云计算数据中心 HDFS 差异性存储节能优化算法

杨 挺 王 萌 张亚健 赵英杰 盆海波

(天津大学电气自动化与信息工程学院 天津 300072)

摘 要 在云计算的基础设施——数据中心内,Hadoop 分布式文件存储系统(Hadoop Distributed File System,HDFS)以高容错性、高可靠性、高可扩展性的优势被广泛使用.但 HDFS 中遵循机架感知的存储策略没有考虑数据间的差异性和使用频度,所有数据以相同副本数复制后分散存储在不同的 DataNode 节点中,这势必会开启过多的 DataNode 而导致数据中心能耗过高.针对这一问题,突破现有 HDFS 对数据块的恒定副本个数存储的限制,提出保证数据块可用性的可变副本存储策略.建立了分布式文件存储超图模型,数学表述了数据块、文件和 DataNode 间的多对多关系.基于模型提出一种 $\tilde{\kappa}$ 横贯超边计算方法实现数据中心 HDFS 可变 $\tilde{\kappa}$ 重极小覆盖集选择,从而确定保证数据可用性的最小数量 DataNode 开启集合,实现数据中心存储单元节能.在原问题的可行域中会存在多个最优解的情况,即在满足数据块 $\tilde{\kappa}$ 覆盖的条件下,存在开启 DataNode 数目最少且相等的多种方案,因此该问题是一个多态函数优化问题,该文提出采用贪心萤火虫算法加以求解.算法性能测试实验通过 Hadoop 环境下的 WordCount、TeraSort 和 Grep 三种典型计算实例运算实验,进行了数据可用性实验,HDFS 集群存储负载均衡实验,集群能耗分析以及数据中心网络性能试验.实验结果表明,可变 $\tilde{\kappa}$ 数据副本最小覆盖集算法在保证数据块和文件可用的条件下,可以实现更少的 DataNode 开启,有效节省 HDFS 集群能耗,并且通过开启 DataNode 的合理配置,缓解了网络传输拥塞.

关键词 云计算数据中心;分布式文件存储系统;节省能量;超图;$\tilde{\kappa}$ 横贯

中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2019.00721

HDFS Differential Storage Energy-Saving Optimal Algorithm in Cloud Data Center

YANG Ting WANG Meng ZHANG Ya-Jian ZHAO Ying-Jie PEN Hai-Bo

(*School of Electrical and Information Engineering, Tianjin University, Tianjin* 300072)

Abstract In Data center, as the infrastructure of Cloud, Hadoop Distributed File System (HDFS) have been widely used for handling large amounts of data due to their excellent performance in terms of fault tolerance, reliability and scalability. Large size of files stored in the HDFS-based datacenter are split into a number of small size of data blocks, and the default size of each data block is 64M. In order to improve the reliability of data blocks, HDFS creates multiple replicas for each data block in the datacenter. The replicas and the original data blocks will be stored in different data nodes according to the rack-aware storage strategy. With this strategy, if any kind of failure happens to a data node, the availability of data hosted on this physical machine can be guaranteed since its replicas can still be retrieved from other data nodes. However, these storage systems usually adopt the same replication and storage strategy to guarantee data availability, i.e. creating the same number of replicas for all data sets and randomly storing them across data

收稿日期:2017-01-14;日期:2018-01-25.本课题得到国家自然科学基金(61571324)、天津市自然科学基金重点项目(16JCZDJC30900)和国家国际科技合作专项(2013DFA11040)资助.**杨 挺**,男,1979 年生,博士,教授,主要研究领域为云计算数据中心、高性能计算.E-mail: yangting@tju.edu.cn.**王 萌**,女,1994 年生,硕士研究生,主要研究方向为云计算与大数据存储.**张亚健**,男,1991 年生,博士研究生,主要研究方向为云计算.**赵英杰**,男,1993 年生,硕士研究生,主要研究方向为云计算.**盆海波**,男,1989 年生,博士后,主要研究方向为数据中心资源管理.

nodes. Such strategies do not fully consider the difference requirements of data availability on different data sets. More servers than necessary should thus be used to store replicas of rarely-used data, which will lead to increased energy consumption. With the increasing number of datacenters built around the world to maintain cloud computing capabilities, huge amount of electricity bills have to face. To address this issue, this paper studies the HDFS differential storage energy-saving optimal algorithm applying in Cloud Data center. Breaking through the limitation of the constant number of replicas in existing storage methods, we propose a variable number of active replicas storage strategy for each data block according to user requirements of data availability. Firstly, this paper develops a novel hypergraph-based storage model for Cloud data centers, which can precisely represent the many-to-many relationship among files, data blocks, data racks, and data nodes. Based on the hypergraph-based storage model, a $\widetilde{\kappa}$-transverse hyperedge algorithm is proposed to calculate the minimum set of data nodes variable $\widetilde{\kappa}$ covering. Because of just running the minimum number of required data nodes, it can not only save energy for the datacenter, but also maintain full functionality. Analyzing this optimal problem, there is more than one optimal solution in the feasible region. That is, there are multi-solutions with the minimum and equal number of active data nodes to satisfy the data blocks $\widetilde{\kappa}$-coverage constraints. It is a polymorphic function optimizal problem, and this paper proposed a greedy firefly algorithm to solve it. We have also implemented our proposed algorithm in a HDFS based prototype data-center with WordCount, TeraSort, and Grep cloud computing cases for performance evaluation, and the four different aspects, namely, data availability, load balance, energy consumption and network performance of the data center are analyzed. Experimental results show that the variable hypergraph coverage based strategy can not only reduce energy consumption with less number of data nodes active, but can also relieve the delivery congestion problem in data center network.

Keywords cloud data center; distributed file storage system; energy-saving; hypergraph; $\widetilde{\kappa}$-transverse

1 引 言

随着信息技术和新兴产业的快速发展,互联网、物联网和智能电网等业务的数据正以几何级数的形式快速增长,服务业、能源业、制造业、医疗卫生、科教文化等领域都积累了 TB 级、PB 级甚至 EB 级的大数据,例如 Digital Earth(数字地球)项目[1]. 据统计,社交网络 Facebook 中现已存储超过了 500 亿张照片,纽约证券交易所每天由于交易能产生 1 TB 的数据,全球连锁超市沃尔玛每小时需处理 100 余万条用户请求,存储了超过 2.5 PB 的数据[2]. IDC(Internet Data Corporation,国际数据公司)对于 2007～2012 年全球的数据量进行过统计,2007 年全球数据总量约为 165 EB,2012 年时增长至 2.7 ZB,预计到 2020 年时数据总量会突破 35 ZB[3].

