基于分布式文件系统 HDFS 的节能算法

廖 彬[1),2)] 于 炯[1),2)] 张 陶[2)] 杨兴耀[2)]

[1)](新疆大学软件学院 乌鲁木齐 830008)

[2)](新疆大学信息科学与工程学院 乌鲁木齐 830046)

摘 要 与传统数据中心节能算法不同,MapReduce 计算任务的数据依赖性使得设计 HDFS(Hadoop Distributed File System)节能算法时必须保证集群中所有数据块的可用性,即任意数据块或其副本中的至少一块处于活动状态.根据 HDFS 集群结构与数据块存储等特点建立了 DataNode 节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵与数据块状态矩阵,为后续研究建立了基础模型.结合数据块状态矩阵与数据块可用性之间的关系设计了 DataNode 节点休眠验证算法.概率分析了由于机架感知的存储策略带来数据块分布的随机性,使得在不改变数据块存储结构与存储策略的情况下并不能通过休眠 DataNode 节点达到节能的目的.进而设计了数据块存储结构配置节能算法与基于对称数据块存储策略下的节能算法,分别从改变数据块的存储结构与存储策略两方面对 HDFS 进行节能改进.实验结果表明:两种节能算法都能解决 HDFS 集群的能耗低利用率问题,并且集群负载越低节能效率越高.

关键词 云计算;分布式文件系统;节能计算;副本策略;绿色计算

中图法分类号 TP311 **DOI 号** 10.3724/SP.J.1016.2013.01047

Energy-Efficient Algorithms for Distributed File System HDFS

LIAO Bin[1),2)] YU Jiong[1),2)] ZHANG Tao[2)] YANG Xing-Yao[2)]

[1)](*School of Software, Xinjiang University, Urumqi* 830008)

[2)](*School of Information Science and Engineering, Xinjiang University, Urumqi* 830046)

Abstract Different from traditional energy-efficiency algorithms in data center, data-dependent computing mechanism of MapReduce makes energy-efficiency algorithm in HDFS (Hadoop Distributed File System) must ensure the availability of all data blocks in cluster, that means at least one data block or its replica should in active state. DataNode matrix, DataNode status matrix, file block matrix, block storage matrix and block status matrix are created based on the HDFS cluster structure and block storage mechanism etc., and those matrixes established foundational models for further research. Based on the relationship between the availability of data blocks and its block status matrix, algorithm for make sure if a DataNode can sleep is designed. Probability analysis makes out that it is difficult to save energy in HDFS cluster without changing the data block's storage structure or replica placement mechanism because randomness distribution of the data block result from rack-awareness replica placement mechanism. So we design data block storage structure configuration energy-efficiency algorithm and energy-efficiency algorithm under symmetric replica placement mechanism to save the energy consumption of the HDFS cluster from changing and improving of block's storage structure and replica placement mechanism respectively. Mathematical analysis and experiments prove that two energy-efficiency algorithm solve HDFS

收稿日期:2011-12-03;最终修改稿收到日期:2012-08-26.本课题得到国家自然科学基金(60863003,61063042)和新疆维吾尔族自治区自然科学基金(2011211A011)资助.**廖 彬**,男,1986 年生,博士研究生,主要研究方向为数据库技术、网格与云计算.E-mail:liaobin665@163.com.**于 炯**,男,1964 年生,博士,教授,博士生导师,主要研究领域为网络安全、网格与分布式计算.**张 陶**,女,1988 年生,硕士研究生,主要研究方向为分布式计算、网格计算.**杨兴耀**,男,1984 年生,博士研究生,主要研究方向为分布式计算、网格计算.

cluster's high energy consumption but low-efficiency problem, the lower utilization of the cluster the more energy consumption it can save.

Keywords cloud computing; distributed file system; energy-efficient computing; replica placement strategy; green computing

1 引 言

低碳节能已经成为全球热点问题之一,而信息与通信技术行业的高速发展也带来了高能耗、高二氧化碳排放量的问题.据统计①,目前 IT 领域的二氧化碳排放量占全球的 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 8680 亿度电,占全球总耗电量的 5.3%.根据文献[1]的预测,到 2025 年 IT 行业的平均能耗将达到 2006 年的 5 倍,而网络设备的能耗更会达到 13 倍.与此同时,Barroso 等人在文献[2]中对 Google 内部 5000 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,完全没有达到高效利用的要求.造成 Google 服务器利用率低的主要原因是传统的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的同时没有考虑到系统负载率与能耗利用率之间的关系,导致用户的服务请求被分发到了过量的而不是适量的服务器上,由此造成了大量电能的浪费.通过图 1 可以看出服务器在负载很低(小于 10%)的情况下电能消耗也超过了峰值能耗的 50%,并且能耗利用率随着系统负载的增加而增加.由此可见,IT 设备的高能耗、高二氧化碳排放量与能耗的低利用率问题已经成为信息与通信技术行业亟待解决的问题.

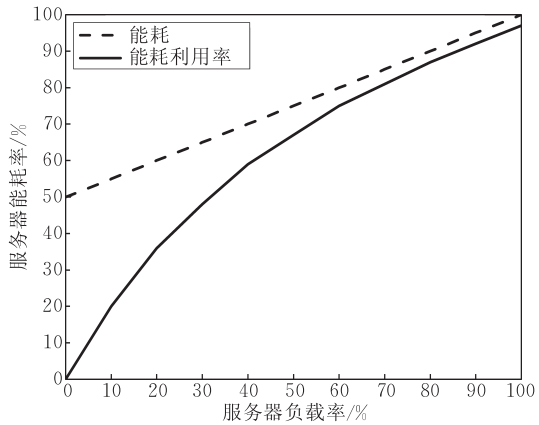


图 1 服务器能耗利用率与负载率之间的关系[2]

Hadoop② 作为新的分布式存储与计算架构,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界事实上的海量数据并行处理标准.Hadoop 作为分布式系统基础架构,用户可以在不了解分布式系统底层细节的情况下,充分利用集群的高速运算和存储能力,开发分布式应用程序.Hadoop 参考 Google 的分布式文件系统 GFS(Google File System)[3],实现了分布式文件系统 HDFS;参考 MapReduce[4] 计算模型实现了自己的分布式计算框架;参考BigTable[5] 实现了分布式数据库 HBase.虽然 Hadoop 拥有诸多优点,但是由于 Hadoop 底层的分布式文件系统 HDFS 基于机架感知的数据块存储策略使得数据块在集群中的分布具有随机性,并且 MapReduce 的移动计算理念使得计算任务与数据块依赖紧密,这使得系统必须保证所有数据块的可用性.Hadoop 通过副本策略与节点失效处理等等方法保证数据块可用性的同时并没有考虑集群负载率与系统能耗之间的关系,即使在 Hadoop 集群利用率很低的情况下,集群中所有的 DataNode 节点都保持活动状态以保证系统中数据块的可用性,其高能耗低效率的情况与 Google 服务器集群类似,并由此造成了大量电能的浪费.为解决 HDFS 分布式集群高能耗低效率的问题,本文主要做了以下工作:

(1)通过分析 HDFS 集群节点结构、文件分块与存储机制建立了 DataNode 节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵与数据块状态矩阵.用矩阵表达的集群结构、节点状态、文件分块、数据块存储、数据块状态规范了 HDFS 集群的建模并为后续研究打下了基础.

(2)通过休眠闲置节点进行节能的方法在 HDFS 集群中受到数据块可用性的挑战.为了验证特定 DataNode 节点的休眠是否影响数据块的可用

① Global Action Plan. An Inefficient Truth. Global Action Plan Report. http://globalactionplan.org.uk, Dec. 2007

② Borthaku D. The Hadoop Distributed File System: Architecture and Design. http://hadoop.apache.org/common/docs/r0.18.2/hdfs_design.pdf, 2007-07-01

性，结合数据块状态矩阵与数据块可用性之间的关系设计了 DataNode 节点可休眠验证算法. 通过概率分析证明了在不改变数据块存储结构与存储策略的情况下并不能通过休眠 DataNode 节点达到节能的目的.

(3) 在原机架感知的存储策略基础上设计了数据块存储结构配置节能算法与 RACK 区域划分算法. RACK 区域划分算法将 RACK 划分成 Active-Zone 与 Sleep-Zone 两个区域，根据数据块的访问规律将数据块存储在不同的区域中，在保证数据块可用性的前提下通过休眠 Sleep-Zone 区域中 DataNode 节点达到节能的目的.

(4) 设计了对称数据块存储策略，并设计了该存储策略下的节能算法. 在不影响原 HDFS 可靠性、高效性与可扩展性的前提下，对称数据块存储策略使得具有相同存储结构的 DataNode 节点之间能够实现任务的转移与替换，算法通过任务调度产生并休眠空闲节点从而达到节能的目的.

本文第 2 节对相关工作进行介绍；第 3 节定义 DataNode 节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵与数据块状态矩阵的建立并给出示例；第 4 节提出 DataNode 节点可休眠验证算法、数据块存储结构配置节能算法、RACK 区域划分算法与基于对称数据块存储策略下的节能算法；第 5 节对相关算法建立数学分析模型；第 6 节通过模拟实验验证算法的有效性；最后一节对全文进行总结并提出未来研究工作的方向.

2　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来保障 QoS 与系统可靠性[6]，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. 学术与工业界分别从硬件、操作系统、虚拟机、数据中心 4 个层次去解决 IT 系统的能耗问题. 硬件层次上主要采用 DCD[7-10] (Dynamic Component Deactivation) 与 DPS[11-14] (Dynamic Performance Scaling) 两种节能技术，其中 DCD 通过在一段时间内关闭或休眠系统硬件的某些部件，而 DPS 通过调节适应于当前负载的 CPU 频率与电压从而达到节能的目的. 文献[7-10]研究了 DCD 节能技术将关闭或休眠后的部件重新启动的时间延迟影响用户 QoS 与休眠部件初始化时的高能耗等关键问题. DPS 通过降低 CPU 工作频率与电压进行节能的同时也带来了 CPU 性能降低的问题，由此文献[11-14]对 CPU 能耗与性能之间的平衡点问题进行了相关研究. 操作系统层面上，文献[15-18]研究了 Nemesis OS、Linux、ECOsystem 等系统采用系统资源管理、任务调度与适应、硬件节能技术等方法对操作系统进行节能处理的相关问题. 虚拟机层面上，Xen、VMware、KVM 等虚拟机 VMM (Virtual Machine Monitor) 利用 DPS 或者 DCD 等技术控制系统资源的使用来达到节能的目的. 文献[18-29]对数据中心层面上的节能技术进行了大量研究，数据中心层面上的节能技术通常采用合并任务的方法，即在满足用户 QoS 约束的前提下将任务分配到较小的资源集上，将空闲下来的节点资源休眠以达到节能的目的. 本文即是基于数据中心的节能思想，并假设所有的 DataNode 节点都具有 DPS 能力以便根据当前负载动态调节服务器能耗，在满足 HDFS 集群数据块可用性的前提下，通过休眠空闲 DataNode 节点达到节能的目的.

现阶段针对 MapReduce、HDFS 节能方面的研究较少，文献[30]对基于 MapReduce 与 HDFS 框架的数据中心能耗进行了测量，提出了存在的能耗问题，并通过优化系统配置参数来提高能源利用率. 文献[31]提出了数据块子集 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[32]设计了 HDFS 集群数据块配置算法，该算法根据当前 HDFS 集群的工作负载动态配置数据块的存放. 当节点负载达到设定阈值时，通过算法自动地打开与关闭某些 DataNode 节点以达到节能的目的. 文献[33-34]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，根据数据块的访问次数将其放在 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.

