能耗累加值等)可通过 USB 接口实时地传输到能耗数据监测机上,实现能耗数据的收集.实验总体环境描述如表 2 所示:

Table 2　Description of Experimental Environment

表 2　总体实验环境描述

Parameters	Values
OS	Debian 7.0
Java version	1.6 for Linux
Hadoop version	1.0.4
Energy consumption data measurement	Nortel power monitor (USB intelligent edition), the standard is GB/T17215—2003, the power error value is between 0.01 - 0.1 W, the sampling frequency is 1.5 - 3 s, the unit is kW·h
Energy consumption data acquisition	Monitoring and management system for power monitoring system V1.0.1
Energy consumption unit	Power: Watt (W), energy consumption: Joule (J)
Data sampling frequency	Acquisition data 1 time each second
DataNode CPU	Intel core2 duo E8400 3.00 GHz
DataNode RAM	2 GB-DDR2-800 MHz
DataNode Hard Disk	Hitachi HDP725032GLA380(320GB 7200 r/s)
Network	Realtek RTL8168/8111 PCI-E Gigabit Ethernet NIC-100Mbps

4.2　能耗模型与作业资源行为分析

本节选用不同种类的作业进行实验,通过数据分析总结 MapReduce 作业能耗与不同资源(主要包括 CPU、磁盘、内存及网络)利用行为之间的关系.

本节选取 WordCount, TeraSort, NutchIndex, *K*-means, Bayes, PageRank 六种作业进行实验,作业参数设置如表 3 所示:

Table 3　Description of MapReduce Jobs

表 3　作业类型说明

Job	Parameter Configuration
WordCount	The total data volume is 9 759.6 MB, and the number of Map and Reduce task is 10.
TeraSort	The total data volume is 9 536.7 MB, and the number of Map and Reduce task is 10.
NutchIndex	The number of page is 1 000 000, and the number of Map is 80, and the number of Reduce is 10.
K-means	Cluster number is 5, and the number of sample is 40 000 000, and the number of sample in each input file is 4 000 000, and dimension is 20, and the maximum number of iterations is 1, and the number of Map is 10, and the number of Reduce is 1.
Bayes	The number of page is 50 000, and the classification number is 100, and the number of ngrams is 10, and the number of Map is 1, and the number of Reduce is 100.
PageRank	Page number is 3 000 000, and the number of iterations is set to 3, and the parameters of Block and Block_width are set to 0 and 16 respectively, and the number of Map is 10, and the number of Reduce is 1.

本实验中将 DataNode 节点规模配置为 10,通过设置作业的 Map 与 Reduce 任务数(如表 3 所示)为 DataNode 节点的整数倍,使得 Map 与 Reduce 任务均分到每个 DataNode 节点.通过 10 台能耗监测仪对作业运行中的所有 DataNode 节点功耗进行采样,利用时间戳关联实时功耗数据与节点 CPU 利用率.6 种作业运行时节点功耗与 CPU 负载之间的关系比如图 7 所示:

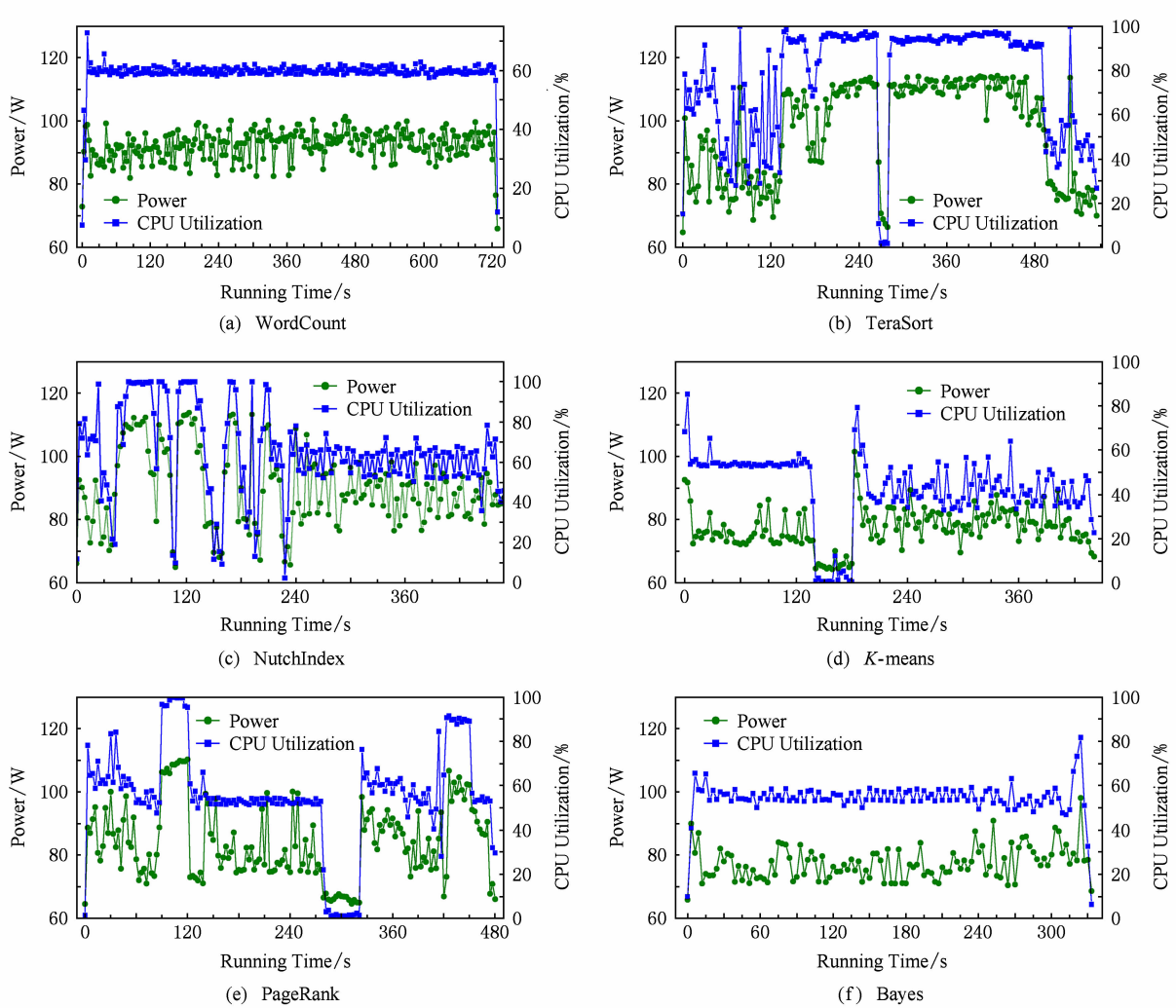


Fig. 7 Comparison between real-time power consumption and CPU utilization.

图 7 作业实时功耗与 CPU 利用率之间的比较

通过图 7 可以看出 6 种作业实时功耗与 CPU 利用率之间联系紧密;当 CPU 利用率上升时能耗上升;当 CPU 利用率下降时能耗下降;CPU 变化趋势与能耗变化趋势基本一致. 事实上,通过图 7 表明了本文 2.2.1 节中提出的基于 CPU 利用率估算的能耗模型的可行性. 通过大量的能耗数据分析得出节点功耗与 CPU 利用率之间的关系如图 8(a)所示.