建立在云计算[4-5]技术之上的新型数据中心具有高可靠性和方便共享等优势,为大数据的存储和应用提供了技术支撑. 很多企业出于自身业务需求,都开始投建数据中心,但是在提供优质服务的同时,巨大的电能消耗和居高不下的运营成本问题也凸显了出来. 据统计,到 2016 年前后,数据中心规模在 100 个机架以上的约占整个数据中心市场的 60%,这些大型的数据中心电力消耗弥巨. 以 2000 个 Rack 的数据中心为例,每个 Rack 平均功率按 3 kW 计算,数据中心每小时耗电量高达 6000 度,全年耗电量将达到 5.26×10^7 kW·h,加上数据中心配套设施(照明、冷却等)的耗电量,该数据中心全年的电费接近 6000 万元.

斯坦福大学的研究表明,2010 年全球数据中心的耗电量为 2355 亿 kW·h,占据了全球电力消耗的 1.3%左右,其中,美国数据中心耗电总量占全美电能消耗的 2%[6]. 我国的数据中心能耗形势同样严峻,2009 年我国数据中心总耗电量占当年全国耗电量的 1%[7],截止到 2011 年底,我国各类数据中心已达到 43 万个,总耗电量占当年全社会用电总量的

1.5%[8]，截止到 2012 年底，全国数据中心能耗总量已占当年全国工业用电总量的 1.8%. 因此，倡导绿色云计算，建设绿色数据中心已成为学术界和工业界一项极其紧迫的任务和重要的研究方向.

面对海量大数据的处理需求，Hadoop 参考 Google 的分布式存储系统 GFS 设计了新的开源分布式文件存储系统 HDFS(Hadoop Distributed File System)[9]. 由于其具有高容错性、高可靠性、高可扩展性等特征，且易在通用平台部署实现，为超大数据集(Large Data Set)的应用处理带来了很多便利，目前 HDFS 已经在分布式计算领域得到了广泛运用，并且逐渐成为工业与学术界事实上的海量数据并行处理标准.

分布式存储软件框架采用的是副本策略来保证数据的可用性. 然而数据块副本在存储集群中的配置将直接影响数据在存储服务器和计算服务器间的读写速度[10]，并且大量的数据副本不仅消耗了存储资源，也因为更多的存储服务器运行带来集群的高能耗.

在数据中心里，存储数据被使用概率并非是完全一致的，仅有少量数据块被频繁使用，而更多的数据块被读取频率较低[11]. 基于该特性，本文提出在保证数据块可用性的前提下的灵活可变数据块副本存储策略. 并给出一种可变 $\tilde{\kappa}$-横贯超边计算方法完成具有不同副本数需求的数据块的最小 $\tilde{\kappa}$ 重覆盖集选择，进而确定最优的 DataNode 开启集合，实现系统存储单元节能功能. 优化的 DataNode 子集也将减少数据传输对数据中心网络的压力，提升传输效率，进而提升数据的获取速度.

本文的主要贡献包括：

(1) 建立了数据中心 Hadoop 架构的数据存储超图模型，准确表述了文件-数据块和 DataNode 节点-机架间的复杂多对多关系；

(2) 将 HDFS 机架感知存储策略的数据块副本个数恒定准则拓展为在保证数据块可用性的前提下的灵活可变数据块存储策略；

(3) 提出一种可变 $\tilde{\kappa}$ 覆盖最小集算法——$\tilde{\kappa}$-横贯超边计算. 依据最小覆盖集确定最优的 DataNode 开启集合，实现系统存储单元节能，同时降低数据读取/存储通道的负载.

本文第 2 节对分布式节能存储的相关工作进行介绍；第 3 节建立数据中心 HDFS 存储的超图横贯模型；基于超图横贯模型，在第 4 节给出可变 $\tilde{\kappa}$ 覆盖极小集算法；并在第 5 节提出 HDFS 存储负载均衡策略实现数据的动态优化存储；第 6 节实验验证算法的有效性；最后一节对全文进行总结.

2　相关工作

分布式存储系统具有高存储带宽和易扩展的特点，是大数据和云计算的主要存储模式[12]. 然而随着数据量的急剧增加，采用该软件框架所产生的高能耗，低能效问题日益凸显[13]. 为此，部分研究已经开始致力于数据中心存储单元的节能，主要分为两类：改变存储策略和不改变存储策略的节能研究.

在改变存储策略的研究中，文献[14]提出一种弹性的副本复制策略(ERMS)，ERMS 利用一个复杂事件处理引擎来区分出实时数据，然后动态地增加高访问量实时数据的温度，降低低访问量数据的温度，并使用擦除码擦除冷数据以达到节能目的. 文献[15]提出了一种自适应数据复制算法(DARE)，通过利用现有的远程数据访问系统优化文件副本个数和数据定位问题. 文献[16-17]则通过研究 Yahoo 公司 HDFS 集群的数据块访问规律，提出基于数据划分的 Green HDFS 集群，将集群人为地分成 Cold 区和 Hot 区，在 Hot 区的节点驻留着常用数据且具有高功率、高性能的 CPU，集群的负载较轻时可将处于 Cold 区转化成低功耗模式以节省能耗. ebay 的 Hadoop 集群也采用了相似的“数据温度”概念和分层存储技术. 依照数据的使用频度分为“热 HOT”、“温 WARM”、“冷 COLD”、“冷冻 FROZEN”四种属性，依照属性将数据副本在磁盘层和归档层上存储或移动. 文献[18]设计了功率控制器和数据块的迁移策略，提出了一种动态重新配置数据块的放置方法，该算法根据集群的工作负载率高于或低于给定的阈值而开启或关闭 DataNode 节点，从而达到了节能的目的. 学者廖斌等借鉴文献[16-18]的思想在文献[19]中对分布式存储系统下的节能问题进行定义，将 HDFS 集群 Rack 划分为两个存储区域：Active-Zone 和 Sleep-Zone，将 Sleep-Zone 数据迁移到 Active-Zone 中，从而让 Sleep-Zone 中 DataNode 空闲进入休眠状态，达到节能目的. 此外，百度 HDFS 集群采用存储压缩机制，对冷数据和老数据进行周期性压缩，减小系统存储量，也达到降低数据中心资源使用和能耗的效果.