本文与以上工作不同之处在于通过建立分析矩阵对 HDFS 集群结构、节点状态、文件分块、数据块存储、数据块状态进行建模，为后续研究建立了基础. 本文设计的节能算法以保证数据块的可用性为前提，而已有研究则很少考虑到 HDFS 集群这一特点. 数据块存储结构配置节能算法与基于对称数据块存储

策略下的节能算法分别从数据块存储结构与数据块存储策略两方面对 HDFS 集群能耗进行改进. 其中数据块存储结构配置节能算法结合了文献[31-33]的思想,首先 RACK 区域划分算法将 RACK 划分成 Active-Zone 与 Sleep-Zone 两个区域,并通过休眠 Sleep-Zone 区域中 DataNode 节点达到节能的目的. 而对称数据块存储策略下的节能算法首先重新设计了数据块的存储策略,在保证 HDFS 可靠性、高效性与可扩展性等优点的同时,使得 DataNode 节点的休眠摆脱了数据块可用性的约束,从而达到通过休眠空闲节点实现节能目标.

3 HDFS 集群节能问题建模及示例

3.1 节能问题建模

本节将对 HDFS 集群节能问题的相关概念进行建模,具体包括 DataNode 节点矩阵、DataNode 节点状态矩阵、文件分块矩阵、数据块存储矩阵、数据块状态矩阵、HDFS 集群节能问题定义等等.

HDFS 集群一般由多个机架 RACK 组成,而一个 RACK 内部又由多个服务器组成,并且 HDFS 集群通常由一个 NameNode 和多个 DataNode 节点构成,因文本考虑的是将 DataNode 节点休眠以达到节能的目的,所以不考虑 NameNode 节点的节能问题.

定义 1(DataNode 节点矩阵). 设某个 HDFS 集群由集合 $Cluster=\{\langle rack_1,s_1\rangle,\langle rack_2,s_2\rangle,\cdots,\langle rack_i,s_i\rangle\}$ 组成,其中元素 $\langle rack_1,s_1\rangle$ 表示编号为 $rack_1$ 的 RACK 机架中有 s_1 台 DataNode 服务器,该集群由 i 个 RACK 组成,即 $|Cluster|=i$. 用 dn 表示 DataNode 节点服务器,那么可将 HDFS 集群中的所有 DataNode 节点表示为矩阵 $\boldsymbol{c}_{sm\times i}$:

$$\boldsymbol{c}_{sm\times i}=\begin{bmatrix} dn_{11} & dn_{12} & \cdots & dn_{1i} \\ dn_{21} & dn_{22} & \cdots & dn_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ dn_{sm1} & dn_{sm2} & \cdots & dn_{smi} \end{bmatrix},$$

其中矩阵 $\boldsymbol{c}_{sm\times i}$ 中的列表示编号为 $rack_i$ 的 RACK 中的 s_i 个 DataNode 服务器,其中 sm 表示 RACK 中节点数量集合 $\{s_1,s_2,\cdots,s_i\}$ 中的最大值. 设集群中 DataNode 节点的数量为 k,那么 $k=\sum s_i$,而 DataNode 节点矩阵 $\boldsymbol{c}_{sm\times i}$ 可以表示 $sm\times i$ 个元素,当 $sm\times i>\sum s_i$ 时,用 $sm\times i-\sum s_i$ 个 0 填充 $\boldsymbol{c}_{sm\times i}$ 矩阵,表示 RACK 中该位置没有 DataNode 节点.

定义 2(DataNode 节点状态矩阵). HDFS 集群中的 DataNode 节点可能存在多种状态,设状态标识集合为 $state=\{0,1,2,3\}$,其中 0 表示该位置没有服务器,而 1、2、3 分别表示 DataNode 节点处于活动、休眠、宕机状态. 宕机有可能是因为系统故障、电源中断、硬件故障等等原因造成;休眠与宕机的共同点都是此时 DataNode 节点不可用,但是休眠可以通过唤醒机制将 DataNode 节点转化为活动状态. 在 DataNode 节点矩阵 $\boldsymbol{c}_{sm\times i}$ 的基础上,HDFS 集群中的 DataNode 节点状态矩阵表示为

$$DS_{sm\times i}=\begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1i} \\ s_{21} & s_{22} & \cdots & s_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ s_{sm1} & s_{sm2} & \cdots & s_{smi} \end{bmatrix},$$

其中 $s_{mn}\in\{0,1,2,3\}(1\leq m\leq sm,1\leq n\leq i)$.

定义 3(文件分块矩阵). HDFS 集群中文件的存储都被拆分成一系列的数据块存放在 DataNode 节点中,并采用在同一个集群中存储多个副本的策略来提高数据的可靠性. 设 HDFS 集群中有 $k(k>3)$ 个 DataNode$\{dn\in\boldsymbol{c}_{sm\times i}\}$,文件 F 数据块副本系数为 m,数据块大小为 bs,则 HDFS 中大小为 $n\times bs$ 的 F 会有 $n\times m$ 个数据块随机存储在这 k 个 DataNode 中,其文件分块可由 $\boldsymbol{F}_{n\times m}$ 矩阵表示. 其中,原文件 F 由数据块$\{b_{11},b_{21},\cdots,b_{n1}\}$组成,而矩阵 $\boldsymbol{F}_{n\times(m-1)}$ 表示该文件的副本,n 表示文件 $\boldsymbol{F}$ 的原始数据块数量.

$$\boldsymbol{F}_{n\times m}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix},\quad \boldsymbol{F}_{n\times(m-1)}=\begin{bmatrix} b_{12} & \cdots & b_{1m} \\ b_{22} & \cdots & b_{2m} \\ \vdots & \ddots & \vdots \\ b_{n2} & \cdots & b_{nm} \end{bmatrix}.$$

HDFS 机架感知的数据块存储策略如图 2 所示,当用户向 HDFS 集群上传数据时,第 1 个数据块 b_{11} 随机存放在某个 DataNode 节点中,第 2 个数据块 b_{12}(b_{11} 的副本 1)存放在与 b_{11} 不同的机架上的

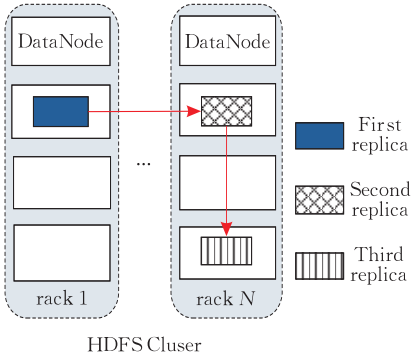


图 2 基于机架感知的数据块存储策略

DataNode 节点中，第 3 个数据块 b_{13}（b_{11} 的副本 2）存放在与 b_{12} 相同的机架但不同的 DataNode 节点中. 如果副本系数 $m>3$，其余的数据块就随机的存放在除 b_{11}、b_{12}、b_{13} 存储节点以外的任意 DataNode 节点中，由此可见基于机架感知的数据块存储策略使得数据块的存储具有随机性.

定义 4（数据块存储矩阵）. 文件 F 首先根据矩阵 $\boldsymbol{F}_{n\times m}$ 对文件进行分块，然后根据机架感知的数据块存储策略将数据块存储于 HDFS 集群中的 DataNode 节点上，即文件分块矩阵 $\boldsymbol{F}_{n\times m}$ 中的 $n\times m$ 个数据块 b_{ij}（$1\le i\le n, 1\le j\le m$）存储于矩阵 $dn\in \boldsymbol{c}_{sm\times i}$ 中的 k 个 DataNode 服务器中，并遵循一个数据块 b_{ij} 只能存储在一个 DataNode 节点中，b_{ij} 与其副本数据块不能存储在同一个 DataNode 节点中的规律. 那么，可将数据块与存储该数据块的 DataNode 节点之间的对应关系转化表示为数据块存储矩阵 $\boldsymbol{S}_{n\times m}$：

$$\boldsymbol{S}_{n\times m}=\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nm} \end{bmatrix},$$

其中，$d_{ij}\in \boldsymbol{c}_{sm\times i}$（$1\le i\le n, 1\le j\le m$）且 $\boldsymbol{S}_{n\times m}$ 中任意一行不能存在相同的元素.

定义 5（数据块状态矩阵）. 存储于不同状态的 DataNode 节点中的数据块可用性不同，根据 DataNode 节点状态矩阵 $\boldsymbol{DS}_{sm\times i}$ 与文件数据块存储矩阵 $\boldsymbol{S}_{n\times m}$ 之间的 DataNode 节点关联可得到文件数据块状态矩阵 $\boldsymbol{BS}_{n\times m}$：

$$\boldsymbol{BS}_{n\times m}=\begin{bmatrix} bs_{11} & bs_{12} & \cdots & bs_{1m} \\ bs_{21} & bs_{22} & \cdots & bs_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ bs_{n1} & bs_{n2} & \cdots & bs_{nm} \end{bmatrix},$$

其中 $bs_{ij}\in\{1,2,3\}$（$1\le i\le n, 1\le j\le m$），状态 1 表示该数据块存储在活动的 DataNode 节点中，数据块处于可用状态；2 表示该数据块存储在休眠的 DataNode 节点中，3 表示该数据块存储在宕机的节点中，状态 2 与 3 都表示数据块处于不可用状态.

定理 1. HDFS 集群中的任意文件 F 处于可用状态的条件是该文件数据块状态矩阵 $\boldsymbol{BS}_{n\times m}$ 中的每一行至少存在一个状态为 1 的数据块.

证明. 用反证法证明.

假设文件 F 的数据块状态矩阵 $\boldsymbol{BS}_{n\times m}$ 中存在第 $i(1\le i\le n)$ 行 $\{bs_{i1}, bs_{i2}, \cdots, bs_{im}\}$ 中不存在状态为 1 的数据块，即 $\{bs_{i1}, bs_{i2}, \cdots, bs_{im}\}\in\{2,3\}$（$1\le i\le n$）.

根据定义 3 与定义 5 可知此时文件 F 的 n 个分块中第 i 个数据块与其副本都处于休眠或宕机的 DataNode 节点中，数据块 $\{b_{i1}, b_{i2}, \cdots, b_{im}\}$ 都处于不可用状态.

用户读取文件 F 时，因为数据块 $\{b_{i1}, b_{i2}, \cdots, b_{im}\}$ 都处于不可用状态，n 个数据块中的第 i 个数据块将读取失败，造成不能将文件 F 组合成功返回给用户. 此时，文件 F 对于读取用户不可用. 证毕.

定义 6. 将 HDFS 中的节能问题定义为四元组 $p=\langle \boldsymbol{c}_{sm\times i}, SD, files, flag\rangle$. 其中 $\boldsymbol{c}_{sm\times i}$ 为 HDFS 集群 DataNode 节点矩阵，表示集群中所有 k 个 DataNode 节点的集合. $SD=\{dn_1, dn_2, \cdots, dn_u\}$ 表示处于不可用状态的 DataNode 节点的集合，$u=|SD|$ 表示不可用节点的数目，其中包括休眠与宕机状态的 DataNode 节点，由于本文讨论的范围为通过休眠 DataNode 节点节能，所以不考虑宕机情况，将所有不可用的 DataNode 节点统一考虑为休眠状态. $files=\{F_1, F_2, \cdots, F_w\}$ 表示集群中存储的所有文件的集合，其中 $w=|files|$ 表示文件的个数. $flag\in\{0,1\}$ 表示集群中所有文件的可用性标识，当集群中所有文件 $files=\{F_1, F_2, \cdots, F_w\}$ 满足定理 1 时，$flag=1$；当集群中存在文件不满足定理 1 时，$flag=0$.

此时通过休眠 HDFS 集群中的 DataNode 节点以达到节能目的问题转化为在保证四元组 $p=\langle \boldsymbol{c}_{sm\times i}, SD, files, flag\rangle$ 中 $flag=1$ 的前提下，怎样使得 $SD=\{dn_1, dn_1, \cdots, dn_u\}$ 集合最大的问题. 其中 $sp=u/k$ 的值表示集群中休眠 DataNode 节点的比例，sp 值越大，节能效率越高.