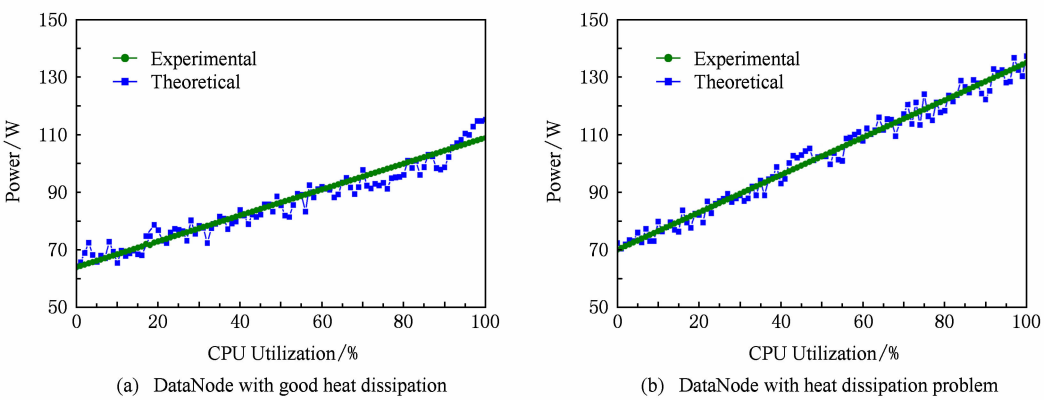


Fig. 8 The relationship between CPU utilization and power consumption .

图 8 DataNode 节点 CPU 利用率与功耗之间的关系

如图 8 所示分别为散热良好(图 8(a))与散热故障(图 8(b))的 DataNode 节点 CPU 利用率与功耗之间的关系.实验值取大量测试数据的平均值,如 CPU 利用率在 50% 时功耗测试数据为{83.1,85.7,88.4,87.5,89.2,84.7,90.3,83.5,82.4,87.2,82.6,85.8,86.9,85.1,84.3,88.2,86.1,83.2,88.8,86.8},取其平均值 85.99 为实验值.实验中我们发现散热良好的与出现散热故障的节点功耗存在较大的差异,散热良好的节点静态功耗为[64,65]W,运行时 CPU 温度稳定在 50～65℃;而出现散热故障的节点静态功耗为[70,72]W,运行时 CPU 在 70～88℃.基于 CPU 利用率估算的能耗模型的计算方法(式(5)与式(6)),可得出散热良好 DataNode 节点 CPU 利用率与功耗之间的理论函数如式(37):

$$P(u)=64+0.5u, \tag{37}$$

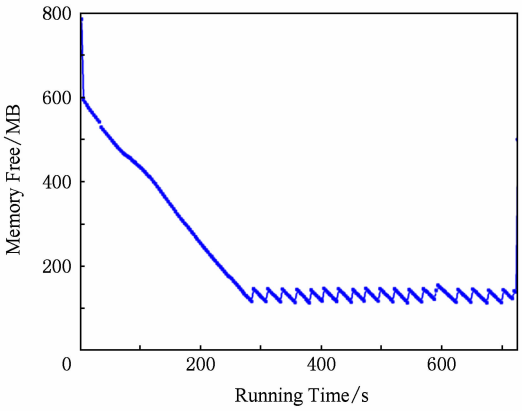
其中,u 表示 CPU 利用率,$P(u)$表示 CPU 利用率为 u 时节点的功耗.节点静态功耗为 64 W,峰值功耗为 110 W,由式(6)得出参数 $a\approx 0.5$.同样方法可得到出现散热故障的 DataNode 节点 CPU 利用率与功耗之间的理论函数:

$$P(u)=70+0.65u. \tag{38}$$

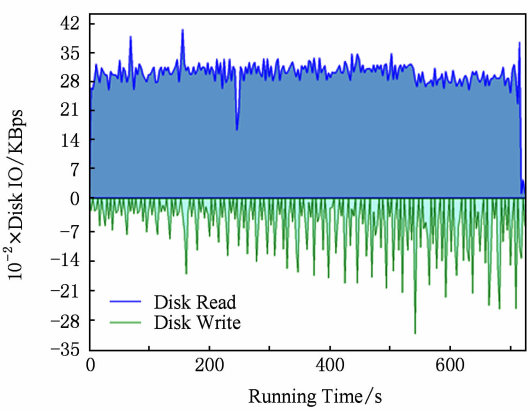
当节点出现散热故障时,节点静态功耗为 70 W,峰值功耗为 135 W,参数 $a\approx 0.65$.值得一提的是,式(37)(38)及图 8 中所示的 CPU 与功耗关系曲线与文献[57]中的结论一致,表明基于 CPU 利用率估算能耗模型的可行性.

利用基于 CPU 利用率估算能耗模型对 MapReduce 作业进行能耗计算的唯一前提条件是取得所有任务运行过程中的 CPU 利用率变化序列.基于 CPU 利用率估算的能耗模型是将来开发 Hadoop 能耗优化及监控软件的理论基础.

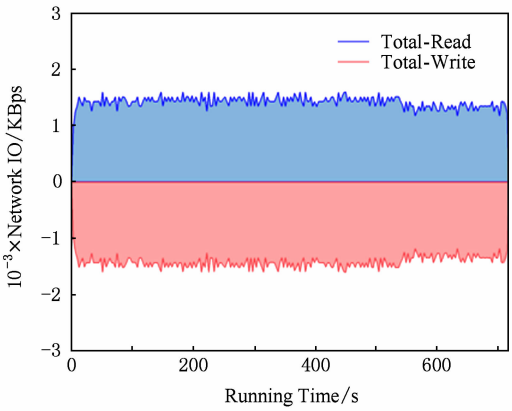
由于基于主要部件能耗累加的能耗模型不仅考虑了 CPU 能耗,还考虑了内存、磁盘及网络等部件的能耗使用.所以除了关注作业执行过程中能耗与 CPU 之间的关系,本实验还对作业执行过程中 DataNode 节点主要资源(内存、磁盘与网络)进行了监测,WordCount,TeraSort,NutchIndex,K-means,PageRank,Bayes 六种作业的内存、磁盘与网络资源运行时状态变化如图 9 所示:



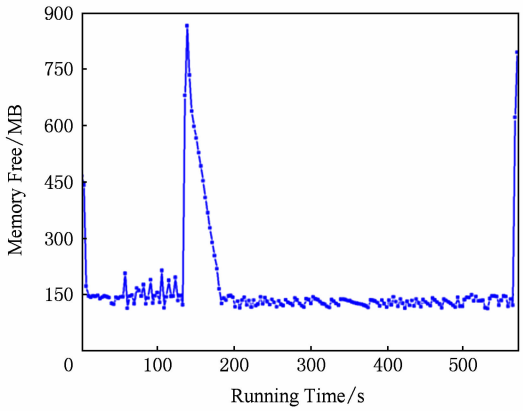
(a) The memory usage of WordCount



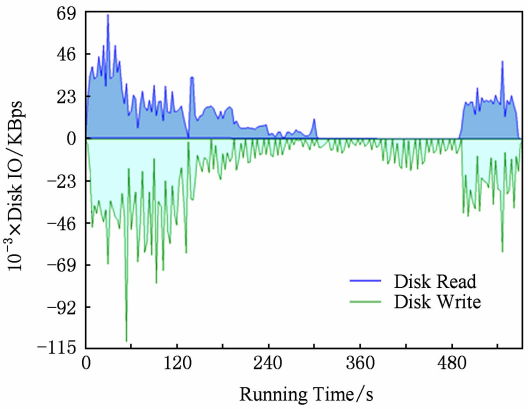
(b) The disk usage of WordCount



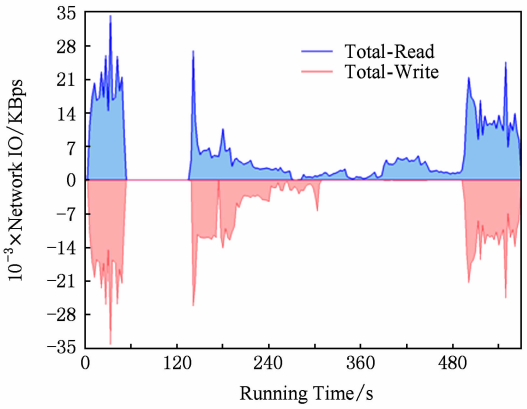
(c) The network usage of WordCount



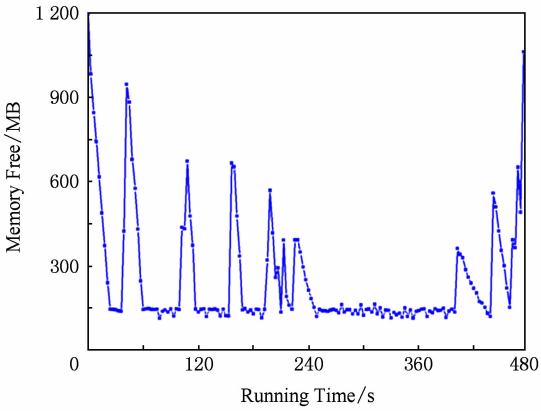
(d) The memory usage of TeraSort



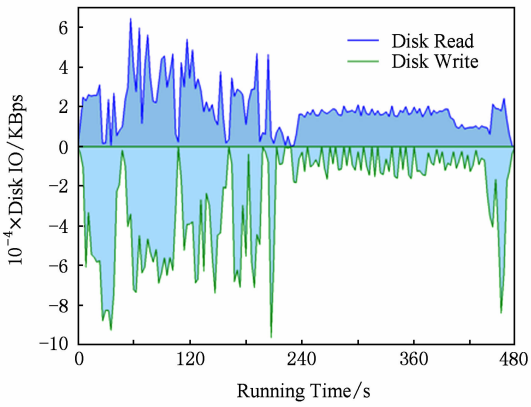
(e) The disk usage of TeraSort



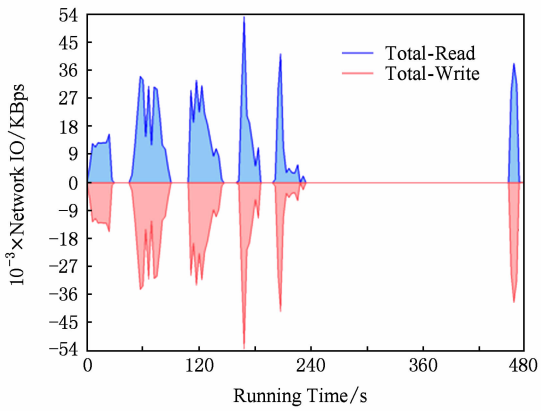
(f) The network usage of TeraSort



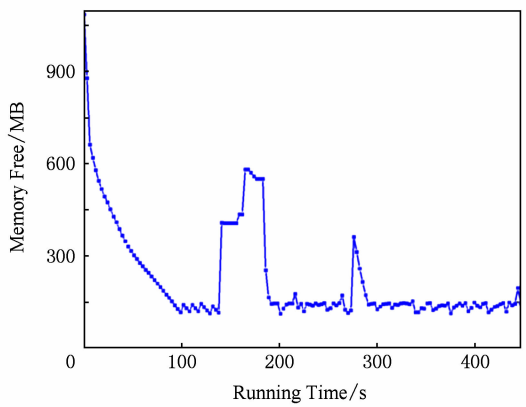
(g) The memory usage of NutchIndex



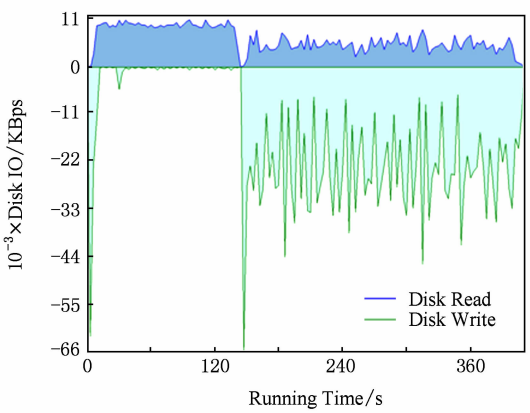
(h) The disk usage of NutchIndex



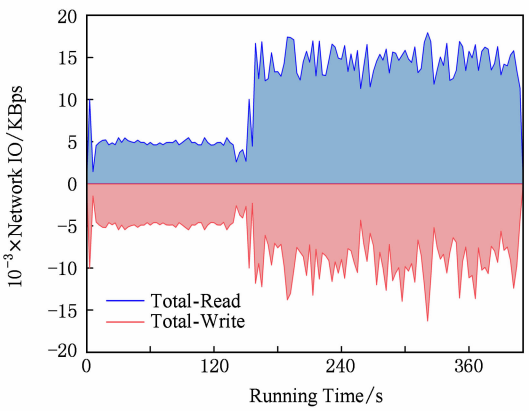
(i) The network usage of NutchIndex



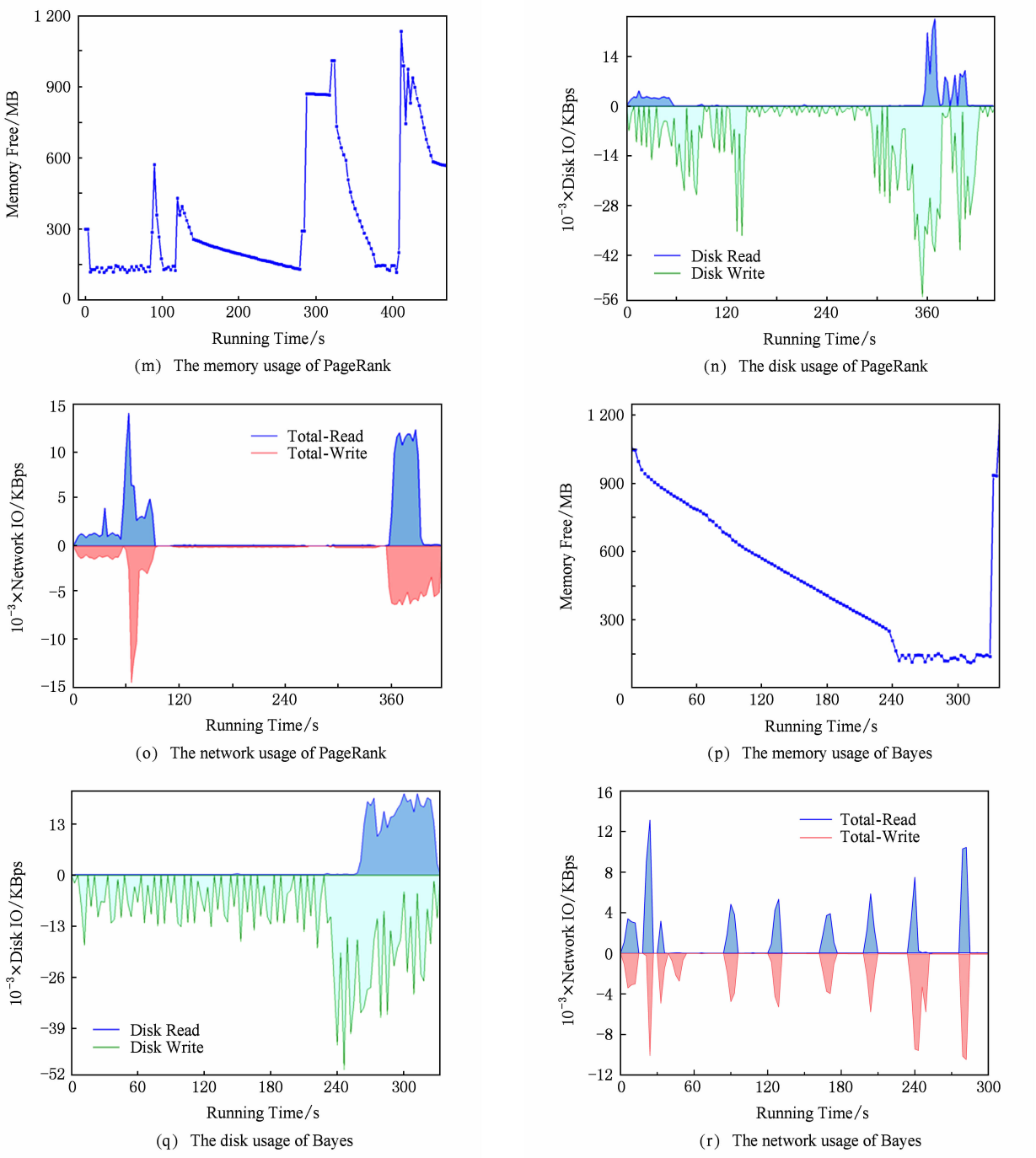
(j) The memory usage of *K*-means



(k) The disk usage of *K*-means



(l) The network usage of *K*-means



(m) The memory usage of PageRank　(n) The disk usage of PageRank

(o) The network usage of PageRank　(p) The memory usage of Bayes

(q) The disk usage of Bayes　(r) The network usage of Bayes

Fig. 9　The comparison of memery, disk and network usage while job running.