随着对数据中心运行和耗能研究深入，学者发现对数据存储单元的节能策略与其它，如计算部分，

的节能策略有很大区别. 虚拟机整合和迁移技术能够在很小迁移代价下有效提升计算服务器 CPU 利用率,进而获得显著的节能效果. 但对于数据存储单元,数据的迁移将耗费大量系统资源,占用网络资源,并可能由于传输故障造成数据的不完整而失效. 因此对于数据存储单元,迁移并非有效的节能策略. 百度 HDFS 集群存储压缩机制中就提出了本地性,尽量不进行跨数据节点的压缩操作.

基于此思路,近些年学者重点研究了不改变存储策略的节能方法. Harnik 等人[20]分析实际系统发现,数据访问具有很强的周期性,昼夜访问频度差别很大. 基于此,论文通过引入辅助节点,寻求全时段特别是低能耗时段可保证每一个数据项都可用的全覆盖子集,通过关闭覆盖子集外的磁盘或存储节点以降低能耗. Standford 研究组的 Kim 等人在文献[21]中提出将数据块或其副本至少 1 个放在 Covering Set 节点上,并保持 CS 节点开启以确保数据块的可访问性,关闭其它接节点以达到节能目的,但论文仅提出了覆盖子集工作架构,并没有给出覆盖子集的求解策略. 随后,Kim 等人又在文献[22]中给出了求解 1 覆盖子集和 k 覆盖子集的算法. Yazd 等人[23]提出了一种镜像数据块副本的数据存储方法. 同样是先计算极小覆盖集,然后其他节点上的副本放置遵循覆盖节点集上的副本放置规律,即形成镜像. 当覆盖节点集不足以启动一个新任务的时候,使用新的数据块副本镜像,而不需要搜索并启动更多节点,从而降低能耗,但该方法仅考虑了 CPU 的能耗,并假设集群系统是同构的,具有一定的局限性.

因此,目前不改变存储策略的节能方法多是基于完全覆盖集. 然而,现有算法和 HDFS 机架感知策略都未考虑文件和数据被访问的频度差异,以固定的 3 副本数或 1 覆盖率进行计算. 这样的配置,对于高访问频度的文件,单一活跃数据块的多需求端的频繁读取会造成传输通道的拥塞,降低数据读取准确性. 而如果整体提升覆盖率则又会急剧增加开启 DataNode 服务器个数,能耗同步剧增.

针对以上研究不足,本文进行了可变 $\tilde{\kappa}$ 重覆盖的数据中心 HDFS 节能存储方法研究. 通过建立数据中心 DataNode 存储超图模型,准确表述了数据块、文件、任务和 DataNode 节点间的多对多的关系. 进而提出在保证数据块可用性的前提下的可变数据块存储策略,并给出一种可变 $\tilde{\kappa}$ 横贯超边计算方法完成可变覆盖集选择. 则依据文件的访问需求,可动态设定不同数据块的副本个数,依据极小可变 $\tilde{\kappa}$ 覆盖集确定最优的 DataNode 开启集合,实现系统存储单元节能,同时在机架间优化的数据配置,也有效降低了数据读取/存储通道的负载率,提升了数据正确率.

3 HDFS 存储超图横贯模型

3.1 HDFS 存储的超图模型

典型的 HDFS 是由多个机架 Rack 组成,一个机架内部包含多个存储服务器 DataNode,并通过数据中心内部网络实现高速数据交换,如图 1 所示.

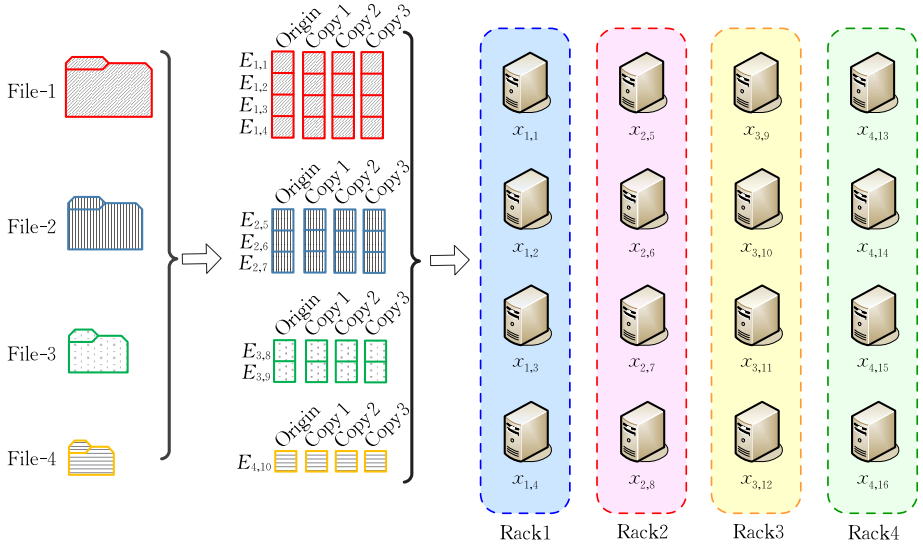

图 1 HDFS 数据文件存储

HDFS 集群中的文件在存储伊始首先被拆分成一系列的数据块，Hadoop 默认设置为 64 M/块. 并且为提升数据的可靠性，HDFS 对同一数据块在相同集群中存储多个副本. 进而将数据块和副本依序机架感知存储策略存放入 DataNode 中.

定义 1(机架感知存储策略).　第一个数据块的第一个副本被随机地存放于某一个数据节点中，第二个副本存放在与第一个数据块不同的机架上的任意一个数据节点中，第三个副本存放在与第二个副本相同的机架但是不同的数据节点中；如果文件的副本系数＞3，剩下的数据块就被随机地存放在除上述三个数据节点以外的任意数据节点中[24].

图 1 表示客户端向集群中写入 4 个数据文件 File-1、File-2、File-3、File-4，每个文件依据文件大小被分割成多个数据块. 例如，File-1 被分割成 $E_{1,1}$、$E_{1,2}$、$E_{1,3}$、$E_{1,4}$ 四个数据块，File-2 被分割成 $E_{2,5}$、$E_{2,6}$、$E_{2,7}$ 三个数据块，而 File-4 大小不足 64 M 则保持一个数据块 $E_{4,10}$. 每个数据块则遵循 HDFS 机架感知存储策略被放置在集群的不同 DataNode 节点中，存储结果表示如图 2 所示.