定理 2. 在不改变任意数据块原始存储结构的前提下，四元组 $p=\langle \boldsymbol{c}_{sm\times i}, SD, files, flag\rangle$ 中 $flag=1$ 的概率为

$$\left(1-\frac{A(u,m_1)}{A(k,m_1)}\right)^{n_1}\times\left(1-\frac{A(u,m_2)}{A(k,m_2)}\right)^{n_2}\times$$
$$\left(1-\frac{A(u,m_3)}{A(k,m_3)}\right)^{n_3}\times\cdots\times\left(1-\frac{A(u,mw)}{A(k,mw)}\right)^{n_w},$$

其中 $(n_1, n_2, n_3, \cdots, n_w)$ 分别表示集群中的 w 个文件 $files=\{F_1, F_2, \cdots, F_w\}$ 的原始分块数，$(m_1, m_2, m_3, \cdots, m_w)$ 分别表示 w 个文件的副本系数.

证明. 在拥有 k 个 DataNode 节点的 HDFS 集群中有 u 个不可用的节点，根据定理 1，当同一数据块的 m 个备份同时存储于不可用节点集合 $SD=\{dn_1, dn_1, \cdots, dn_u\}$ 中时，该数据块不可用，此事件发生的概率为

$$\frac{A(u,m)}{A(k,m)} \tag{1}$$

所以,该数据块可用的概率为

$$1-\frac{A(u,m)}{A(k,m)} \tag{2}$$

文件 F 由 n 个数据块组成,当 n 个数据块都可用时,文件 F 可用,此事件发生的概率为

$$\left(1-\frac{A(s,m)}{A(k,m)}\right)^{n} \tag{3}$$

由定义 6,$p=\langle \boldsymbol{c}_{sm\times i}, SD, files, flag\rangle$ 中 $flag=1$ 表示集群中任意文件 $files=\{F_1,F_2,\cdots,F_w\}$ 都满足定理 1,即集合 $files=\{F_1,F_2,\cdots,F_w\}$ 中所有文件都可用,此事件发生的概率为

$$\left(1-\frac{A(u,m_1)}{A(k,m_1)}\right)^{n_1}\times\left(1-\frac{A(u,m_2)}{A(k,m_2)}\right)^{n_2}\times \left(1-\frac{A(u,m_3)}{A(k,m_3)}\right)^{n_3}\times\cdots\times\left(1-\frac{A(u,m_w)}{A(k,m_w)}\right)^{n_w} \tag{4}$$

证毕.

3.2 HDFS 集群与数据块矩阵示例

如图 3 所示为一个由 4 个 RACK,每个 RACK 中分别由 3,4,2,5 个 DataNode 节点服务器组成的 HDFS 集群,其中节点 dn_{31} 与 dn_{44} 处于休眠状态,dn_{32} 与 dn_{41} 处于宕机状态. 根据定义 1 该集群 DataNode 节点矩阵表示为 $\boldsymbol{C}_{5\times4}$,根据定义 2 集群 DataNode 节点状态矩阵表示为 $\boldsymbol{DS}_{5\times4}$.

$$\boldsymbol{C}_{5\times4}=\begin{bmatrix} dn_{11} & dn_{12} & dn_{13} & dn_{14}\\ dn_{21} & dn_{22} & dn_{23} & dn_{24}\\ dn_{31} & dn_{32} & 0 & dn_{34}\\ 0 & dn_{42} & 0 & dn_{44}\\ 0 & 0 & 0 & dn_{54}\end{bmatrix},\ \boldsymbol{DS}_{5\times4}=\begin{bmatrix}1&1&1&3\\1&1&1&1\\2&3&0&1\\0&1&0&2\\0&0&0&1\end{bmatrix}.$$

设 F_1 是分块为 $n=2$、副本系数 $m=3$,F_2 是分块为 $n=3$、副本系数 $m=2$ 的文件存储于 $\boldsymbol{C}_{5\times4}$ 集群中,那么根据定义 3,F_1、F_2 文件分块矩阵为 $\boldsymbol{F}_{1\,2\times3}$、$\boldsymbol{F}_{2\,3\times2}$,根据定义 4 数据块存储矩阵为 $\boldsymbol{S}_{\boldsymbol{F}_{1\,2\times3}}$、$\boldsymbol{S}_{\boldsymbol{F}_{2\,3\times2}}$,根据定义 5 数据块状态矩阵为 $\boldsymbol{BS}_{\boldsymbol{F}_{1\,2\times3}}$、$\boldsymbol{BS}_{\boldsymbol{F}_{2\,3\times2}}$.

$$\boldsymbol{F}_{1\,2\times3}=\begin{bmatrix}b_{11}&b_{12}&b_{13}\\b_{21}&b_{22}&b_{23}\end{bmatrix},\ \boldsymbol{S}_{\boldsymbol{F}_{1\,2\times3}}=\begin{bmatrix}dn_{11}&dn_{24}&dn_{54}\\dn_{22}&dn_{13}&dn_{23}\end{bmatrix},$$

$$\boldsymbol{BS}_{\boldsymbol{F}_{1\,2\times3}}=\begin{bmatrix}1&1&1\\1&1&1\end{bmatrix},\ \boldsymbol{F}_{2\,3\times2}=\begin{bmatrix}b_{11}&b_{12}\\b_{21}&b_{22}\\b_{31}&b_{32}\end{bmatrix},$$

$$\boldsymbol{S}_{\boldsymbol{F}_{2\,3\times2}}=\begin{bmatrix}dn_{42}&dn_{21}\\dn_{24}&dn_{13}\\dn_{12}&dn_{34}\end{bmatrix},\boldsymbol{BS}_{\boldsymbol{F}_{2\,3\times2}}=\begin{bmatrix}1&1\\1&1\\1&1\end{bmatrix}.$$

假设 DataNode 节点 dn_{13} 状态转化为 2、dn_{24} 转化为 3 时,HDFS 集群 DataNode 节点状态矩阵 $\boldsymbol{DS}_{5\times4}$、文件 F_1、F_2 数据块状态矩阵 $\boldsymbol{BS}_{\boldsymbol{F}_{1\,2\times3}}$、$\boldsymbol{BS}_{\boldsymbol{F}_{2\,3\times2}}$ 改变为

$$\boldsymbol{DS}_{5\times4}=\begin{bmatrix}1&1&2&3\\1&1&1&1\\2&3&0&3\\0&1&0&2\\0&0&0&1\end{bmatrix},\ \boldsymbol{BS}_{\boldsymbol{F}_{1\,2\times3}}=\begin{bmatrix}1&3&1\\1&2&1\end{bmatrix},$$

$$\boldsymbol{BS}_{\boldsymbol{F}_{2\,3\times2}}=\begin{bmatrix}1&1\\3&2\\1&1\end{bmatrix}.$$

从数据块状态矩阵 $\boldsymbol{BS}_{\boldsymbol{F}_{1\,2\times3}}$ 与 $\boldsymbol{BS}_{\boldsymbol{F}_{2\,3\times2}}$ 我们可以看出,节点 dn_{13} 与 dn_{24} 的状态改变并没有影响文件 F_1 数据块的可用性,但是使得文件 F_2 第 2 个数据

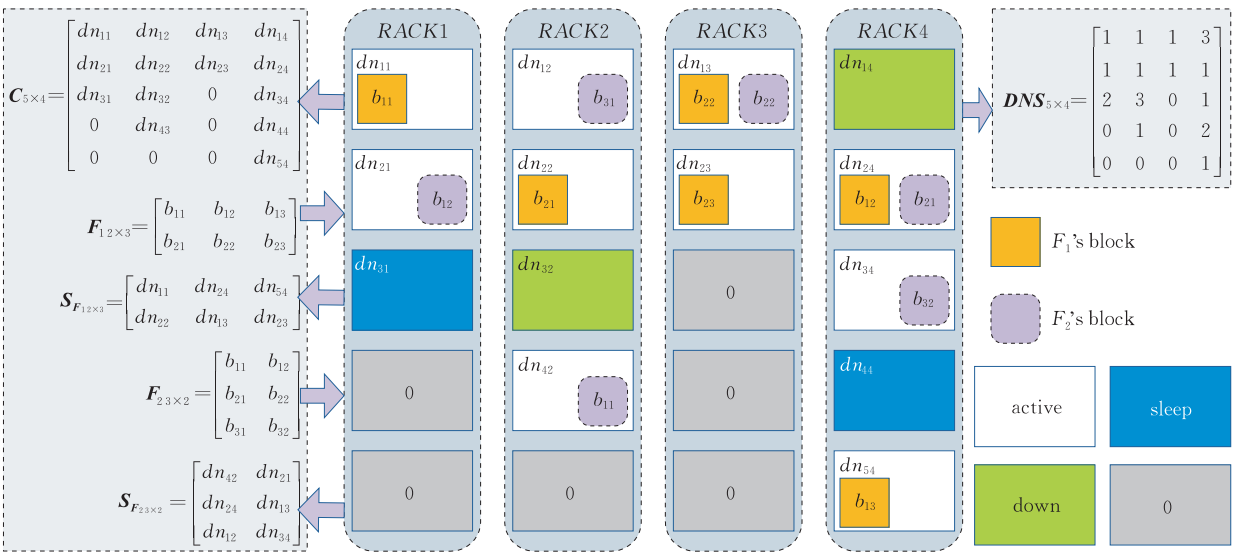


图 3 HDFS 集群中文件数据块存储与状态示例

块变得不可用. 由此可见节点 dn_{13} 与 dn_{24} 不能同时进入休眠状态，否则将影响文件 F_2 的数据块可用性.

通过定理 2 的证明过程与上述示例可以看出，当集群中处于不可用状态的节点数量大于等于文件副本系数，即 $u \geq m$ 时，该文件都有可能处于不可用状态. 在不改变数据块原始存储结构的情况下，最多能休眠 $u=m-1$ 个 DataNode 节点，在 k 值较大时达到的节能效果甚微. 所以，本文下面设计了数据块存储结构配置算法，根据文件的访问规律动态配置数据块的存储结构，RACK 区域划分算法将 RACK 分成 Active-Zone 与 Sleep-Zone 两个区域，通过休眠 Sleep-Zone 中的 DataNode 节点达到节能的目的. 对称数据块存储策略与该存储策略下的节能算法在不影响原 HDFS 可靠性、高效性与可扩展性的前提下，对称数据块存储策略使得具有相同存储结构的 DataNode 节点之间能够实现任务的转移与替换，算法通过任务调控产生并休眠空闲节点从而达到节能的目的.

4　HDFS 节能算法

4.1　DataNode 节点休眠验证算法

算法 1.　DataNode 节点休眠验证算法.

```
输入：待验证的节点 ID，datanodeID；
初始化：
  Set blocks←new List〈Block〉();
  block←new Block();
  file←new File();
  rowNum←0;
  cloumnNum←0;
  activeNumInRow←0;
  rowValue[]←new int[];
  blockStatusMatrix←new BlockStatusMatrix();
  updatedBlockStatusMatrix ← new BlockStatusMatrix();
1. blocks←getBlocks(datanodeID);
2. for i=0 to blocks.size()−1 do
3.   block←blocks.get(i);
4.   file←getFile(block);
5.   blockStatusMatrix←getBlockStatusMatrix(file);
6.   updatedBlockStatusMatrix←
       updateMatrix(blockStatusMatrix,block,2);
7.   rowNum←updatedBlockStatusMatrix.getRowNum();
8.   cloumnNum←updatedBlockStatusMatrix.getCloumnNum();
9.   for j=0 to rowNum−1 do
10.    rowValue[cloumnNum]←
         updatedBlockStatusMatrix.getRowValue(j);
11.    for k=0 to cloumnNum−1 do
12.      if rowValue[k]==1 then
13.        activeNumInRow++;
14.      end if
15.    end for
16.    if activeNumInRow==0 then
17.      return false
18.    end if
19.  end for
20. end for
21. return true
```

定理 1 可用于验证一个 DataNode 节点由活动状态转化为休眠状态是否影响数据块的可用性. 当任意 DataNode 节点由活动状态转化到休眠状态时，使得该节点上所有的数据块不可用，通过更新并验证休眠 DataNode 节点上所有数据块所属文件的数据块状态矩阵 $\boldsymbol{BS}_{n\times m}$，是否违反数据块可用性原则来验证该节点能否进入休眠状态. DataNode 节点可休眠验证算法如算法 1 所示.