图 9　不同作业运行时内存、磁盘及网络资源使用情况对比

通过图 9 可以看出不同作业在运行过程中的内存、磁盘与网络资源的利用存在较大的差异,并且相同作业在不同时间点资源利用率变化也很大. 这使得利用本文 2.2.2 节中提出的基于主要部件能耗累加的能耗模型进行能耗计算时需要进行大量的计算. 确定 CPU 利用率与能耗之间具有规律性,但归纳内存、磁盘与网络资源利用率与能耗之间的关系则具有较大的难度. 虽然基于主要部件能耗累加的能耗模型能够准确地表达任务在执行过程中 CPU、内存、磁盘、网卡等各设备的能耗,但由于模型的参数偏多且参数值的取得较为困难,加之资源利用率的动态性,使得基于主要部件能耗累加的能耗模型实践难度较大.

将 WordCount, TeraSort, NutchIndex, *K*-means, PageRank, Bayes 六种作业分别执行 10 次(减小实验误差),详细的记录各作业 Map 与 Reduce 任务的

开始与结束时间,通过能耗监测仪的采样数据得到每个任务的执行能耗.任务的平均功耗由任务的执行能耗除以任务的执行时间得到,节点对任务的计算能力通过式(21)计算获得.6种作业Map与Reduce任务的平均功耗及计算能力情况如表4所示.其中,WordCount单任务处理数据量为975.88 MB,TeraSort单任务处理数据量为953.67 MB,PageRank单任务处理300 000个页面,*K*-means单任务处理4 000 000个SAMPLE,Bayes单任务处理30 000个页面.

Table 4 Average Power Consumption and Computing Capability of Map and Reduce Task

表4 作业Map与Reduce任务的平均功耗及计算能力

Job	Job Decomposited into Tasks	Map Task Average Power Consumption/W	Map Computing Ability	Reduce Task Average Power Consumption/W	Reduce Computing Ability
WordCount	N/A	89.526	1.3064 MBps	71.2	81.323 3 MBps
TeraSort	N/A	83.4	5.96 MBps	82.75	5.36 MBps
NutchIndex	N/A	89	436.68 page/s	90.5	233.1 page/s
K-means	Cluster Iterator	77.46	5 797.1 sample/s	77.47	7 130.1 sample/s
	Cluster Classification	81.2	7 619 sample/s	N/A	N/A
PageRank	Stage1	83.5	3 333.33 page/s	80	1 685.393 page/s
	Stage2	84	3 333.33 page/s	89.42	7 692.3 page/s
Bayes	DocumentProcessor: DocumentTokeuizer	84.7	681.8 page/s	79.02	1 111 page/s
	CollocDriver.generateCollocations	80.1	12.85 page/s	100.7	158.73 page/s
	CollocDriver.computeNGrams	80.55	434.78 page/s	80.98	833.33 page/s
	DictionaryVectorizer: MakeParitialVectors	91.59	1 250 page/s	84.25	1 111 page/s
	ParticalVectorMerger: MergePartialVectors	82.6	4761.9 page/s	72.85	2 439.02 page/s
	VectorTfld Document Frequency Count	71.39	4 966.89 page/s	73.755	2 481.6 page/s
	MakePratialVectors	85.42	4 862.5 page/s	74.256	2 479.3 page/s
	ParticalVectorMerger: MergePartialVectors	75.92	4 874.2 page/s	73.26	2 463.6 page/s
	TrainNaiveBayesJob IndexInstancesMapper-Reducer	70.18	4 951.45 page/s	75.18	2 015.48 page/s
	TrainNaiveBayesJob WeightsMapper-Reducer	77.05	4 858.04 page/s	71.23	2 562.82 page/s

当作业的任务分解及调度结果确定后,结合表4中的数据,可通过基于平均功耗估算的能耗模型(式(18))对作业的能耗进行预测.对于WordCount,TeraSort,NutchIndex,*K*-means,PageRank这5种作业,由于作业阶段性任务较少,利用基于平均功耗估算的能耗模型进行能耗预测相比Bayes作业容易.因为Bayes作业多达10个阶段性任务,使得任务的分解及调度复杂度较高,即增加了利用平均功耗估算的能耗模型对Bayes作业进行能耗预测的难度;同时,较多的阶段性任务增大了预测的计算量.值得注意的是,表4表示的是同构集群环境(节点配置如表2所示)中的数据,在异构环境中,由于节点计算能力及能耗属性的不同,大大增加了构造表4数据的难度与工作量.即基于平均功耗估算的能耗模型适合对同构环境中阶段性任务较少(或任务分解与调度较简单)的作业进行能耗预测.由于基于平均功耗估算的能耗模型主要用于作业能耗的预测,并且依赖于作业的分解与调度结果,所以是将来研究节能的MapReduce作业调度系统的基础.

4.3 能耗模型运用及误差分析

本节对基于CPU利用率估算的能耗模型与基于平均功耗估算的能耗模型进行运用,选取了WordCount, TeraSort, Sort, NutchIndex, *K*-means, PageRank,Bayes七种作业作为研究对象,作业的配置如表3所示.在作业运行前根据作业的任务分解及调度结果,运用基于平均功耗估算的能耗模型对7种作业的能耗进行估算,各任务不同阶段的计算能力及平均功耗如表4所示.作业运行过程中利用能耗监测仪对各任务执行能耗(任务开始到任务结束时间段内节点能耗)进行监测,累加得到作业能耗实际值,并将此能耗值作为基准值.作业执行过程中还需对各任务的CPU利用率进行监测,作业执行

完毕后运用基于 CPU 利用率估算的能耗模型对各作业能耗进行计算(CPU 利用率与节点能耗值的对应关系如图 8 所示).

此外,由于 2 种能耗模型都是估算模型,难免存在一定程度的误差,表 5 对 2 种能耗模型的估算值、能耗测量值及估算误差进行了记录.