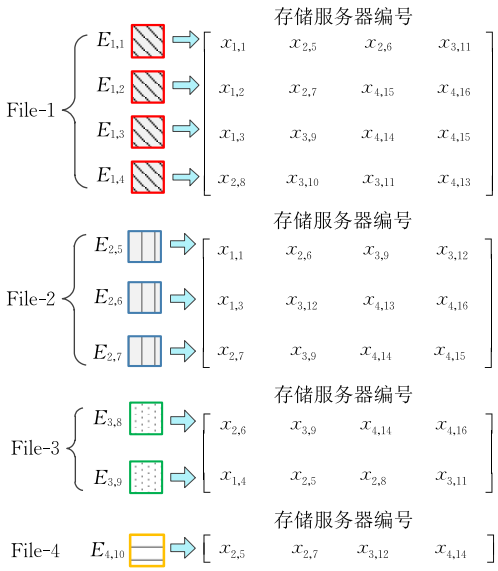

图 2　文件、数据块与 DataNode 节点间的对应关系

由图可见，HDFS 的数据分割-多副本存储方式形成了数据块(副本)、文件和 DataNode 节点、机架间的关联关系，如图 3 所示二部图.

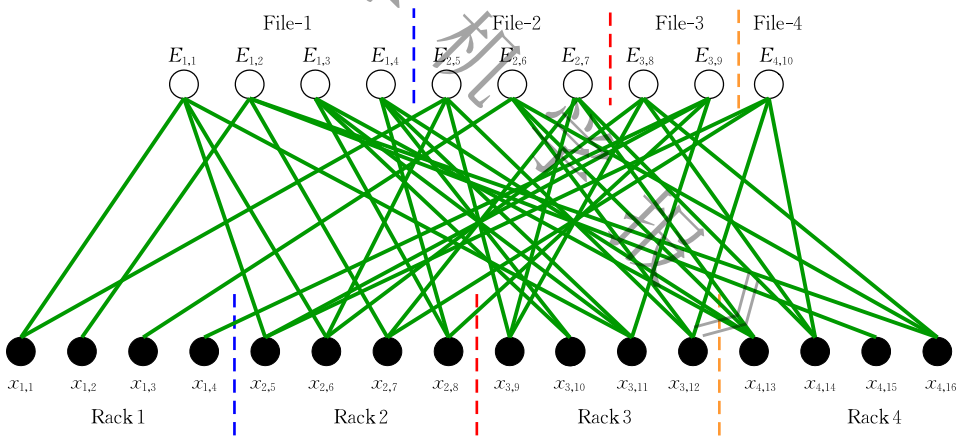

图 3　数据块及其副本与 DataNode 间映射二部图

传统简单图可以准确表示事物间的二元关系，但用以表示文件、数据块和 DataNode 间这种多对多复杂映射关联关系时较为繁琐. 因此，本文提出采用超图理论建立 HDFS 存储结构的超图模型.

定义 2(超图).　令 $X=\{x_1,x_2,\cdots,x_n\}$ 是一个有限集，若 $E_i\neq\Phi(i=1,2,\cdots,m)$ 和 $\bigcup_{i=1}^{m}E=X$，则称二元关系 $H=(X,E)$ 为超图. 其中 X 的元素 x_1，x_2，$\cdots$，x_n 称为超图的顶点，$E=\{E_1,E_2,\cdots,E_n\}$ 为超图的边集合.

在 HDFS 存储结构的超图模型中，映射 DataNode 存储服务器集为超图 H 的顶点集 X，数据块种类映射为超边 E_d，则数据块和 DataNode 节点间多对多关系可通过超图表示：

(1) 数据块 d 和其副本被存储在不同的 DataNode 上，即可表示为该种数据块的超边与节点间关系 $E_d\supset\{x_i,x_j,\cdots,x_k\}$，其中 x_i 代表存储的 DataNode 节点，$x_i\in X$. 而超边 E_d 所包含的节点个数称为超边的秩，$r(E_d)$，即为该数据块和其副本所占据的不同 DataNode 数目.

(2) 某 DataNode 节点 x_i 中所存储的不同数据块种类，即为超图中与顶点 x_i 连接的超边个数，称为顶点 x_i 的度，记为 $d_H(x_i)$.

定义 3(超图关联矩阵).　矩阵 $\boldsymbol{A}$ 表示超图 $H(X,$

E)的关联矩阵,由 $\boldsymbol{A}(a_{ij})$来数学表述:m 列对应 H 的 m 条超边$\{E_1,E_2,\cdots,E_m\}$,n 行对应 H 中包含的 n 个顶点 $X=\{x_1,x_2,\cdots,x_n\}$. 当 $x_i\in E_j$ 时,$a_{ij}=1$;反之 $x_i\notin E_j$ 时,$a_{ij}=0$.

则由定义 2 可得,超图中节点 x_i 的节点度

$$d_H(x_i)=\sum_{j=1}^{m}a_{ij}.$$

3.2 HDFS 可变 $\tilde{\kappa}$ 覆盖问题建模

在 HDFS 中,数据块被访问的频率以及重要性是有差别的. 对 Yahoo 公司 HDFS 集群的数据块访问日志进行分析发现,90.26%的数据会在上传后的两天内产生第 1 次访问,89.61%的数据块会在上传后的十天内产生最后 1 次访问,40%的源数据块会在最后一次访问的 20 天内被删除[25]. 因此研究在保证数据块可用性前提下的灵活可变数据块副本存储策略,可有效节省存储资源,并通过对空闲出的 DataNode 休眠,节省系统能耗.

该问题可数学建模为数据块存储的可变 $\tilde{\kappa}$ 覆盖问题:寻找一个最小 DataNode 子集,集合中覆盖所有种类数据块的 $\tilde{\kappa}$ 个副本. 其中每种数据块的副本数 $\tilde{\kappa}$ 可变,其可基于对数据块访问频率的统计计算活动因子的方法获得[19].

定义 4($\tilde{\kappa}$-横贯). 设超图 $H=(X,E)$,若集合 $\Gamma\subset X$ 与超图 H 中每条超边 E_j 的交集满足:

$$\|\Gamma\cap E_j\|\geq k_j,\ \kappa=\{k_1,k_2,\cdots,k_j,\cdots,k_m\}.$$

则称 Γ 是 H 的一个 $\tilde{\kappa}$-横贯.

因此 HDFS 可变 $\tilde{\kappa}$ 覆盖即为求所对应的超图 H 的一个 $\tilde{\kappa}$-横贯.