算法第 1 行通过输入参数 *datatnodeID* 得到 DataNode 节点上所有数据块的集合，遍历所有数据块并通过数据块信息得到数据块所属文件的 *file* 对象，通过 *file* 对象可以得到该文件的数据块状态矩阵. 算法第 6 行更新 DataNode 节点上所有数据块状态由 1(活动)转化为 2(休眠)后的数据块状态矩阵. 循环数据块状态矩阵的每一行，验证新的数据块状态矩阵行是否能满足数据块可用性的要求. 如果有任意数据块变得不可用，第 17 行返回 false，表示该 DataNode 节点不能进入休眠状态；如果循环结束，验证所有的数据块可用性并没有受到 DataNode 节点休眠的影响，返回 true，此时可将该节点休眠节能.

4.2　数据块存储结构配置节能算法

文献[32-33]通过对 Yahoo 公司 HDFS 集群内部数据块访问日志的分析，得出 90.26% 的数据块都会在其上传 2 天内进行第一次访问，89.61% 的数据块都会再其上传后的 10 天内进行最后一次访问，40% 的数据块最后一次读取时间到最后删除的时间跨度都不会超过 20 天. 由此可见 HDFS 集群内部数据块的访问具有较强的规律性，数据块访问的热点期大多集中在其上传后较短的一段时间内，过后则进入空闲期. 根据数据块这种访问规律并结合数据块矩阵本文设计了数据块存储结构配置节能算法. 如图 4 所示，RACK 区域划分算法将 RACK 分成 Active-Zone 与 Sleep-Zone 两个区域，Active-Zone 中的 DataNode 节点用于存放状态为 1 的数据块，而 Sleep-Zone 中的节点用于存放状态为 2 的数据块，并通过休眠处于 Sleep-Zone 的节点来达到节

能的目的.算法在保证数据块可用性的前提下,在算法执行时间周期 T 内根据文件的访问频率调整该文件的数据块状态矩阵 $\boldsymbol{BS}_{n\times m}$ 中处于活动状态数据块的数量.如果一个文件在时间周期 T 内访问次数小于阈值 *sleepCount*,算法则会减小该文件的活动数据块的数量,相反如果访问次数大于阈值 *activeCount*,则增加活动数据块的数量;算法在达到节能目的的同时保证了数据块的可用性.算法如算法 2 所示.

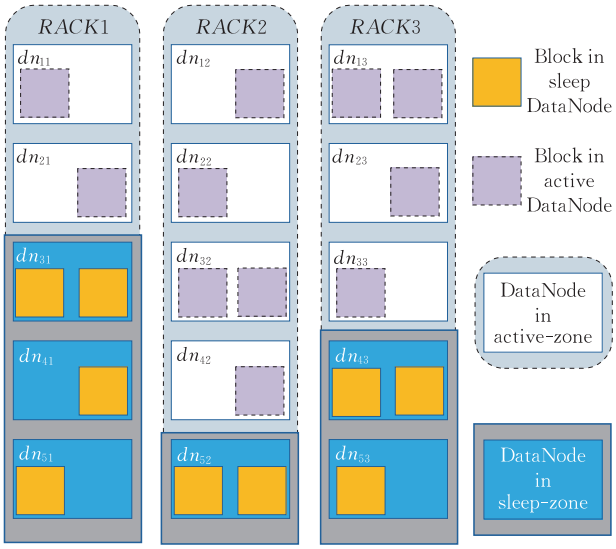


图 4 RACK 划分示意图

算法 2. 数据块存储结构配置节能算法.

输入:

HDFS 集群中所有数据文件: *List⟨File⟩ allFileInHDFS*;

算法执行周期: *T*;

将数据块从 Active-Zone 移动到 Sleep-Zone 的阈值: *sleepCount*;

将数据块从 Sleep-Zone 移动到 Active-Zone 的阈值: *activeCount*;

初始化

Set *file*←*new File*();

fileNumInHDFS←0;

visitCount←0;

activeBlockNumInRow←0;

cloumnNum←0;

1. if *executionTime*(*T*) then
2. *acitveAllSleepDataNodes*();
3. *fileNumInHDFS*←*allFileInHDFS.size*();
4. for *i*=0 to *fileNumInHDFS*−1 do
5. *file*←*allFileInHDFS.get*(*i*);
6. *visitCount*←*getVisitCountFromLog*(*file*);
7. *blockStatusMatrix*←*getBlockStatusMatrix*(*file*);
8. *cloumnNum*←*blockStatusMatrix.getCloumnNum*();
9. *activeBlockNumInRow*←*blockStatusMatrix.getAcitveNum*();
10. if *visitCount*<*sleepCount* then
11. if *activeBlockNumInRow*>1 then
12. *reduceActiveBlockNum*(*blockStatusMatrix*);
13. *moveBlockToSleepZone*(*block*,2);
14. end if
15. end if
16. if *visitCount*>*activeCount* then
17. if *activeBlockNumInRow*<*cloumnNum* then
18. *increaseActiveBlockNum*(*blockStatusMatrix*);
19. *moveBlockToActiveZone*(*block*,1);
20. end if
21. end if
22. *updateBlockStoreMatrix*(*file*);
23. *updateBlockStatusMatrix*(*file*);
24. end if
25. end if

算法输入参数 T 为算法执行时间周期(如一天、一周等),可根据不同的 HDFS 集群的特点设置不同的 T 值并最好选择在 HDFS 集群较为空闲时执行算法;*sleepCount* 与 *activeCount* 为减小与增加文件活动数据块数量的阈值,参数 *allFileInHDFS* 表示集群中所有文件的集合.算法第 1 行表示达到执行时间周期 T 条件时开始执行算法,代码行 2 首先唤醒所有的休眠节点,以便后续操作的顺利进行.代码行 4 循环遍历 HDFS 集群中所有的文件数据块状态矩阵,并根据文件的访问日志得到数据块的访问次数,与阈值参数 *sleepCount*、*activeCount* 进行比较判断该文件是减小还是增加活动数据块的数量,并将状态改变后的数据块转移到相应的区域中,最后 22、23 行更新该文件的数据块存储与状态矩阵.数据块存储结构配置节能算法需要 RACK 区域划分算法的支持.RACK 区域划分算法如算法 3 所示.

算法 3. RACK 区域划分算法.

输入:

HDFS 集群中所有的 Rack: *List⟨Rack⟩ racks*;

初始化:

Set *rack*←*new Rack*();

block←*new Block*();

dataNode←*new DataNode*();

dataNodesInRack←*new List⟨DataNode⟩*();

blocksInDataNode←0;

blocksInRack←0;

activeBlocksInRack←0;

dataNodeNum←0;

activeDataNodeNum←0;

```
sleepDataNodeNum←0;
1. for i=0 to racks.size()−1 do
2.   rack←racks.get(i);
3.   dataNodeNum←rack.getDataNodeNum(i);
4.   dataNodesInRack←rack.getDataNodes(i);
5.   for j=0 to dataNodeNum−1 do
6.    dataNode←racks.get(j);
7.    blocksInDataNode←dataNode.getBlockNum();
8.    blocksInRack+=blocksInDataNode;
9.    for k=0 to blocksInDataNode−1 do
10.     block←dataNode.getBlock(k);
11.     if block.status==1 then
12.       activeBlocksInRack++;
13.     end if
14.   end for
15. end for
16. activeDataNodeNum←
      (activeBlocksInRack/blocksInRack)*
       dataNodeNum;
17. sleepDataNodeNum←dataNodeNum−
      activeDataNodeNum;
18. return activeDataNodeNum,sleepDataNodeNum
19. end for
```

RACK 区域划分算法解决了 RACK 中 Active-Zone 与 Sleep-Zone 区域节点分配问题，算法采用与 Hadoop 原 balancer 相同的策略，即将 RACK 中的所有数据块平均分配到该 RACK 所属节点中，按照 RACK 中活动与休眠数据块的比例计算活动与休眠节点的数量，算法伪代码如算法 3 所示. 算法首先遍历 RACK 中所有的 DataNode 节点与节点中的数据块，算法第 12 行计算 RACK 中活动数据块的数量，算法第 16 行通过计算活动数据块与总数据块的比值来计算 Active-Zone 区域中节点的数量，代码第 17 行通过 RACK 中总的节点数量减去 Active-Zone 区域中节点的数量便得到 Sleep-Zone 区域节点的数量. RACK 区域划分算法应选择在数据块存储结构配置节能算法第 13 行代码前执行.

4.3　对称数据块存储策略下的节能算法

机架感知的数据块存储策略下集群中数据块的存储具有随机性，任意两个 DataNode 节点存储结构不可能完全相同，此种情况下只能通过 4.2 节重新配置数据块存储结构以达到节能的目的. 假设存在相同数据块存储结构的 DataNode 节点 dn_1 与 dn_2，即 dn_1 与 dn_2 之间的任务可以相互转移与替换. 当 dn_1，dn_2 负载率都较低时，将 dn_1 与 dn_2 其中之一进行休眠处理，此时 dn_1，dn_2 中的数据块可用性将不会受到任何影响. 基于这种思想本文设计了对称数据块存储策略如图 5 所示. 按照数据块分块矩阵 $\boldsymbol{F}_{n\times m}$ 对称地存储于 HDFS 集群中，其中 Rack1 中存放着文件 F 的原始数据块，Rack2 到 Rack$m$ 都存放原始数据块的副本，即为矩阵 $\boldsymbol{F}_{n\times(m-1)}$ 中表示的数据块. DataNode 节点矩阵为 $\boldsymbol{C}_{r\times m}$：

$$\boldsymbol{C}_{r\times m}=\begin{bmatrix} dn_{11} & dn_{12} & \cdots & dn_{1m} \\ dn_{21} & dn_{22} & \cdots & dn_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ dn_{r1} & dn_{r2} & \cdots & dn_{rm} \end{bmatrix}.$$

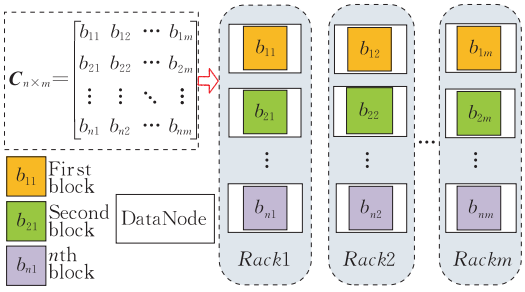


图 5　对称数据块存储策略

当数据块上传到集群中时按照对称数据块存储策略进行存储，该存储策略使得 $\boldsymbol{C}_{r\times m}$ 矩阵中每一行中的 m 个 DataNode 数据块存储结构完全一致，互为备份. 对称数据块存储策略简化了 HDFS 集群数据块可用性的判断条件，只要 $\boldsymbol{C}_{r\times m}$ 中的每行保证至少一个节点处于活动状态，HDFS 集群中所有数据块都可用. 由于副本系数 m 为一可变因子，设不同 m 值的 $\boldsymbol{C}_{r\times m}$ 矩阵为一组，一个 HDFS 集群可由多个组构成，其结构表示为

$$\boldsymbol{C}_{\mathrm{HDFS}}=\boldsymbol{C}_{r1\times 1}+\boldsymbol{C}_{r2\times 2}+\cdots+\boldsymbol{C}_{rm\times m}.$$