Table 5　The Usage Demonstration and Error Analysis for Energy Model

表 5　能耗模型估算值及误差

Job	Job Decomposited into Tasks	Measured Energy Consumption/J	CPU Utilization Estimation Model Computing Value/J	CPU Utilization Estimation Model Deviation/%	Average Energy Consumption Estimation Model Computing Value/J	Average Energy Consumption Estimation Model Deviation/%
WordCount	N/A	675 879	718 894	6.364	633 304.7	−6.299
TeraSort	N/A	491 630.5	517 729.3	5.31	443 444	−9.801
Sort	N/A	1 003 315	1 062 113.8	5.86	931 299.8	−7.18
NutchIndex	N/A	604 150	632 391.6	4.67	570 971.2	−5.49
K-means	Cluster Iterator	109 916	115 621.3	5.19	104 852.7	−4.606
	Cluster Classification	211 120	219 503.2	3.971	194 069.4	−8.076
PageRank	Stage1	286 072.5	292 842.9	2.367	273 113.5	−4.53
	Stage2	138 194	143 971.4	4.18	123 528	−10.613
Bayes	DocumentProcessor: DocumentTokeuizer	10 164	10 874.6	6.992	8 904	−12.395
	CollocDriver.generateCollocations	331 318.5	343 110	3.559	316 053	−4.607
	CollocDriver.computeNGrams	11 937.15	12 644.1	5.922	11 692.6	−2.049
	DictionaryVectorizer: MakeParitialVectors	12 939	14 103.7	9.002	12 311.5	−4.85
	ParticalVectorMerger: MergePartialVectors	2 143	2 315.9	8.068	1 785.7	−16.673
	VectorTfld Document Frequency Count	2 095	2 257.6	7.762	1 743.8	−16.766
	MakePratialVectors	2 167	2 466	13.8	1 839.3	−15.123
	ParticalVectorMerger: MergePartialVectors	2 024	2 325.5	14.894	1 821.9	−9.986
	TrainNaiveBayesJob IndexInstancesMapper-Reducer	2 236	2 496.9	11.669	2 000.5	−10.53
	TrainNaiveBayesJob WeightsMapper-Reducer	1 986	2 140.8	7.796	1 718.2	−13.485

总结表 5 中对 2 种能耗模型的估算值、能耗测量值及估算误差数据可以得到以下 4 个结论:

1) 基于 CPU 利用率估算能耗模型的估算值总是比实际测量值大. 如图 8(a)DataNode 节点 CPU 利用率与功耗之间的关系所示,大多数实验测量值都小于理论值,由于使用式(37)所确定的理论能耗值进行计算,所以造成估算值大于实际测量值.

2) 基于平均功耗估算能耗模型的估算值总是比实际测量值小. 基于平均功耗估算的能耗模型在任务分解及调度结果确定后,进行能耗估算时没有考虑到任务的执行失败问题. 而在任务的实际运行过程中存在不少任务的失败现象,在对失败任务重调度及重新执行过程中,消耗了额外的能耗,使得估算值小于实际测量值.

3) 基于 CPU 利用率估算的能耗模型估算值总体上较基于平均功耗估算的能耗模型估算值误差小. 这是由于前者是作业完成后的估算,计算过程只依赖于 CPU 利用率与能耗之间的对应关系,CPU 利用率与能耗估算值与实际值之间的误差决定了能耗模型的误差;而后者则是作业运行前的估算,首先需要对每个任务的完成时间进行估算,然后再基于估算的任务完成时间对能耗进行计算,2 次估算过程增大了误差.

4) 任务执行时间越短,任务数量越多,估算误差

越大.较短的任务执行时间使得能耗采样数据较小,导致能耗测量值本身的误差.

4.4 能耗优化方法实验

根据3.1节对能耗优化方法讨论的结果,本节分别对优化MapReduce作业执行能耗及减少MapReduce任务等待能耗2种能耗优化方法进行实验.由于提高MapReduce集群的能源利用效率主要涉及MapReduce作业及任务的调度问题,虽然不作为本文的研究重点,但却是将来的重要研究内容.

4.4.1 优化MapReduce作业执行能耗

采用低能耗高效率的硬件设备或体系结构,对MapReduce集群中的高能耗设备进行替换,该方法能够大幅度降低MapReduce作业的执行能耗.但是硬件的更换面临过高的成本问题,该方法已经在文献[59,61-62]中进行了大量的讨论,本文则不对此方法进行重复的实验.本实验将DataNode节点数设置为50,即设置Map与Reduce任务资源槽数各为50,将表3中配置的6种作业随机提交到集群时,可利用式(36)计算满足截止时间要求的Map与Reduce任务数.虽然式(36)中的不同作业不同阶段的完成时间可通过表4已知实验值确定,但多个作业同时提交到系统时,涉及到复杂的MapReduce作业及任务的调度问题,故本文首先考虑解决优化MapReduce单个作业执行能耗问题.对于节能的作业及任务调度研究,是下一步研究的主要内容.以TeraSort作业为例,将任务运行10次取平均数,其任务数与作业完成时间及能耗之间的关系如表6所示:

Table 6 The Relationship between Job Completion Time and Energy Consumption with Different Task Numbers

表6 任务数与作业完成时间及能耗之间的关系

Reduce Task Number	1 907.4 MB Data, 40 Map Tasks		2 861 MB Data, 60 Map Tasks		4 768.4 MB Data, 80 Map Tasks	
	Job Completion Time/s	Energy Consumption/J	Job Completion Time/s	Energy Consumption/J	Job Completion Time/s	Energy Consumption/J
1	410	57 970	613	86 788	1 031.3	145 445
2	235	60 286	354.5	90 759	536	142 192
4	193	79 980	249	106 564	388	168 671
6	188	104 143	246	138 340	348	199 784
8	154	107 452	229.5	160 185	326	229 573
10	151	125 160	213	176 569	307	254 562
12	122	116 057	222	212 813	255	240 493
14	120	128 634	216	234 658	256	270 613
16	106	124 662	228	278 350	248	289 148
18	118	154 782	153	196 427	252	322 909
20	94	129 295	159.5	224 561	239.3	330 356
22	92	135 914	153	231 181	235	348 891
24	86	133 928	147	236 642	226.7	358 058
26	91	153 622	152	263 783	231	391 588
28	82	142 533	142.5	258 983	218	386 456
30	88	165 702	146	283 311	221	416 741
32	85	167 025	138	278 345	212	416 244

如图10所示为作业任务数对完成时间及能耗的影响.一方面,理论上任务数越大作业完成时间越小,但实际上任务数目与任务完成时间之间并不呈线性关系,当任务数量增加到某临界点时,作业完成时间并不会因为任务数量的增加而减小(或减小效果不明显);另一方面,随着任务数量的不断增大,作业能耗不断地增加,即完成相同作业,任务数越大,完成该任务能耗越大.这是因为当任务数较多时,任务之间的数据传输增加,任务之间的关联变得复杂,任务协调成本增加,任务启动操作及任务完成后的清理动作增加,以上4方面都导致了能耗的增加.理论上,作业被分解的任务数越少,作业能耗利用率越高;但是作业具有截止时间QoS约束时,需满足作业完成时间deadline约束需求.

利用 3.3 节提出的截止时间约束下的最小资源槽 *slot* 分配方法计算出的作业分解方案,能够在满足作业完成时间 deadline 约束前提条件下最小化作业执行能耗. 所以,截止时间约束下的最小资源槽 *slot* 分配方法是一种适应节能的作业的任务分解模型.

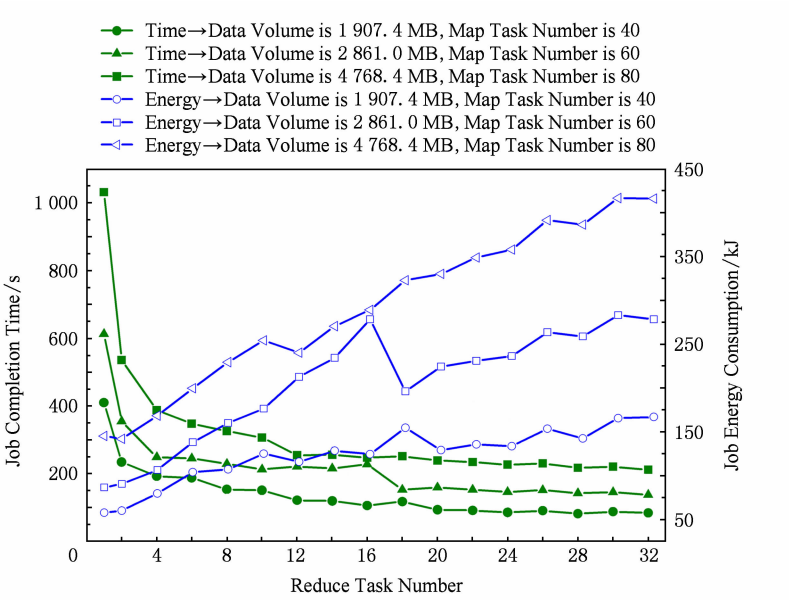


Fig. 10　The comparison of job completion time and energy consumption with different task numbers.