若 $k_1=k_2=\cdots=k_m=k$,称为恒定 k-横贯,若 $k=1$ 则为 1-横贯. 因此 1-横贯是 k-横贯的特例,可变 $\tilde{\kappa}$-横贯是 k-横贯的更一般形式. 已有的 HDFS 改进存储策略,例如按完全覆盖集划分活跃区域,本质上是 1-横贯.

定义 5(极小 $\tilde{\kappa}$-横贯). 若 Γ 是 H 的一个 $\tilde{\kappa}$-横贯,当集合 Γ 去除任意节点 x 都会导致 $\|(\Gamma-x)\cap E_j\|<k_j$,则 Γ 是超图 H 的一个极小 $\tilde{\kappa}$-横贯.

因此找到超图 H 的一个极小 $\tilde{\kappa}$-横贯,即确定了满足数据块可用性的极小 DataNode 服务器子集. 则可休眠其它 DataNode,实现 HDFS 极大化节能.

则 HDFS 可变 $\tilde{\kappa}$ 覆盖问题的数学模型为

$$\text{Min}\left\{\sum_{i=1}^{n}\Re_i\right\} \tag{1}$$

$$\text{s.t.}\ \sum_{i=1}^{n}(a_{ij}\cdot\Re_i)\geq k_j,\ j=\{1,2,\cdots,m\}$$

约束条件中,$\Re_i=0$ 或 1,表示第 i 个 DataNode 服务器的工作状态 $i=\{1,2,\cdots,n\}$,$\Re_i=0$ 表示存储服务器 i 休眠,反之 $\Re_i=1$ 表示存储服务器 i 被选中开启.

4 HDFS 可变 $\tilde{\kappa}$ 覆盖极小集算法

由第 3 节分析可知,求解 HDFS 可变 $\tilde{\kappa}$ 覆盖问题等价于求解超图最小 $\tilde{\kappa}$-横贯问题,是 NP-Hard 问题[26]. 具体分析原问题发现,可行解中会存在多个最优解的情况,即在满足数据块 $\tilde{\kappa}$ 覆盖的条件下,开启 DataNode 数目最少且相等的多种方案,只是具体开启哪些 DataNode 有所不同. 因此该问题是一个多态函数优化问题. 本文采用贪心萤火虫算法加以求解.

4.1 FA 算法执行流程分析

萤火虫算法(Firefly Algorithm,FA)源于模拟自然界萤火虫在晚上的群聚活动的自然现象而提出的一种仿生群智能优化算法. 萤火虫种群作为初始解随机的分布在搜索空间中,每只萤火虫被视为该解空间的一个解. 然后根据自然界萤火虫散发荧光素越亮的其号召力越强,也吸引更多的同伴向它靠拢的移动特点,进行解空间中每只萤火虫的移动. 最终会聚集到较亮的萤火虫周围,即是找到解空间内多个极值点,从而达到种群寻优的目的. 由此可见,FA 算法本质即具有在解空间并行寻找多个优点的特性,极其适合求解多态函数优化问题.

在采用 FA 求解超图最小 $\tilde{\kappa}$-横贯问题时,将目标函数设定为萤火虫的绝对亮度.

定义 6(绝对亮度). 对于萤火虫 i,$r=0$ 处的初始光强为该萤火虫的绝对亮度 I_i.

因为原问题是求极小 $\tilde{\kappa}$-横贯,为保持和 FA 算法的萤火虫向最大亮度聚集一致. 变换原目标函数为

$$I_i=\text{Max}\left\{N-\sum_{i=1}^{n}\Re_i\right\} \tag{2}$$

其中,N 为数据中心 DataNode 总数.

萤火虫 j 接收的荧光强度 I 与距离光源的距离 r_{ij} 的平方存在反比例关系,且空气会吸收光强,用函数来准确表述萤火虫相对亮度 I_{ij}.

定义 7(相对亮度). 萤火虫 i 在萤火虫 j 处的光强度为 i 对 j 的相对亮度 I_{ij}.

$$I_{ij}(r_{ij})=I_i\,\mathrm{e}^{-\gamma r_{ij}^2} \tag{3}$$

式中,γ 为光介质吸收因子,一般取 1;r_{ij} 为萤

火虫 i，j 间的距离，用笛卡尔距离表示，即

$$r_{ij}=\|w_i-w_j\|=\sqrt{\sum_{k=1}^{d}(w_{i,k}-w_{j,k})^2} \tag{4}$$

假设萤火虫间的吸引度正比于相对亮度 I_{ij}，则由式(3)相对亮度的定义可得萤火虫 i 对 j 的吸引度 $\beta_{ij}(r_{ij})$ 为

$$\beta_{ij}(r_{ij})=\beta_{\min}+(\beta_0-\beta_{\min})\mathrm{e}^{-\gamma r_{ij}^2} \tag{5}$$

式中，萤火虫 i 对其距离为 0 处的吸引度定义为 β_0，萤火虫 i 对萤火虫 j 的最小吸引度定义为 $\beta_{\min}$，在此 $\beta_0=1$，$\beta_{\min}=0.2$.

当萤火虫 j 被更强亮度吸引向其靠近，直至聚集稳定在极值点附近. 萤火虫 j 每次移动的更新位置为

$$\boldsymbol{w}_j(t+1)=w_j(t)+\beta_{ij}(r_{ij})\cdot(w_i(t)-w_j(t))+\alpha\cdot\varepsilon_j \tag{6}$$

式中，t 为 FA 的迭代次数，$w_i(t)$ 和 $w_j(t)$ 分别为第 t 次迭代时萤火虫 i 和 j 所处的位置，α 为 $[0,1]$ 间的常数，$\boldsymbol{\varepsilon}_j$ 为是一个服从高斯分布或均匀分布的随机向量. 考虑到 DataNode 状态为开启/休眠两状态的布尔型，即 0/1 两种状态. 则以 0.5 为界，若解向量 $\boldsymbol{w}_j(t+1)$ 的第 k 个元素 $w_{j,k}(t+1)\leq 0.5$，则令 $w_{j,k}(t+1)=0$，反之，若 $w_{j,k}(t+1)>0.5$，则令 $w_{j,k}(t+1)=1$.

4.2　FA 算法贪心策略

注意到本文的 HDFS 可变 $\tilde{\kappa}$ 覆盖的数学模型式(1)是约束优化问题. 即并非整个 N 维欧式空间 R^n 都是可行域. 因此初始萤火虫配置和萤火虫移动都有位于不可行域的危险. 在此，我们采用贪心策略加以修正.