对称数据块存储策略使得 $\boldsymbol{C}_{r\times m}$ 组中每一行 m 个 DataNode 节点数据块存储结构完全一致，m 个 DataNode 节点之间的任务可以相互转移与替换，节能算法通过控制任务的调度分配产生空闲节点，将空闲节点休眠以达到节能的目的. 图 6 表示了对称

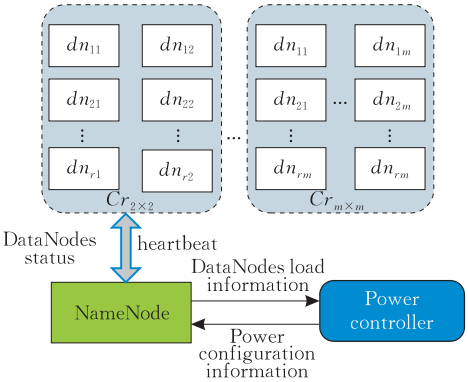


图 6　对称数据块存储策略下的节能算法架构图

数据块存储策略下的节能算法架构图，DataNode 通过心跳向 NameNode 发送负载信息，能耗控制器 PowerController 用于接收与处理 NameNode 发送过来的节点负载信息，PowerController 根据不同的负载情况进行处理，将处理后的配置信息传送回 NameNode 节点，最后由 NameNode 执行 PowerController 发出的节能控制信息. 本文以 $m=3$ 时为例阐述对称数据块存储策略下的节能算法的步骤，设其 DataNode 负载状态如图 7 所示.

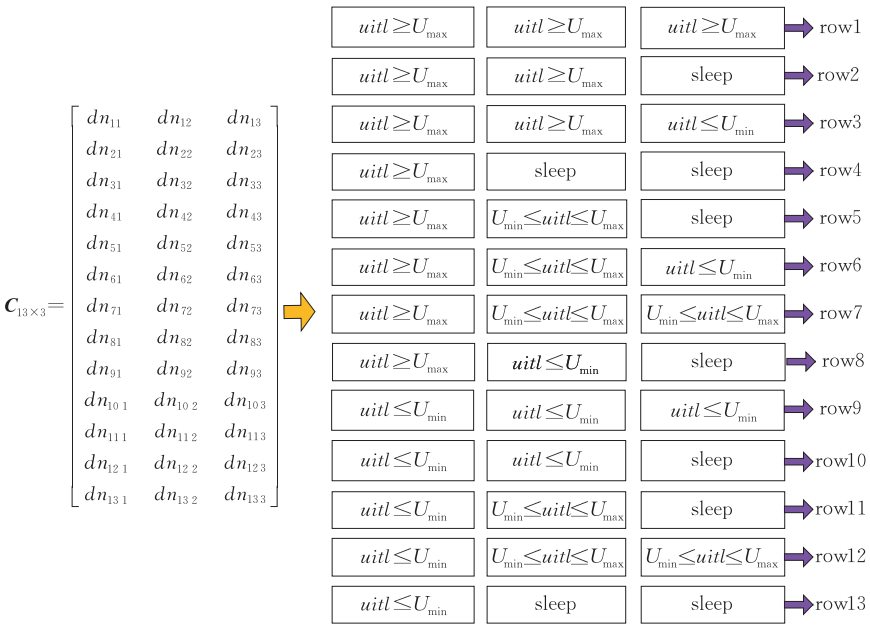


图 7 当 $m=3$ 时 DataNode 负载状态示意图

算法 4. 对称数据块存储策略下的节能算法.

1. 初始化. 设置 DataNode 最大负载阈值 U_{max} 与休眠负载阈值 U_{min}，算法执行时间周期 T.

2. DataNode 节点定期向 NameNode 节点发送 heartbeat 心跳信息，心跳信息包括 DataNode 节点负载状态信息. 当达到算法执行时间周期 T 时，NameNode 将负载介于 U_{min} 与 U_{max} 之间的节点视为正常节点，当有节点负载小于 U_{min} 或者大于 U_{max} 时，NameNode 将该 DataNode 节点与其同行所有的 DataNode 负载状态信息放入 List〈RowDataNodeStatus〉中并发送给 PowerController 节点进行处理.

3. 当 PowerController 接收到如 row2 所示负载信息，即有 DataNode 节点负载均超过 U_{max} 时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内停止向 dn_{11}，dn_{12}，dn_{13} 节点分配数据写入操作，新写入数据任务分配到其它组；转向执行步 16.

4. 当 PowerController 接收到如 row2 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：唤醒休眠节点 dn_{23} 并将 dn_{21} 与 dn_{22} 部分任务转移到 dn_{23} 节点执行；转向执行步 16.

5. 当 PowerController 接收到如 row3 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将分配任务到节点 dn_{31} 与 dn_{32} 的任务转移到 dn_{33} 节点；转向执行步 16.

6. 当 PowerController 接收到如 row4 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：唤醒 dn_{42} 与 dn_{43} 中休眠时间较长的节点并在时间周期 T 内停止向 dn_{41} 分配任务；转向执行步 16.

7. 当 PowerController 接收到如 row5 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{51} 部分任务转移到 dn_{52} 执行，如果在周期 T 内出现 row2 所示情况，则执行步 4；否则转向执行步 16.

8. 当 PowerController 接收到如 row6 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{61} 任务转移到 dn_{63} 执行；转向步 16.

9. 当 PowerController 接收到如 row7 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{71} 任务转移到 dn_{72} 与 dn_{73} 负载较轻的节点；转向执行步 16.

10. 当 PowerController 接收到如 row8 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{81} 任务转移到 dn_{82} 节点，如在时间周期 T 内出现 row2 情况，转向步 4，否则转向步 16.

11. 当 PowerController 接收到如 row9 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{91}，dn_{92}，dn_{93} 中负载最低的节点（假设为 dn_{92}）任务转移到其它两个节点（dn_{91}，dn_{93}），当 dn_{92} 负载降为 0 时进入休眠状态；转向执行步 16.

12. 当 PowerController 接收到如 row10 所示负载信息

时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{10_1} 负载转移到 dn_{10_2}（假设 dn_{10_1} 负载小于 dn_{10_2}），当 dn_{10_1} 负载降为 0 时进入休眠状态；转向执行步 16.

13. 当 PowerController 接收到如 row11 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{11_1} 负载转移到 dn_{11_2}，当 dn_{11_1} 负载降为 0 时进入休眠状态；转向执行步 16.

14. 当 PowerController 接收到如 row12 所示负载信息时，PowerController 向 NameNode 节点返回配置信息：在时间周期 T 内将 dn_{12_1} 负载转移到 dn_{12_2} 与 dn_{12_3} 中负载较低节点，当 dn_{12_1} 负载降为 0 时进入休眠状态；转向执行步 16.

15. 当 PowerController 接收到如 row13 所示负载信息时，不作任何处理；转向执行步 16.

16. 执行 List〈RowDataNodeStatus〉队列的 next()方法执行下一行 DataNode 状态，如果队列中所有的行都处理完成，算法结束.

5　算法分析模型

本节给出 HDFS 下数据库存储结构配置节能算法与对称数据块策略下的节能算法的分析模型，其中符号定义说明如表 1 所示.

表 1　符号定义说明表

符号	定义
F	文件表示符号
n	文件分块原始数据块的数量
m	数据块副本系数
k	HDFS 集群中 DataNode 节点的数量
u	集群中处于不可用状态的 DataNode 节点的数量
sd	集群中休眠的 DataNode 节点的数量
w	HDFS 集群中所有文件的个数
z	组成 HDFS 集群的 $\boldsymbol{C}_{r\times m}$ 组数
r	$\boldsymbol{C}_{r\times m}$ 中 DataNode 节点的行数
P	DataNode 节点服务器功耗
E	DataNode 节点服务器能耗
a	DataNode 节点服务器电路翻转频率
C	DataNode 节点服务器负载电容
f	DataNode 节点服务器时钟频率
V	电压
S	DataNode 节点服务器服务速率
Pro	数据块可用概率

服务器中系统能耗主要由处理器、内存、磁盘 I/O、散热设备等等组成，据文献[35]服务器功耗由静态功耗与动态功耗组成：

$$P_{\text{server}} = P_{\text{static}} + P_{\text{dynamic}} \tag{5}$$

其中 P_{dynamic} 可表示为

$$P_{\text{dynamic}} = aCV^2 f \tag{6}$$

其中，a 为电路翻转频率，C 是负载电容，f 为时钟频率，V 为电压. 电压 V 与时钟频率 f 成正比，f 时钟频率与系统性能成正比. 文献[35]将服务器系统功耗简化表示为

$$P_{\text{server}} = \sigma_e + \mu_e s^a,\ a > 1 \tag{7}$$

其中，σ_e 为空闲功耗或者静态功耗；s 为服务器服务速率，与式(5)中 f 成正比；μ_e 和 a 为常数，与具体的设备相关. 文献[36]指出 a 的取值通常在 3 之间波动，而文献[35]对 Intel PXA270、Pentium M770 与 TCP/IPOffload Engine 进行了研究并得出三者 a 值分别为 1.11、1.62、1.66.

由于能耗与功耗是完全不同的概念，能耗为服务器在一段时间内所消耗的总能量，单位为焦耳(J)，在 Δt 时间内的能耗定义如式(8)所示：

$$E = \int_{t}^{t+\Delta t} P \times \mathrm{d}t \tag{8}$$

数据块存储结构配置节能算法靠休眠 Sleep-Zone 区域的 DataNode 节点达到节能的目的，设所有 RACK Sleep-Zone 中休眠节点数量为 sd，那么 Active-zone 中的节点数量为 $k-sd$，节能算法执行前集群中 k 个节点都处于活动状态，此时集群功耗为 k 个节点功耗的总和：

$$P_{\text{hdfs}} = \sum_{i=1}^{k} (\sigma_e + \mu_e s_1^a)_i,\ a > 1 \tag{9}$$

当休眠 sd 个节点后，剩余 $k-sd$ 个节点消耗能量，此时集群功耗为

$$P_{\text{sleep}} = \sum_{j=1}^{k-sd} (\sigma_e + \mu_e s_2^a)_j,\ a > 1 \tag{10}$$

那么由数据块存储结构配置节能算法节省的功耗为

$$P_{\text{save}} = \sum_{i=1}^{k} (\sigma_e + \mu_e s_1^a)_i - \sum_{j=1}^{k-sd} (\sigma_e + \mu_e s_2^a)_j \tag{11}$$

其中，s_1 为算法执行前的服务器速率，s_2 为算法执行后的服务器速率，一般情况下有 $s_1 < s_2$. 由数据块存储结构配置节能算法在 Δt 时间内节省的能量可表达为

$$E_{\text{save\_}\Delta t} = \int_{t}^{t+\Delta t} \left(\sum_{i=1}^{k} (\sigma_e + \mu_e s_1^a)_i - \sum_{j=1}^{k-sd} (\sigma_e + \mu_e s_2^a)_j \right) \times \mathrm{d}t \tag{12}$$