图 10　作业任务数对完成时间及能耗的影响

此外,实验过程中我们还发现,当 DataNode 节点 CPU 温度较高时,完成相同的任务,需要消耗更多的能耗. 图 8(b)所示当 CPU 温度较高时(70～88℃之间)DataNode 节点 CPU 利用率与功耗之间的关系. 即在进行任务与资源之间的绑定过程中,需要考虑 CPU 的温度.

4.4.2　减少 MapReduce 任务等待能耗

为了构造异构集群实验环境,我们向同构集群中添加了 1 台性能较差的 DataNode 节点. 实验中异构集群由 9 台同构 DataNode 节点(配置如表 2)及 1 台配置为单核 Intel Pentium 4 1.5 GHz、512 MB 内存及 40 GB 硬盘的低配节点组成. 实验准备阶段,首先对新加入的异构节点计算能力进行测试. 测试的方法是在异构节点上运行作业,记录作业完成时间与已有节点进行对比,最后得到异构节点是原节点性能的 0.228 倍. 数据布局阶段,首先按照机架感知的数据块布局策略将 7 种不同的作业数据进行分布存储,数据块大小相同;再次,按照 3.2 节中异构环境下数据的放置策略和节点的计算能力进行数据布局,异构节点数据量是原节点的 0.228 倍. 2 种布局策略下 6 种作业(作业配置参数如表 3 所示)的能耗及完成时间(实验 10 次平均值)对比如表 7 所示:

Table 7　Comparison of Job Completion Time and Energy Consumption with Different Data Layout Strategies

表 7　不同数据布局策略下作业完成时间及能耗对比

Job	Isomorphic Environment Running Time/s	Heterogeneous Environment Running Time/s		Energy Consumption under Homogeneous Environment/J	Energy Consumption under Heterogeneous Environment/J	
		Original Layout Strategy	Adapt the Computing Ability Layout Strategy		Original Layout Strategy	Adapt the Computing Ability Layout Strategy
WordCount	890	919	901	675 879	704 571	694 812
TeraSort	603	629	614	491 630	520 980	507 290
NutchIndex	786	824	794	604 150	644 972	620 953
K-means	435	451	442	321 036	338 637	326 912
PageRank	543	623	580	424 266	468 368	442 625
Bayes	484	558	502	373 419	433 896	389 573

如表 7 数据及图 11 所示,与异构环境下原数据布局策略(机架感知的数据布局策略)相比较,适应异构集群的数据布局策略(按计算力布局策略)分别提高 WordCount,TeraSort,NutchIndex,*K*-means,PageRank,Bayes 作业完成时间 18 s,15 s,30 s,9 s,43 s,56 s,提升作业完成时间 1.959%,2.385%,3.64%,1.99%,6.902%,10.036%;分别节能 9 759 J,13 690 J,24 019 J,11 725 J,25 743 J,44 323 J,节能率为 1.385%,2.628%,3.724%,3.462%,5.496%,10.215%.可以发现适应异构集群的数据布局策略对于 WordCount,TeraSort,NutchIndex,*K*-means 四种作业完成时间及节能效率提升并不明显,而 PageRank 及 Bayes 有较为明显的提升,这是因为 PageRank 与 Bayes 作业 Reduce 任务需要等待所有 Map 任务完成才能开始,造成等待时延较长;而其他作业完成部分 Map 任务 Reduce 任务就能开始,等待时延较短.所以,利用 3.2 节中提出的数据布局策略,能够达到减小作业运行时间并提高 MapReduce 作业能耗利用率的目的.

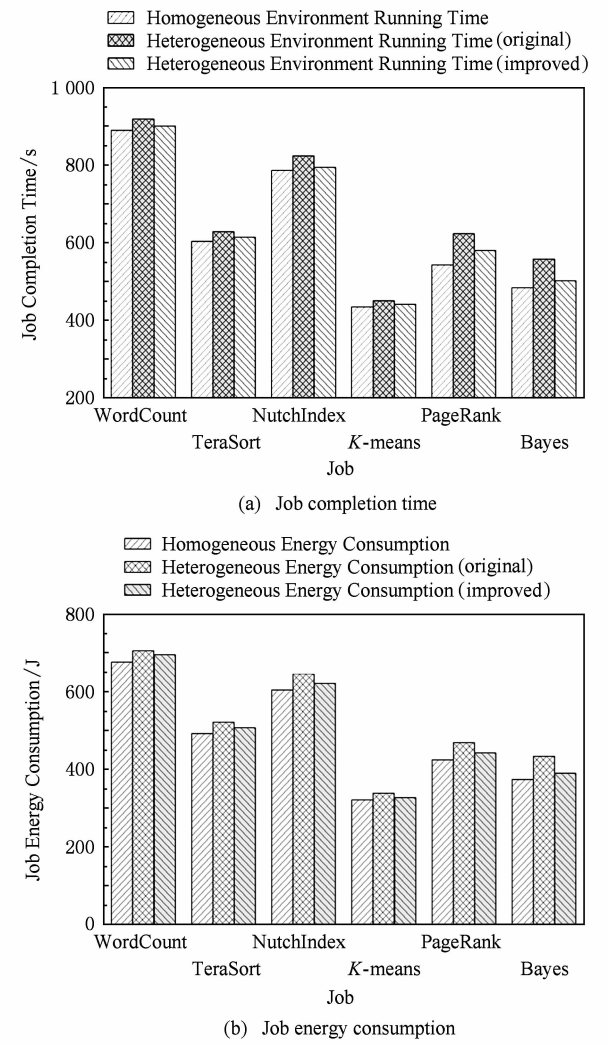


Fig. 11 The comparison of job completion time and energy consumption.

图 11 作业运行时间及能耗对比

5 结 论

首先,本文对 MapReduce 系统模型进行了分析,提出 3 种任务级的能耗模型:基于 CPU 利用率估算的能耗模型、基于主要部件能耗累加的能耗模型及基于平均功耗估算的能耗模型,并在这 3 种模型基础上得到了 MapReduce 作业能耗模型.其中,基于 CPU 利用率估算的能耗模型用于作业完成后的能耗的估算,是将来开发 Hadoop 作业能耗监控及优化软件的理论基础;基于平均功耗估算的能耗模型用于作业运行前的能耗预测,是将来研究节能的 MapReduce 作业调度系统的基础;基于主要部件能耗累加的能耗模型参数较多,能够清晰地表达作业执行过程中各部件的能耗行为.其次,对 MapReduce 作业能耗优化进行了分析,提出从单个 MapReduce 作业的执行能耗与提高 MapReduce 集群能源效率 2 个层面,以及优化 MapReduce 作业执行能耗、减少 MapReduce 任务等待能耗与提高 MapReduce 集群能源利用效率 3 个方向对 MapReduce 进行能耗优化,并提出异构环境下数据的放置策略及截止时间约束下的最小资源槽 *slot* 分配方法分别从减少 MapReduce 任务等待能耗及 MapReduce 作业执行能耗 2 种方法提高能耗利用率.最后,通过大量的实验及能耗数据分析,证明本文所提能耗模型及能耗优化方法的正确性.下一步工作主要集中在以下 4 个方面:

1) 对基于主要部件能耗累加的能耗模型进行研究,对各部件的能耗参数进行确定,实现对作业执行过程中各部件能耗的精细化监控,并在本文的基础上研究 MapReduce 执行过程中网络传输的能耗行为及优化方法.

2) 在基于 CPU 利用率估算的能耗模型基础上,研究并开发 Hadoop 作业能耗监控及优化软件,实现对 MapReduce 作业能耗的自动化计算.

3) 基于本文提出的截止时间约束下的最小资源槽 *slot* 分配方法及基于平均功耗估算的能耗模型,研究节能的作业及任务调度算法,从提高 MapReduce 集群能源效率角度对能耗进行优化.