检查当前时刻 t 的萤火虫 i 的位置 $w_i(t)=\{w_1,w_2,\cdots,w_n\}$，如果其代表的 DataNode 的开闭状态导致 m 个约束 $\sum_{i=1}^{n}(a_{ij}\cdot\Re_i)\geq k_j$，$j=\{1,2,\cdots,m\}$ 中任意一个不满足，则启动贪心策略加以修正：

将此解中每台服务器存储的数据块总量按照降序排列，由大到小的方向依次将 $w_a=0$ 置换为 $w_a=1$（$a=1,2,\cdots,n$），直至该解满足约束集 $\sum_{i=1}^{n}(a_{ij}\cdot\Re_i)\geq k_j$，$j=\{1,2,\cdots,m\}$，即成为可行解.

FA 贪心策略有效解决了具有多约束的极小 $\tilde{\kappa}$-横贯优化问题，使得计算中每个解都保持可行解.

4.3　FA 算法变异策略

研究发现 FA 算法能够较好的求解多态函数优化问题，可快速稳定在多个极值点处. 但也正是因为此特点，导致算法易陷入局部最优解而失去寻找全局最优解的动力. 为解决该问题，本文引入变异策略，小扰动局部萤火虫个体，跳出局部极值点，去寻找最优解. 对于二进制编码方式的解群，变异策略以极低的算法复杂度即可实现. 按照变异概论 $P_{mu_1}=0.01\%$ 在每代解群中选择待变异个体 w_{mu}，随后对萤火虫的每一位二进制编码以变异概论 $P_{mu_2}=0.001\%$ 进行变异，即将 $0\leftrightarrows 1$ 翻转.

4.4　贪心 FA 算法执行流程

基于上述算法表述，给出贪心 FA 算法求解可变 $\tilde{\kappa}$ 覆盖集的伪代码如表 1 所示.

表 1　贪心萤火虫算法求解伪代码

```
Algorithm：贪心萤火虫算法(Greedy FA)
1. 目标函数：Max{N−Σ_{i=1}^{n}ℜ_i}
2. 输入：HDFS 存储结构对应的超图模型关联矩阵 A(a_ij)，初始
   化算法参数：M，α，β_min，β_0，γ
3. 输出：最亮萤火虫 gbest(x_i)，其 0-1 编码代表最小覆盖集
执行：
4. 产生萤火虫初始种群{w_1，w_2，…，w_M}，w_i=Randi([0,1]，M，n)
5. 执行贪心策略：检查每个初始萤火虫代表的解是否是可行解，
   即满足约束集Σ_{i=1}^{n}(a_ij·ℜ_i)≥k_j，j={1，2，…，m}，非可行解执行
   贪心策略修复为可行解
6. While (dr<Max_Dr) //种群内个体未足够聚集
7.   for i=1：M
8.     计算萤火虫 w_i的绝对亮度 I_i=Max{N−Σ_{i=1}^{n}ℜ_i}；
9.   for i=1：M
10.    for j=1：M
11.      if(I_i>I_j)
12.        依照式(4)计算萤火虫 i 对萤火虫 j 的笛卡尔距离
13.        进而依式(5)计算萤火虫 i 对萤火虫 j 的吸引度
14.        萤火虫 j 依式(6)向 i 移动，更新萤火虫 j 位置
15.      end if
16.  for i=1：M
17.    执行变异策略；//小概率增加种群多样性
18.    执行贪心策略；//保证解为可行解
19. end while
20. return gbest(w_i) //最亮萤火虫 w_i所表示的最小覆盖集
```

4.5　HDFS 存储机架选择优化策略

如前分析可知，HDFS 存储可变 $\tilde{\kappa}$ 覆盖问题是多态优化问题，存在满足数据块 $\tilde{\kappa}$ 覆盖的条件下，开启 DataNode 数目最少且相等的多种最优方案. 回到数据中心节能优化存储问题本身，我们为进一步优化 HDFS 性能，设计了 2 项进一步优化策略：

(1) 在具有相同 $\sum_{i=1}^{n}\Re_i$ 的两个或多个覆盖集中，计算节点度之和 $\sum_{i=1}^{n}d_H(x_i)|_{\Re_i=1}$，选择最小者为配置方案. 此时可使得开启相同数目的 DataNode 服务器中存储数据块总数最小. 这将有利于读写操作，

并且为后续输入数据提供更多的存储空间.

(2) 当具有相同的$\sum_{i=1}^{n}\Re_i$和$\sum_{i=1}^{n}d_H(x_i)|_{\Re_i=1}$情况下,选择占据最少的 Rack 方案. 这将能够通过关闭更多的 Rack 和配套冷却设备进一步降低能耗.

最终可通过修改 Hadoop 源代码重新编译实现 HDFS 支持弹性副本机制如图 4 所示.

```
〈code class="language-Java"〉LocatedBlocks
getBlockLocations(String clientMachine, String src, long
offset,long length) throws IOException
{
LocatedBlocks blocks = getBlockLocations( src, offset,
length, true, true, true);
    if (blocks!=null)
    {
     /* if it is available to all of the data replicas,
established the hyper-graph H=(X,E); */
    /* Call the κ̃-transverse of the hypergraph
algorithm, and then calculate the minimal variable κ̃-
replica coverage set; */
     getTransverseHypergraph(String src, long offset,
long length, long k, String H);
    }
    public LocatedBlocks getBlockLocations(String src,
long offset, long length
    throws IOException
    {
    return getBlockLocations(src, offset, length, false,
true, true);
    } /* return block locations within the specified
range */
}
```

图 4 HDFS 源码修改支持弹性副本机制

4.6 贪心 FA 算法复杂度分析

已证明超图中求解 $\tilde{\kappa}$-横贯是 NP-Hard 问题,即 HDFS 存储可变 $\tilde{\kappa}$ 覆盖问题为 NP-Hard 问题,不存在多项式时间复杂度内求解出全局最优解的有效算法. 只能采用枚举法遍历整个解空间寻找全局最优解,其复杂度为 $O(2^N)$. 这对于大型云计算数据中心含有 DataNode 个数 N 巨量时,是无法实用化的.

分析本文提出的贪心 FA 算法,其迭代过程是两两萤火虫进行绝对亮度比较,更新所在位置. 则当有 M 个萤火虫时,其算法复杂度为 $O(M^2)$,设直至算法终止迭代共进行了 G 代迭代,因此复杂度为 $O(G\cdot M^2)$,明细比枚举法具有更小的复杂度.