对称数据块存储策略下的节能算法进行 DataNode 之间任务转移与替换时将改变 DataNode 服务速率 s，即减少了任务转移节点的功耗而增加了任务转移目标节点的功耗，特别的当某一个 DataNode 节点进入休眠状态后，其功耗可忽略不计，即 $P_{\text{server}} = 0$. 所以在 HDFS 集群某副本系数为 m 组的某一行中，由节能算法带来的功耗节省可表示为

$$P_{\text{save\_row}}=\sum_{i=1}^{m}(P-P')_i$$

$$=\sum_{i=1}^{m}((\sigma_e+\mu_e s^a)-(\sigma_e+\mu_e s^a)')_i \quad (13)$$

其中，P 表示该行中第 i 个 DataNode 节点节能算法执行前的功耗，P' 表示该节点节能算法执行后的功耗. 那么一个 $\boldsymbol{C}_{r\times m}$ 组功耗节省为

$$P_{\text{save\_group}}=\sum_{j=1}^{r}\sum_{i=1}^{m}((\sigma_e+\mu_e s^a)-(\sigma_e+\mu_e s^a)')_{ij} \quad (14)$$

其中 r 表示 $\boldsymbol{C}_{r\times m}$ 组中 DataNode 节点行数. 假设 HDFS 集群由 z 个 $\boldsymbol{C}_{r\times m}$ 组组成，那么整个 HDFS 集群功耗节省可表示为

$$P_{\text{save\_HDFS}}=\sum_{k=1}^{z}(P_{\text{save\_group}})_k$$

$$=\sum_{k=1}^{z}\Big(\sum_{j=1}^{r}\sum_{i=1}^{m}((\sigma_e+\mu_e s^a)-(\sigma_e+\mu_e s^a)')_{ij}\Big)_k \quad (15)$$

由于 HDFS 集群是负载不断变化的系统，所以其功耗也在不断变化，由节能算法在 Δt 时间内节省的能量为

$$E_{\text{save\_}\Delta t}=\int_{t}^{t+\Delta t}\Big(\sum_{k=1}^{z}\Big(\sum_{j=1}^{r}\sum_{i=1}^{m}((\sigma_e+\mu_e s^a)-(\sigma_e+\mu_e s^a)')_{ij}\Big)_k\Big)\times \mathrm{d}t \quad (16)$$

6 评价与比较

基于第 5 节所述数学分析模型，本文使用 Matlab Cloudsim[37] 对本文算法进行模拟分析，通过实验评价与比较，验证数据块的可用性、存储结构配置节能算法与对称数据块存储策略下的节能算法的有效性. 其中 Cloudsim 模拟产生的 DataNode 节点配置参数如表 2 所示.

表 2 DataNode 节点配置参数

CUP (cores/MIPS)	1 core/10 000 mips
内存 RAM	1 GB
操作系统 OS	Linux
硬盘容量	100 GB
空闲功耗 σ_e	100
能耗常数 μ_e	70
能耗常数 a	1.5

6.1 数据块可用性实验

如图 8 所示为在不改变数据块原始存储结构前提下不可用 DataNode 节点数与 HDFS 集群数据块可用性之间的关系，其中横轴表示不可用节点的数量，纵轴表示集群中所有数据块可用的概率. 其 line1 表示集群中节点数 $k=100$，副本系数 $m=3$，集群中原数据块个数为 10 000 时，不可用状态的节点数量与整个系统可用性概率之间的关系. line1 与 line2 比较，当 k 与 n 条件相同时，副本系数 m 值越大，HDFS 集群可用概率越高；line2 与 line3 比较，当 m 与 n 条件相等时，DataNode 节点数越大，HDFS 集群可用概率越高；line1 与 line4 比较，当 m 与 k 条件相等时，n 越小，HDFS 集群可用概率越高. 但是无论 HDFS 集群有多少 DataNode 节点，其中有多少原始数据块，当集群中不可用的节点数大于等于副本系数 m 时，都有可能引起数据块的不可用，意味着在不改变数据块存储结构与存储策略的情况下并不能通过休眠节点达到节能的目的.

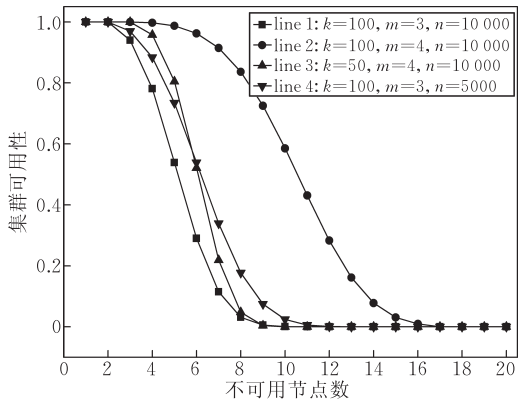


图 8 不可用节点数与集群可用性之间的关系

6.2 存储结构配置节能算法

通过第 5 节的数学建模分析，数据块存储结构配置节能算法通过休眠 Sleep-Zone 区域的 DataNode 节点达到节能的目的，节能效率由 Sleep-Zone 中休眠节点数量 sd 与算法执行前后的服务器速率 s_1 与 s_2 决定. 而 sd、s_1 与 s_2 三个变量的值又与 HDFS 中数据块的负载、算法执行周期 T、阈值 $sleepCount$ 与 $activeCount$ 相关. 利用 Cloudsim 模拟产生 10 个 RACK 且每个 RACK 中有 10 个 DataNode 节点即 $k=100$ 的 HDFS 集群，数据块副本值随机产生 $m=random[2,5]$，原始数据块数 n 取 2～10 的随机数，初始化 10 000 个文件存储在该集群中. 为了模拟出文献[32-33]数据块访问规律，设文件在上传到集群后第 1 天的访问次数取 0～100 的随机数，第 2 天取 0～90 的随机数，第 3 天为 0～80 的随机数，依次类推，10 天及以后访问次数取 0～10 的随机数. 如图 9

所示为设算法执行周期 $T=1$ 天、阈值参数 *sleepCount* 分别取值 5 与 10，阈值参数 *activeCount* 分别取值 30 与 50 时算法模拟的结果.

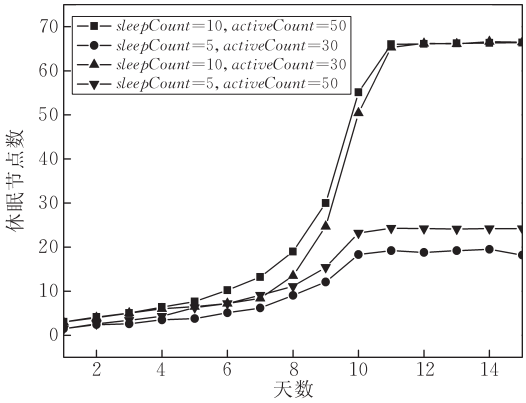


图 9　休眠节点数与算法参数的关系

由图 9 可以看出 *sleepCount* 取值越小，*activeCount* 取值越大时相同集群与负载条件下产生的休眠节点数越多；即相同条件下 *sleepCount* 参数越小，节能效率越高，*sleepCount* 是平衡数据块访问频率与集群节能效率的参数；而 *activeCount* 越小，节能效率越低，但同一数据块备份数越多时，数据块响应效率越高，即 *activeCount* 是平衡数据块响应与节能效率的参数. 不同的 HDFS 集群具有不同的负载情况与用户 QoS 协议，应根据实际的情况设定算法执行周期 T、阈值参数 *sleepCount* 与 *activeCount*. 当集群中所有数据块与其副本中只有一个数据块存储在 Active-Zone 区域中时，休眠节点数量达到最大，此时休眠节点数量达到 $\left(\frac{\overline{m}-1}{\overline{m}}\right)\times k$，其中 $\overline{m}$ 表示集群中所有文件的平均副本系数；此时节能效率达到最高的$(1-1/\overline{m})$.

当空闲功耗参数 $\sigma_e=100$，能耗常数 $\mu_e=300$，如图 10 为在 $k=100$ 的 HDFS 集群中不同休眠节点

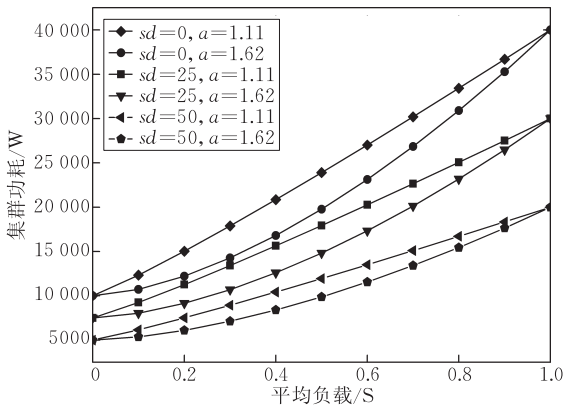


图 10　不同休眠节点与 a 值下集群功耗与平均负载之间的关系

数量与常数 a 取不同值时集群功耗与集群平均负载的关系示意图.

如图 10 可以看出，相同集群平均负载率条件下，休眠节点数越大，集群功耗越小；相同集群休眠节点数与集群平均负载条件下，常数 a 值越小，集群功耗越小. 由此可见存储结构配置节能算法不仅能够通过增加休眠节点数来提高节能效率，也可以通过选取常数 a 值较小的系统硬件来提高系统节能效率.

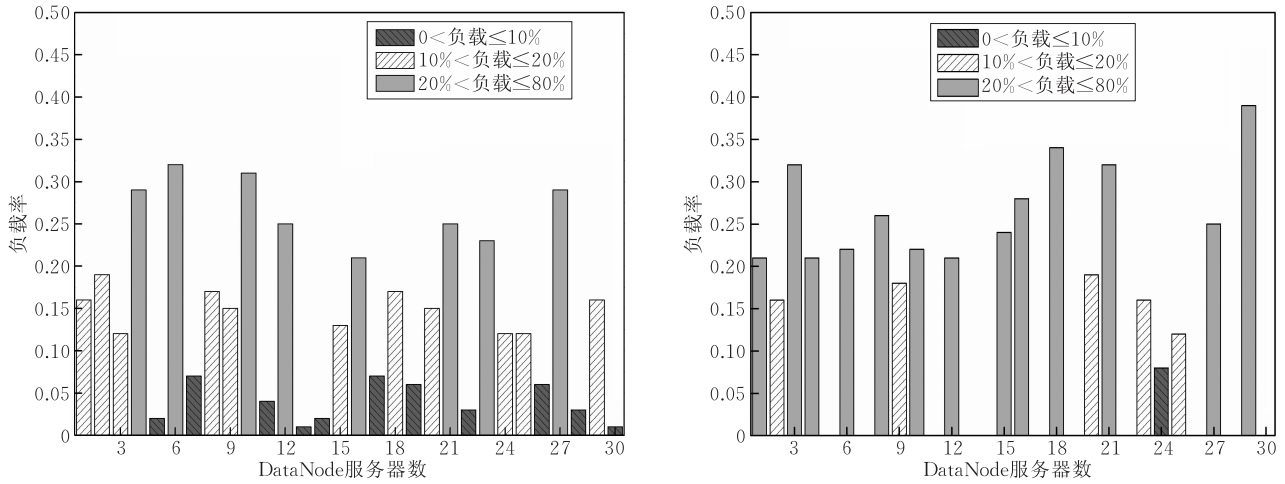
6.3　对称数据块存储策略下的节能算法

利用 Cloudsim 模拟产生的 $m=3, r=10$ 的 DataNode节点 $\boldsymbol{C}_{10\times 3}$ 组(节点配置如表 2 所示)，为了测试不同负载情况下节能算法的效率，分别向该集群在限定范围内施加低负载、随机负载与高负载 3 种情况. 表 3 为 3 种不同负载情况下节能算法执行前后节点负载比较，其中低负载初始平均负载为 0.1403，节能算法执行后平均负载为 0.1453；随机负载初始平均负载为 0.4442，节能算法执行后平均负载为 0.4473；高负载初始平均负载为 0.7083，节能算法执行后平均负载为 0.7093.