4) 设计温度感知的任务资源映射算法,将节点 CPU 温度状态加入到任务到资源的映射模型中,实现节能的任务资源映射模型.

参 考 文 献

[1] Meng Xiaofeng, Ci Xiang. Big data management: Concepts, techniques and challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-149 (in Chinese)
(孟小峰, 慈祥. 大数据管理: 概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-149)

[2] Gantz J, Chute C, Manfrediz A, et al. The diverse and exploding digital universe: An updated forecast of worldwide information growth through 2011 [EB/OL]. 2012[2015-05-25]. http://wwww. ifap. ru/library/book268. pdf

[3] Global Action Plan International. Global action plan, an inefficient truth [EB/OL]. 2007 [2011-02-12]. http://globalactionplan. org. uk

[4] Times N Y. Power, pollution and the Internet [EB/OL]. 2012 [2015-05-20]. http://www. nytimes. com/2012/09/23/technology/data-ceneters-waste-vast-amounts-of-energy-belying-industry-image. html

[5] Barroso L A, Hlzle U. The datacenter as a computer: An introduction to the design of warehouse-scale machines [R]. San Francisco, CA: Morgan & Claypool Publishers, 2009

[6] Borthaku D. The Hadoop distributed file system: Architecture and design [EB/OL]. (2007-07-01) [2015-02-12]. http://hadoop. apache. org/docs/r1. 2. 1/hdfs _ design. html

[7] Ghemawat S, Gobioff H, Leung S T. The Google file system [C] //Proc of the 19th ACM Symp on Operating System Principles. New York: ACM, 2003: 29-43

[8] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [C] //Proc of the Conf on Operating System Design and Implementation (OSDI). New York: ACM, 2004: 137-150

[9] Wang Peng, Meng Dan, Zhan Jianfeng, et al. Review of programming models for data-intensive computing [J]. Journal of Computer Research and Development, 2010, 47(11): 1993-2002 (in Chinese)
(王鹏, 孟丹, 詹剑锋, 等. 数据密集型计算编程模型研究进展[J]. 计算机研究与发展, 2010, 47(11): 1993-2002)

[10] Li D, Wang J E. Energy efficient redundant and inexpensive disk array [C] //Proc of the ACM SIGOPS European Workshop. New York: ACM, 2004: 29-35

[11] Liao X, Jin H, Liu H. Towards a green cluster through dynamic remapping of virtual machines [J]. Future Generation Computer Systems, 2012, 28(2): 469-477

[12] Liao Bin, Yu Jiong, Zhang Tao, et al. Energy-efficient algorithms for distributed file system HDFS [J]. Chinese Journal of Computers, 2013, 36(5): 1047-1064 (in Chinese)
(廖彬, 于炯, 张陶, 等. 基于分布式文件系统 HDFS 的节能算法[J]. 计算机学报, 2013, 36(5): 1047-1064)

[13] Albers S. Energy-efficient algorithms [J]. Communications of the ACM, 2010, 53(5): 86-96

[14] Wierman A, Andrew L L, Tang A. Power-aware speed scaling in processor sharing systems [C] //Proc of the 28th IEEE Int Conf on Computer Communications (INFOCOM 2009). Piscataway, NJ: IEEE, 2009: 2007-2015

[15] Andrew L L, Lin M, Wierman A. Optimality, fairness, and robustness in speed scaling designs [C] //Proc of ACM Int Conf on Measurement and Modeling of Int Computer Systems (SIGMETRICS 2010). New York: ACM, 2010: 37-48

[16] Neugebauer R, McAuley D. Energy is just another resource: Energy accounting and energy pricing in the nemesis OS [C] //Proc of the 8th IEEE Workshop on Hot Topics in Operating Systems. Piscataway, NJ: IEEE, 2001: 59-64

[17] Flinn J, Satyanarayanan M. Managing battery lifetime with energy-aware adaptation [J]. ACM Trans on Computer Systems, 2004, 22(2): 179-182

[18] Meisner D, Gold B T, Wenisch T F. PowerNap: Eliminating server idle power [J]. ACM SIGPLAN Notices, 2009, 44(3): 205-216

[19] Ye K, Jiang X, Ye D, et al. Two optimization mechanisms to improve the isolation property of server consolidation in virtualized multi-core server [C] //Proc of the 12th IEEE Int Conf on High Performance Computing and Communications. Piscataway, NJ: IEEE, 2010: 281-288

[20] Choi J, Govindan S, Jeong J, et al. Power consumption prediction and power-aware packing in consolidated environments [J]. IEEE Trans on Computers, 2010, 59(12): 1640-1654

[21] Ye K, Jiang X, Huang D, et al. Live migration of multiple virtual machines with resource reservation in cloud computing environments [C] //Proc of the 4th IEEE Int Conf on Cloud Computing. Piscataway, NJ: IEEE, 2011: 267-274

[22] Liao X, Jin H, Liu H. Towards a green cluster through dynamic remapping of virtual machines [J]. Future Generation Computer Systems, 2012, 28(2): 469-477

[23] Jang J W, Jeon M, Kim H S, et al. Energy reduction in consolidated servers through memory-aware virtual machine scheduling [J]. IEEE Trans on Computers, 2011, 99(1): 552-564

[24] Wang X, Wang Y. Coordinating power control and performance management for virtualized server cluster [J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22(2): 245-259

[25] Wang Y, Wang X, Chen M, et al. Partic: Power-aware response time control for virtualized Web servers [J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22(2): 323-336

[26] Dasgupta G, Sharma A, Verma A, et al. Workload management for power efficiency in virtualized data-centers [J]. Communications of the ACM, 2011, 54(7): 131-141

[27] Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing [J]. Cluster Computing, 2009, 12(1): 1-15

[28] Garg S K, Yeo C S, Anandasivam A, et al. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers [J]. Journal of Parallel and Distributed Computing, 2010, 71(6): 732-749

[29] Kusic D, Kephart J O, Hanson J E, et al. Power and performance management of virtualized computing environments via lookahead control [J]. Cluster Computing, 2009, 12(1): 1-15

[30] Song Y, Wang H, Li Y, et al. Multi-tiered on-demand resource scheduling for VM-based data center [C] //Proc of the 9th IEEE/ACM Int Symp on Cluster Computing and the Grid (CCGrid 2009). Piscataway, NJ: IEEE, 2009: 148-155

[31] Gmach D, Rolia J, Cherkasova L, et al. Resource pool management: Reactive versus proactive or let's be friends [J]. Computer Networks, 2009, 53(17): 2905-2922

[32] Buyya R, Beloglazov A, Abawajy J. Energy-Efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges [C] //Proc of the 2010 Int Conf on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010). New York: ACM, 2010

[33] Kim K H, Beloglazov A, Buyya R. Power-aware provisioning of cloud resources for real-time services [C] //Proc of the 7th Int Workshop on Middleware for Grids. New York: ACM, 2009: 1-6

[34] Wang Yijie, Sun Weidong, Zhou Song, et al. Key technologies of distributed storage for cloud computing [J]. Journal of Software, 2012, 23(4): 962-986 (in Chinese)
(王意洁, 孙伟东, 周松, 等. 云计算环境下分布存储关键技术[J]. 软件学报, 2012, 23(4): 962-986)

[35] Greenan K M, Long D D E, Miller E L, et al. A spin-up saved is energy earned: Achieving power-efficient, erasure-coded storage [C] //Proc of the HotDep 2008. Berkeley, CA: USENIX Association, 2008

[36] Weddle C, Oldham M, Qian J, et al. A gear-shifting power-aware raid [J]. ACM Trans on Storage, 2007, 3(3): 1553-1569

[37] Li D, Wang J. Conserving energy in conventional disk based RAID systems [C] //Proc of the 3rd Int Workshop on Storage Network Architecture and Parallel I/Os (SNAPI 2005). Piscataway, NJ: IEEE, 2005: 65-72