我们在第 6 节实验评测中也对贪心 FA 算法和枚举法求解相同规模算例的计算性能进行评价,可以看到贪心 FA 算法同样达到了全局最优解,并只是枚举法计算时间的万分之一($1/10^5$). 从数值计算角度验证了本文所提出贪心 FA 算法的全局寻优性和低算法复杂度.

5 HDFS 存储负载均衡策略

本文考虑数据中心实际运行情况:对于优化后持续到来的数据存储. 在采用寻找覆盖集而休眠补集中 DataNode 的方法中,休眠期间节点则不能进行数据存储,否则唤醒过程又将重新增加能耗. 但如果对于后续到达数据一直存储在极小覆盖集中,则会造成集群负载不均衡的问题. 针对该问题,我们提出轮换递进存储策略:

(1) 设定轮换周期,为 DataNode 设定一标示位 λ;本周期伊始,已确定的可变 $\tilde{\kappa}$-覆盖集合 Γ_{cyc}^{g} 中 DataNode 的 $\lambda=1$,覆盖集补集 $\overline{\Gamma_{\text{cyc}}^{g}}$ 中 DataNode 的 $\lambda=0$,其中 g 为轮计数.

(2) 在周期 g 内新到来的数据按照机架感知策略存储在 Γ_{cyc}^{g} 中,则周期内 $\overline{\Gamma_{\text{cyc}}^{g}}$ 休眠 DataNode 无需重新开启,保证节能效果.

(3) 在该周期结束时,首先对本周期内新到数据运行可变 $\tilde{\kappa}$-覆盖集算法,则求出一个针对新数据的可变 $\tilde{\kappa}$-覆盖子集 $\Gamma_{\text{new-data}}^{g+1}$.

(4) 检索 $\Gamma_{\text{new-data}}^{g+1}$,调整旧数据的可用性集合:

$$\tilde{\kappa}_{\text{old-data}}^{g+1}=\tilde{\kappa}_{\text{old-data}}^{g}-\tilde{\kappa}(\Gamma_{\text{new-data}}^{g+1}).$$

即下一轮计算需要保证旧数据保留副本个数 $\tilde{\kappa}_{\text{old-data}}^{g+1}$ 为原始要求 $\tilde{\kappa}_{\text{old-data}}^{g}$ 减去已在新数据的可变 $\tilde{\kappa}$-覆盖子集 $\Gamma_{\text{new-data}}^{g+1}$ 中的.

(5) 以 $\kappa_{\text{old-data}}^{g+1}$ 为约束,搜索范围是 $X-\Gamma_{\text{new-data}}^{g+1}$,进行可变 $\tilde{\kappa}$-覆盖集计算. 对求出 $\{\Gamma_{\text{old-data}}^{g+1}\}_u$ 解集进行选优判断,条件为 $\min\sum_{i\in\Gamma_{\text{old-data}}^{g+1}}\lambda_i$,保证 g 轮未被开启的 DataNode 更多的进入 $g+1$ 轮参与存储新到达数据,实现数据在集群中的存储负载均衡. 则下一轮 $g+1$ 的可变 $\tilde{\kappa}$-覆盖集合 $\Gamma_{\text{cyc}}^{g+1}=(\Gamma_{\text{old-data}}^{g+1})\bigcap(\Gamma_{\text{new-data}}^{g+1})$.

以此轮换递进过程可以实现每个周期内新到来数据仍能够以相同的多副本机架感知存储,并且充分保证在整个运行过程中集群内存储负载均衡.

6 实验与性能评价

6.1 实验环境

对所提出 HDFS 可变 $\tilde{\kappa}$ 覆盖节能算法进行性能评测. 实验选用 WordCount、TeraSort 和 Grep 三种典型任务进行运行,测评算法对数据的可用性,以及对集群节能和运算性能提升效果. WordCount 是

典型的 MapReduce 类计算，TeraSort 是 Hadoop 中对原始数据的排序工作的典型任务，Grep 是对指定文档中指定单词的词频进行计算的另一类典型任务. 实验中我们不仅观测 Map 计算和 Reduce 计算过程的系统耗时和能耗，还通过 NS-2 网络仿真平台测试新的存储方法对数据读取/传递过程的性能提升效果.

为验证算法的普遍适用性，搭建数据中心最常用的 3 种集群结构，所含有 DataNode 数量也逐级递增：

(1) Fat-Tree 拓扑结构. 集群由 6 个机架组成，每个机架包含 9 个 DataNode 节点，共计 54 节点.

(2) $BCube_2$ 拓扑结构. 集群由 4 个机架组成，每个机架包含 16 个 DataNode 节点，共计 64 节点.

(3) $DCell_2$ 拓扑结构. 集群由 5 个机架组成，每个机架包含 20 个 DataNode 节点，共计 100 节点.

单个 DataNode 节点的配置参数如表 2 所示，图 5 给出了集群实物图.

表 2　DataNode 节点配置参数

参数	设定值
CPU	Intel 2.7 GHz/2 cores
OS 版本	Ubuntu12.04
Hard Disk	SATA3.0(6 Gbps)
Java 版本	1.6 for Linux
Hadoop 版本	2.6.5
DataNode 最小运行功耗	168.2 W
DataNode 最大运行功耗	344.8 W

图 5　集群实物图

6.2　数据可用性实验

保证存储数据可用性是数据中心存储的首要任务，本节从数据块和任务两方面讨论 HDFS 集群采用可变 $\tilde{\kappa}$ 覆盖集算法后，休眠 DataNode 节点数目对于存储数据可用性的影响，并与机架感知存储策略和采用计算恒定覆盖率集合优化策略[22]比较.

实验中选用三种不同类任务. $Task_1$：{$Size$=0.25 GB，$Num\ of\ files$=1}，$Task_2$：{$Size$=1 GB，$Num\ of\ files$=1}，$Task_3$：{$Size$=4 GB，$Num\ of\ files$=4}. 整个集群存储了16个 $Task_1$，12 个 $Task_2$，4 个 $Task_3$. 存储伊始按照机架感知的存储策略：对每个文件进行以 64 M 为单位的数据块分割，并为每个数据块保存 3 个副本存储在不同的 DataNode 服务器上，即共有 512 种共计 2048 个数据块随机分配存储在 54 个 DataNode 服务器上.