表 3　节能算法执行前后负载比较

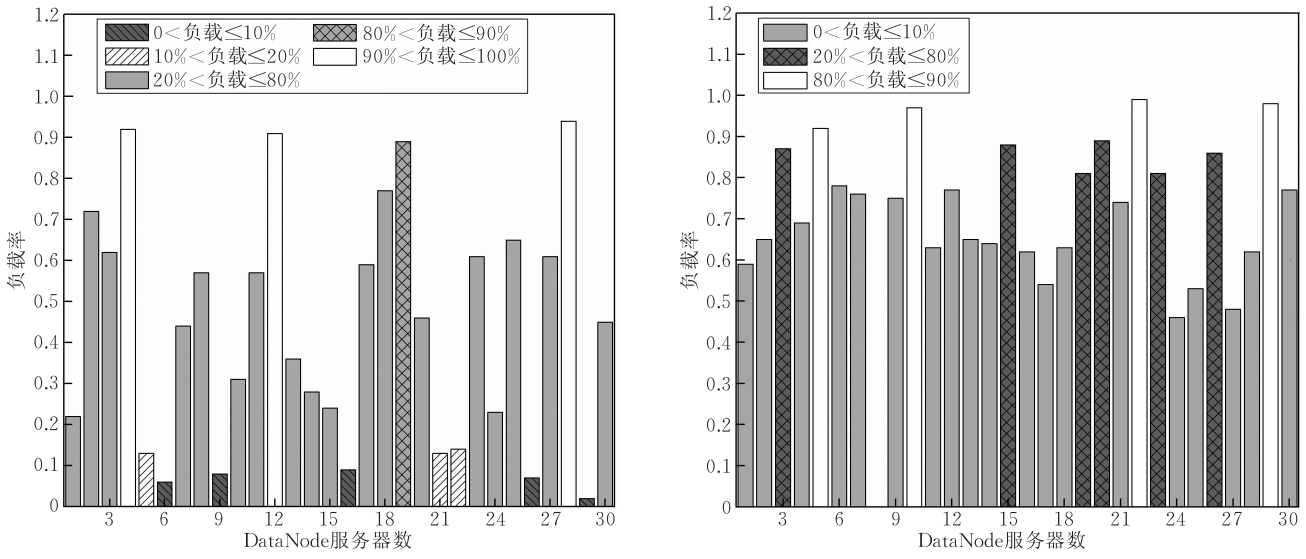
$\boldsymbol{DN}_{r\times m}$	低负载	算法执行后	随机负载	算法执行后	高负载	算法执行后
dn_{11}	0.16	0.21	0.22	0.54	0.59	0.59
dn_{12}	0.19	0.16	0.72	0.67	0.45	0.65
dn_{13}	0.12	0.32	0.62	0.48	0.87	0.87
dn_{21}	0.29	0.21	0.92	0.51	0.69	0.69
dn_{22}	0.02	0	0.13	0.69	0.92	0.92
dn_{23}	0.32	0.22	0.06	0	0.88	0.78
dn_{31}	0.07	0	0.44	0.55	0.76	0.76
dn_{32}	0.17	0.26	0.57	0.34	0.08	0
dn_{33}	0.15	0.18	0.08	0	0.75	0.75
dn_{41}	0.31	0.22	0.31	0.88	0.97	0.97
dn_{42}	0.04	0	0.57	0.42	0.93	0.63
dn_{43}	0.25	0.21	0.91	0.36	0.77	0.77
dn_{51}	0.01	0	0.36	0.43	0.65	0.65
dn_{52}	0.02	0	0.28	0.22	0.64	0.64
dn_{53}	0.13	0.24	0.24	0.38	0.88	0.88
dn_{61}	0.21	0.28	0.09	0	0.62	0.62
dn_{62}	0.07	0	0.59	0.78	0.84	0.54
dn_{63}	0.17	0.34	0.77	0.66	0.33	0.63
dn_{71}	0.06	0	0.89	0.86	0.81	0.81
dn_{72}	0.15	0.19	0.46	0.53	0.89	0.89
dn_{73}	0.25	0.32	0.13	0.34	0.74	0.74
dn_{81}	0.03	0	0.14	0.08	0.99	0.99
dn_{82}	0.23	0.16	0.61	0.35	0.81	0.81
dn_{83}	0.12	0.08	0.23	0.41	0.36	0.46
dn_{91}	0.12	0.12	0.65	0.68	0.53	0.53
dn_{92}	0.06	0	0.07	0	0.86	0.86
dn_{93}	0.29	0.25	0.61	0.78	0.38	0.48
dn_{10_1}	0.03	0	0.94	0.65	0.52	0.62
dn_{10_2}	0.16	0.39	0.02	0.45	0.98	0.98
dn_{10_3}	0.01	0	0.45	0.38	0.76	0.77
平均负载	0.1403	0.1453	0.4442	0.4473	0.7083	0.7093

如图 11 表示低负载状态下节能算法执行前后负载对比，在平均负载为 0.1403 的情况下，节能算法执行后将编号为 dn_{22}、dn_{31}、dn_{42}、dn_{51}、dn_{52}、dn_{62}、dn_{71}、dn_{81}、dn_{92}、dn_{10_1}、dn_{10_3}共 11 个节点都进行了休眠处理，休眠率为 37%，并且节能算法执行后平均负载变为 0.1453. 如图 12 表示随机负载状态下节能算法执行前后负载对比，在平均负载为 0.4442 的情况下，节能算法执行后将编号为 dn_{23}、dn_{33}、dn_{61}、dn_{92}这 4 个节点进行了休眠处理，休眠率为 13.3%. 节能算法执行后平均负载相比之前变为 0.4473，平均负载在节能算法的执行前后变化不大. 如图 13 为高负载状态下节能算法执行前后负载对比，在平均负载为 0.7083 的情况下，节能算法执行后将编号为 dn_{32}的唯一节点进行了休眠处理，休眠率为 3.3%. 节能算法执行后平均负载相比之前变为 0.7093.



(a) 节能算法执行前 (b) 节能算法执行后

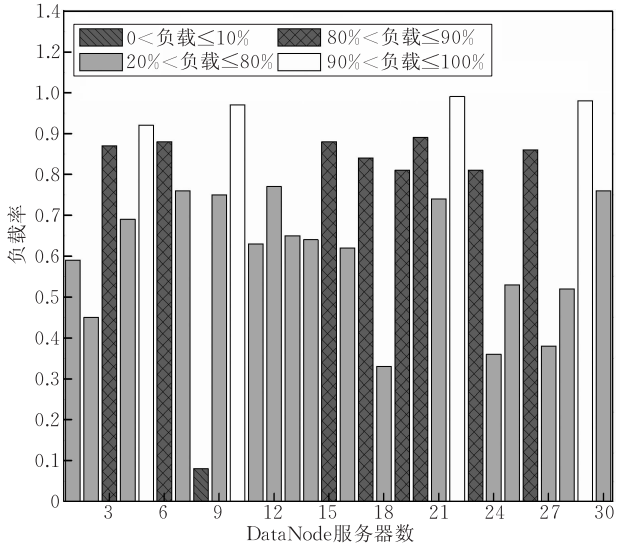
图 11 低负载状态下节能算法执行前后负载对比



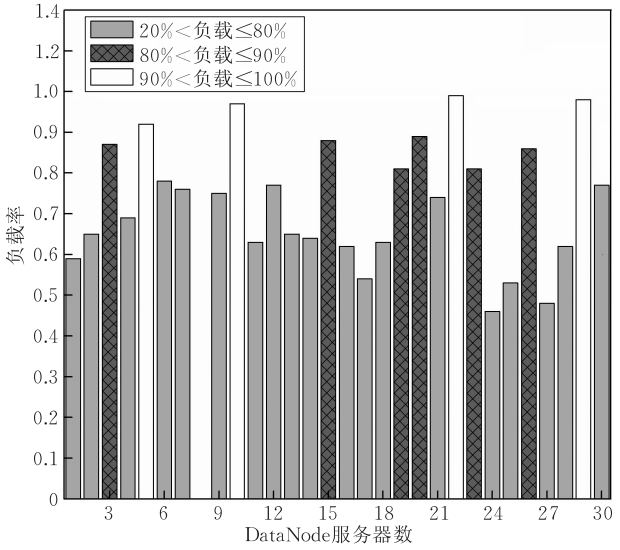
(a) 节能算法执行前 (b) 节能算法执行后

图 12 随机负载状态下节能算法执行前后负载对比

当空闲功耗 σ_e 取值 100，常数 μ_e 与 a 分别取值 70 与 1.5 时，低负载、随机负载与高负载 3 种负载状态功耗如图 14 所示. 其中图 14(a)表示低负载状态下节能算法执行前后节点的功耗；图 14(b)表示随机负载状态下节能算法执行前后节点的功耗；图 14(c)表示高负载状态下节能算法执行前后节点的功耗；图 14(d)表示 3 组负载状态下节能算法执行前后节点的平均功耗对比.