[38] Yao X, Wang J. Rimac: A novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems [C] //Proc of the EuroSys. New York: ACM, 2006: 249-262

[39] Pinheiro E, Bianchini R, Dubnicki C. Exploiting redundancy to conserve energy in storage systems [C] //Proc of the SIGMetrics Performance 2006. New York: ACM, 2006: 15-26

[40] Narayanan D, Donnelly A, Rowstron A. Write off-loading: Practical power management for enterprise storage [J]. ACM Trans on Storage, 2008, 4(3): 253-267

[41] Storer M, Greenan K, Miller E, et al. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage [C] //Proc of the FAST 2008. New York: ACM, 2008: 1-16

[42] Zhu Q, Chen Z, Tan L, et al. Hibernator: Helping disk arrays sleep through the winter [C] //Proc of the 20th ACM Symp on Operating Systems Principles (SOSP). New York: ACM, 2005: 177-190

[43] Vasic N, Barisits M, Salzgeber V. Making cluster applications energy-aware [C] //Proc of the ACDC 2009. New York: ACM, 2009: 37-42

[44] Zhu Q, David F M, Devaraj C F, et al. Reducing energy consumption of disk storage using power-aware cache management [C] //Proc of the HPCA 2004. Piscataway, NJ: IEEE, 2004: 118-129

[45] Chen Y, Keys L, Katz R H. Towards energy efficient MapReduce [R]. Berkeley, CA: EECS Department, University of California, 2009

[46] Leverich J, Kozyrakis C. On the energy (in) efficiency of Hadoop clusters [J]. ACM SIGOPS Operating Systems Review, 2010, 44 (1): 61-65

[47] Kaushik R T, Bhandarkar M. GreenHDFS: Towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster [C] //Proc of the 2010 Int Conf on Power Aware Computing and Systems. Piscataway, NJ: IEEE, 2010: 1-9

[48] Kaushik R T, Bhandarkar M, Nahrstedt K. Evaluation and analysis of GreenHDFS: A self-adaptive, energy conserving variant of the Hadoop distributed file system [C] //Proc of the 2nd IEEE Int Conf on Cloud Computing Technology and Science. Piscataway, NJ: IEEE, 2010: 274-287

[49] Lang W, Patel J M. Energy management for MapReduce clusters [J]. Proceedings of the VLDB Endowment, 2010, 3 (12): 129-139

[50] Wirtz T, Ge R. Improving MapReduce energy efficiency for computation intensive workloads [C] //Proc of Int Green Computing Conf and Workshops (IGCC). Piscataway, NJ: IEEE, 2011: 1-8

[51] Goiri í, Le K, Nguyen T D, et al. GreenHadoop: Leveraging green energy in data-processing frameworks [C] //Proc of the 7th ACM European Conf on Computer Systems. New York: ACM, 2012: 57-70

[52] Cardosa M, Singh A, Pucha H, et al. Exploiting spatio-temporal tradeoffs for energy efficient MapReduce in the cloud [R]. Minneapolis, Minnesota: Department of Computer Science and Engineering, University of Minnesota, 2010

[53] Chen Y, Ganapathi A, Katz R H. To compress or not to compress-compute vs IO tradeoffs for MapReduce energy efficiency [C] //Proc of the 1st ACM SIGCOMM Workshop on Green Networking. New York: ACM, 2010: 23-28

[54] Song Jie, Li Tiantian, Zhu Zhiliang, et al. Benchmarking and analyzing the energy consumption of cloud data management system [J]. Chinese Journal of Computers, 2013, 36(7): 1485-1499 (in Chinese)
(宋杰, 李甜甜, 朱志良, 等. 云数据管理系统能耗基准测试与分析[J]. 计算机学报, 2013, 36(7): 1485-1499)

[55] Lin Chuang, Tian Yuan, Yao Min. Green network and green evaluation: Mechanism, modeling and evaluation [J]. Chinese Journal of Computers, 2011, 34(4): 593-612 (in Chinese)
(林闯, 田源, 姚敏. 绿色网络和绿色评价: 节能机制、模型和评价[J]. 计算机学报, 2011, 34(4): 593-612)

[56] Andrews M, Anta A F, Zhang L, et al. Routing for energy minimization in the speed scaling model [C] //Proc of the 29th IEEE Int Conf on Computer Communications (INFOCOM 2010). Piscataway, NJ: IEEE, 2010: 1-9

[57] Barroso L A, Holzle U. The case for energy-proportional computing [J]. Computer, 2007, 40(12): 33-37

[58] Kansal A, Zhao F, Liu J, et al. Vitrual machine power metering and provisioning [C] //Proc of the 1st ACM Symp on Cloud Computing. New York: ACM, 2010: 39-50

[59] Harnik D, Naor D, Segall I. Low power mode in cloud storage systems [C] //Proc of the IPDPS 2009. Piscataway, NJ: IEEE, 2009: 1-8

[60] Bao Y, Chen M, Ruan Y, et al. HMTT: A platform independent full-system memory trace monitoring system [J]. ACM SIGMETRICS Performance Evaluation Review, 2008, 36(1): 229-240

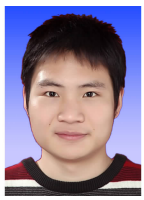
[61] Vasudevan V, Franklin J, Andersen D. FAWN damentally power-efficient clusters [C] //Proc of the HotOS XII. Berkeley, CA: USENIX Association, 2009

[62] Kim H S, Shin D I, Yu Y J, et al. Towards energy proportional cloud for data processing frameworks [C] //Proc of the 1st USENIX Conf on Sustainable Information Technology. Berkeley, CA: USENIX Association, 2010: 1-8

[63] Liao Bin, Yu Jiong, Sun Hua, et al. Energy-efficient algorithms for distributed storage system based on data storage structure reconfiguration [J]. Journal of Computer Research and Development, 2013: 50(1): 3-18 (in Chinese)
(廖彬, 于炯, 孙华, 等. 基于存储结构重配置的分布式存储系统节能算法[J]. 计算机研究与发展, 2013, 50(1): 3-18)

[64] Liao B, Yu J, Sun H, et al. A QoS-aware dynamic data replica deletion strategy for distributed storage systems under cloud computing environments [C] //Proc of the Cloud and Green Computing. Piscataway, NJ: IEEE, 2012: 219-225

[65] Liao Bin, Yu Jiong, Qian Yurong, et al. The node failure recovery algorithm for distributed file system based on measurement of data availability [J]. Computer Sicence, 2013, 40(1): 144-149 (in Chinese)
(廖彬, 于炯, 钱育蓉, 等. 基于可用性度量的分布式文件系统节点失效恢复算法[J]. 计算机科学, 2013, 40(1): 144-149)

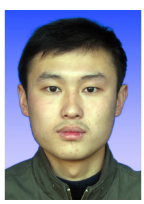


Liao Bin, born in 1986. PhD and associate professor in the School of Statistics and Information, Xinjiang University of Finance and Economics. His main research interests include database theory and technology, big data and green computing, etc.

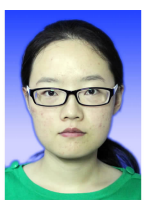
Zhang Tao, born in 1988. PhD candidate in the School of Information Science and Engineering, Xinjiang University. Her main research interests include big data and cloud computing, etc.

Yu Jiong, born in 1964. Professor and PhD supervisor in computer science at the School of Software, Xinjiang University. His main research interests include on grid computing, parallel computing, etc.

Yin Lutong, born in 1992. MSc candidate in the School of Software, Xinjiang University. His main research interests include cloud and green computing, etc.



Guo Gang, born in 1990. MSc candidate in the School of Software, Xinjiang University. His main research interests include data mining and green computing, etc.



Guo Binglei, born in 1991. PhD candidate in the School of Information Science and Engineering, Xinjiang University. Her main research interests include cloud and green computing, etc.