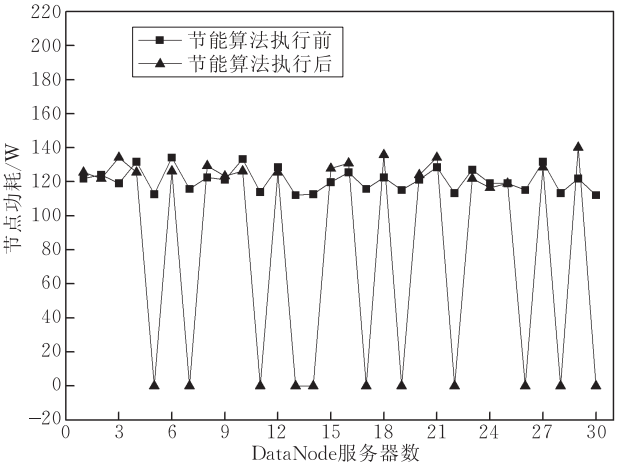


(a) 节能算法执行前

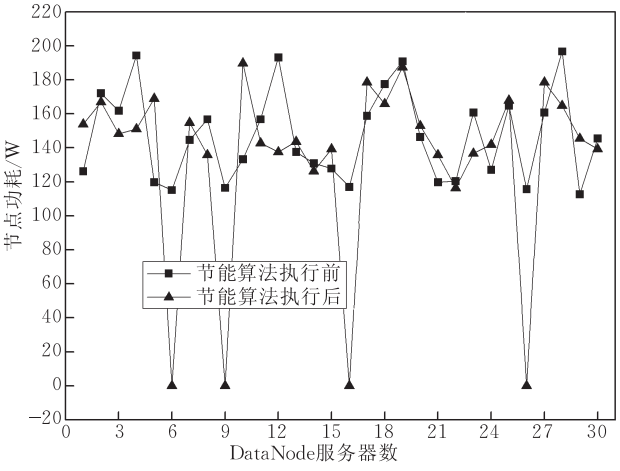


(b) 节能算法执行后

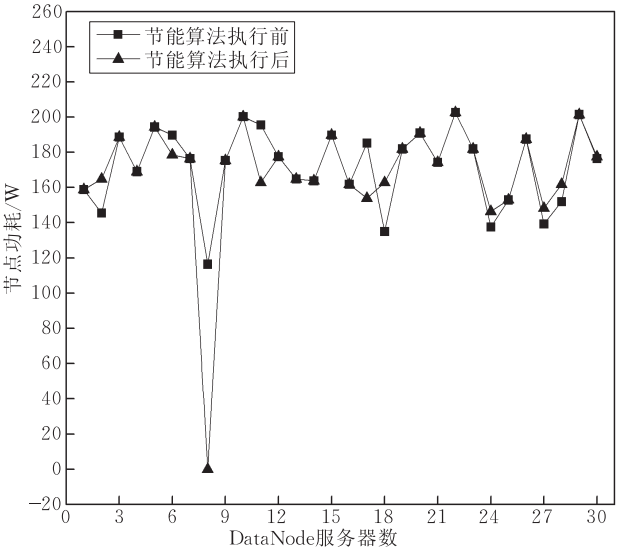
图 13　高负载状态下节能算法执行前后负载对比



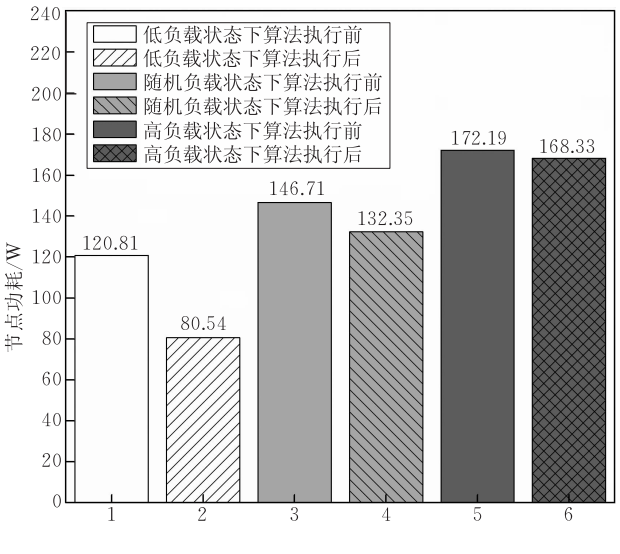
(a) 低负载状态下



(b) 随机负载状态下



(c) 高负载状态下



(d) 3组负载状态下对比

图 14　3 种负载状态下节能算法执行前后 DataNode 节点功耗对比

如图 14 所示,低负载状态下 DataNode 节点平均功耗在节能算法执行后由 120.81 转化为 80.54,功耗节省率为 33.3%;随机负载状态下 DataNode 节点平均功耗在节能算法执行后由 146.71 转化为 132.35,功耗节省率为 9.8%;高负载状态下 DataNode 节点平均功耗在节能算法执行后由 172.19 转化为 168.33,功耗节省率为 2.2%.由此可见功耗节省率与 HDFS 集群负载状态相关,HDFS 集群平均负载越低,功耗节省率越高;相反,HDFS 集群平均负载越高,功耗节省率越低.除此之外,同一集群在相同平均负载率下,功耗节省率还受到节点不同负载率分布的影响,节点负载率分布越均匀,功耗节省率越低,相反则越高.在副本系数为 m 的 DataNode 节点组中,当所有行中有 $m-1$ 个 DataNode 节点处于休眠状态时功耗最低.但是由于任务的转移与替换会加大目标节点的负载,即提高了该节点的功耗,所以每组的最大功耗节省率不会超过 $(1-1/m)$.

7 结论及下一步工作

由于传统 IT 系统超额的资源供给与冗余设计以及负载均衡算法对能耗因素的忽略导致了高能耗低效率问题的日益突出.在 HDFS 集群节能问题上,由于存在数据块可用性的要求,使得 HDFS 并不能简单地采用传统数据中心合并任务与休眠空闲节点的方法解决能耗问题,保证数据块的可用性成为设计 HDFS 集群下节能算法的前提.本文通过研究 HDFS 集群结构、节点与数据块状态与数据块存储机制建立了 DataNode 节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵与数据块状态矩阵,为研究数据块可用性、节能算法等提供了基础模型.利用数据块状态矩阵与数据块可用性之间的关系设计了 DataNode 节点休眠验证算法,算法能够确认 DataNode 节点的休眠是否对数据块的可用性造成影响.通过文件可用性概率分析与证明得出在不改变数据块存储结构与存储策略的情况下并不能通过休眠 DataNode 节点达到节能的目的.所以本文分别从改变数据块的存储结构与存储策略两方面着手对 HDFS 进行节能改进.设计了数据块存储结构配置节能算法与 RACK 区域划分算法,将处于 SleepZone 区域的节点休眠从而达到节能的目的.设计了对称数据块存储策略,解决了节点休眠对数据块可用性的影响问题,对称数据块存储策略下的节能算法将任务转移与替换产生的空闲节点休眠,从而达到节能的目的.通过实验表明两种节能算法都能解决 HDFS 集群的能耗低利用率问题,并且集群负载越低算法节能效率越高.

下一步工作主要集中在以下 3 个方面:(1)节能算法参数的优化问题.由于不用的 HDFS 集群在负载、应用特点等方面存在很大的差异,需要探寻不同环境下节能算法参数的优化方法.(2)分布式文件系统在数据可用性、性能、能耗之间存在相互联系与制约的关系,如何在这三者之间找到合理的平衡点是将来研究的一个方向.(3)分布式文件系统节能的标准化问题.分布式文件系统在节能方面并没有统一的标准与模型,制定统一的节能标准与模型是将来研究的另一个方向.

参 考 文 献

[1] Yun D, Lee J. Research in green network for future Internet. Journal of KIISE, 2010, 28(1): 41-51

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

[3] Ghemawat S, Gobioff H, Leung ST. The Google File System//Proceedings of the 19th ACM Symposium on Operating System Principles (SOSP2003). New York, USA, 2003: 29-43

[4] Dean J, Ghemawat S. MapReduce: Simplifed data processing on large clusters//Proceedings of the Conference on Operating System Design and Implementation (OSDI). San Francisco, USA, 2004: 137-150

[5] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data//Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI). Seattle, USA, 2006: 205-218

[6] Lin Chuang, Tian Yuan, Yao Min. Green network and green evaluation: Mechanism, modeling and evaluation. 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.计算机学报,2011,34(4): 593-612)

[7] Benini L, Bogliolo A, Micheli G D. A survey of design techniques for system-level dynamic power management. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2000, 8(3): 299-316

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

[9] Srivastava M B, Chandrakasan A P, Brodersen R W. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1996, 4(1): 42-55

[10] Hwang C H, Wu A C. A predictive system shutdown method for energy saving of event-driven computation. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2000, 5(2): 241-246

[11] Wierman A, Andrew L L, Tang A. Power-aware speed scaling in processor sharing systems//Proceedings of the 28th Conference on Computer Communications (INFOCOM 2009). Rio, Brazil, 2009: 2007-2015

[12] Andrew L L, Lin M, Wierman A. Optimality, fairness, and robustness in speed scaling designs//Proceedings of the ACM International Conference on Measurement and Modeling of International Computer Systems (SIGMETRICS 2010). New York, USA, 2010: 37-48

[13] Lorch J R, Smith A J. Improving dynamic voltage scaling algorithms with PACE. ACM SIGMETRICS Performance Evaluation Review, 2001, 29(1): 50-61

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

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

[16] Vardhan V, Yuan W, Harris A F, et al. Integrating fine-grained application adaptation with global adaptation for saving energy. International Journal of Embedded Systems, 2009, 4(2): 152-169

[17] Flinn J, Satyanarayanan M. Managing battery lifetime with energy-aware adaptation. ACM Transactions on Computer Systems (TOCS), 2004, 22(2): 137-179

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

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

[20] Gandhi A, Harchol-Balter M, Das R, Lefurgy C. Optimal power allocation in server farms//Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems. New York, USA, 2009: 157-168

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

[22] Raghavendra R, Ranganathan P, Talwar V, Wang Z, Zhu X. No 'power' struggles: Coordinated multi-level power management for the data center. SIGARCH Computer Architecture News, 2008, 36(1): 48-49

[23] Kusic D, Kephart J O, Hanson J E, Kandasamy N, Jiang G. Power and performance management of virtualized computing environments via look ahead control. Cluster Computing, 2009, 12(1): 1-15

[24] Stillwell M, Schanzenbach D, Vivien F, Casanova H. Resource allocation using virtual clusters//Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009). Shanghai, China, 2009: 260-267

[25] Song Y, Wang H, Li Y, Feng B, Sun Y. Multi-Tiered On-Demand resource scheduling for VM—Based data center//Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009). Shanghai, China, 2009: 148-155

[26] Cardosa M, Korupolu M, Singh A. Shares and utilities based power consolidation in virtualized server environments//Proceedings of the 11th IFIP/IEEE Integrated Network Management (IM 2009). Long Island, USA, 2009: 327-334

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

[28] Buyya R, Beloglazov A, Abawajy J. Energy-Efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges//Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010). Las Vegas, USA, 2010: 1-12

[29] Kim K H, Beloglazov A, Buyya R. Power-aware provisioning of cloud resources for real-time services//Proceedings of the 7th International Workshop on Middleware for Grids. Illinois, USA, 2009: 1-6

[30] Chen Y, Keys L, Katz R H. Towards energy effcient Mapreduce. Technical Report UCB/EECS. 2009-109, EECS Department, University of California, Berkeley, 2009

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

[32] Maheshwari N, Nanduri R, Varma V. Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. Future Generation Computer Systems, 2011, 28(1): 119-127

[33] Kaushik R T, Bhandarkar M. GreenHDFS: Towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster//Proceedings of the 2010 International Conference on Power Aware Computing and Systems. Berkeley, USA, 2010: 1-9

[34] Kaushik R T, Bhandarkar M, Nahrstedt K. Evaluation and analysis of GreenHDFS: A self-adaptive, energy- conserving variant of the hadoop distributed file system//Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science. Indianapolis, USA, 2010: 274-287

[35] Andrews M, Anta A F, Zhang L, Zhao Wenbo. Routing for energy minimization in the speed scaling model//Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM'10). San Diego, USA, 2010: 1-9

[36] Brooks D, Bose P, Schuster S, et al. Power-aware micro architecture: Design and modeling challenges for next-generation microprocessors. IEEE Micro, 2000, 20(6): 26-44

[37] Calheiros R N, Ranjan R, Beloglazov A, Rose C A F D, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 2010, 41(1): 23-50

LIAO Bin, born in 1986, Ph. D. candidate. His major research interests include database theory and technology, grid and cloud computing, etc.

YU Jiong, born in 1964, Ph. D., professor, Ph. D. supervisor. His major research interests include network security, grid and distributed computing, etc.

ZHANG Tao, born in 1988, M. S. candidate. Her major research interests include grid and cloud computing.

YANG Xing-Yao, born in 1984, Ph. D. candidate. His major research interests include grid and distributed computing.

Background

With the emergence of cloud computing in the past few years, Hadoop has seen tremendous growth especially for large scale data intensive computing. But the Hadoop Distributed File System (HDFS) presents unique challenges to the existing energy conservation techniques for its rack-awareness replica placement mechanism makes it hard to scale down DataNode servers.

Different from traditional energy-efficiency algorithms in data center, data-dependent computing mechanism of MapReduce makes energy-efficiency algorithm in HDFS must ensure the availability of all data blocks in cluster, that means at least one data block or its replica should in active state. In this paper, cluster DataNode matrix, DataNode status matrix, file block matrix, block storage matrix and block status matrix are created based on the HDFS cluster structure and block storage mechanism. Base on the relationship between the availability of data blocks and its block status matrix, the algorithm for make sure if a DataNode can sleep is designed. The mathematical analysis makes out that it is difficult to save energy in HDFS cluster without changing the data block's storage structure or replica placement mechanism. So the authors design data block storage structure configuration energy-efficiency algorithm and energy-efficiency algorithm under symmetric replica placement mechanism to save the energy consumption from changing and improving of block's storage structure and replica placement mechanism respectively. Mathematical analysis and experiments prove that two energy-efficiency algorithm solve HDFS cluster's high energy consumption but low-efficiency problem, the lower utilization of the cluster the more energy consumption it can save.

This research is supported by the National Natural Science Foundation of China under Grant Nos. 60863003, 61063042 and the Natural Science Foundation Project of Xinjiang Uygur Autonomous Region under Grant No. 2011211A011.