实验中依照数据访问频度设定数据块可用性：属于 16 个 $Task_1$ 的 16×4=64 种数据块需要同时保留 3 份副本，属于 12 个 $Task_2$ 的 12×16=192 种数据块需要保留 2 份副本，属于 4 个 $Task_3$ 的 4×64=256 种数据块设定需要保留 1 份原数据.

图 6 和图 7 分别给出集群关闭 DataNode 节点数与数据块可用百分比和任务完成百分比之间的关系. 定义：

数据块可用百分比：若关闭的 DataNode 节点导致集群中任务数据块的不再满足可变 $\tilde{\kappa}$ 副本要求，则认为该数据块不可用，计算剩余可用数据块占总数据块的百分比；

任务完成百分比：由于数据不可用导致任务无法顺利完成，统计剩余可完成的任务数占总任务数的百分比.

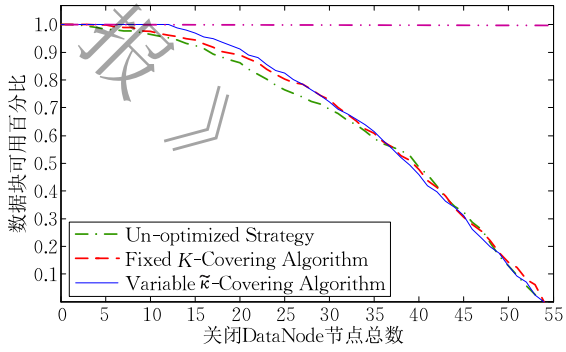

图 6　集群关闭 DataNode 节点数与数据块可用性关系

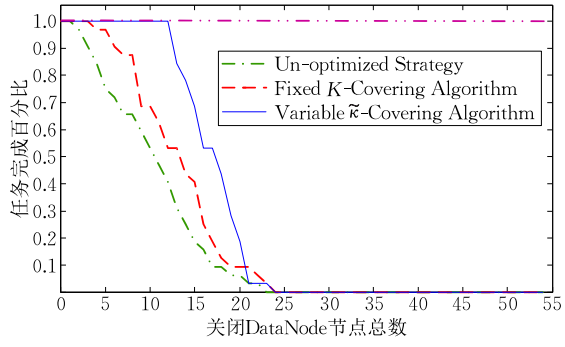

图 7　集群关闭 DataNode 节点数与任务可用性关系

由图 6、图 7 可知，随着存储节点的关闭，数据块的可用性也随之减少，而对 Task 任务能否完成的影响更重. 不采用优化策略时，随机关闭集群中任意一个 DataNode 节点都有可能无法满足数据块的可用性需求，同时也必然破坏任务的完成率；当采用文献[22]中的 iPACS 固定覆盖算法时，为了保证所有种类数据块都满足可用性要求且任务均能顺利完成，则需要选择最大覆盖率 $k=3$，此时可以关闭 3 个 DataNode 节点；而依循数据可用性需求的可变 $\tilde{\kappa}$ 覆盖算法则可实现多达 12 个 DataNode 节点的关闭，同时仍然保证所有数据块可用和计算任务正常完成.

本文还基于 Google Trace 数据[27]对算法进行性能评价. 保证实验的普适性，我们从 Google Trace 中选取了不同大小的 Tasks：64 组 256 M 的小容量 Task_S，48 组 1026 M 的中等容量 Task_M，16 组 4093 M 的大容量 Task_L. 设定 Task_S 需要同时保留 3 份副本，Task_M 需要保留 2 份副本，Task_L 保留 1 份原数据. 存储集群扩展到 8 个机架组成，每个机架包含 16 个 DataNode 节点，共计 128 节点，由 8 口交换机采用 Fat-Tree 拓扑连接. 图 8 和图 9 给出集群关闭 DataNode 节点数与数据块可用百分比和任务完成百分比之间的关系.

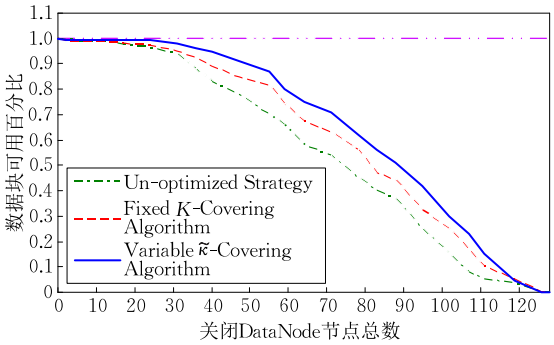

图 8 集群关闭 DataNode 节点数与数据块可用性关系

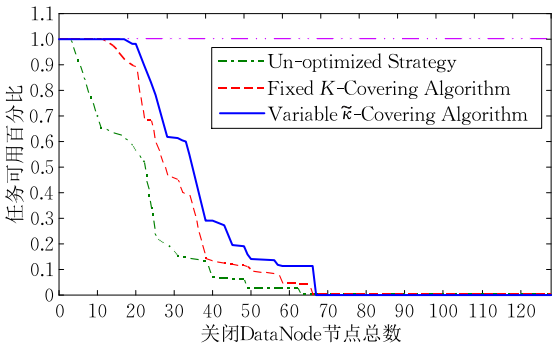

图 9 集群关闭 DataNode 节点数与任务可用性关系

由图 8 和图 9 可知，依循数据可用性需求的可变 $\tilde{\kappa}$ 覆盖算法仍然表现出良好的性能. 在保证所有数据块可用和计算任务正常完成的约束下，仍然能够关闭 19 个 DataNode 节点. 与之相比，采用 iPACS 算法仅能够关闭 11 个 DataNode 节点，而不采用优化策略则仅有 3 个 DataNode 节点空闲.

6.3 HDFS 集群存储负载均衡实验

本节通过实验评测对于持续到达新数据，本文所提出的 HDFS 集群存储负载均衡策略性能. 实验分为独立的 4 组，初始 HDFS 已存储 512 种数据块，数据块的副本个数和可用性需求同 6.2 节. 从第 2 个存储周期开始，每个周期内向集群新存入的数据块种类个数分别为 32、64、128、256 四种，模拟不同的数据存储到达速率，可用性需求设定为 1 活跃副本. 实验对比采用本文提出的存储负载均衡策略和保持将新到达数据一直存入当然开启 DataNode 的性能.

实验表明，采用本文提出的存储负载均衡策略，4 组存储数据到达率情况下，均能够在 3 轮完成所有 DataNode 开启一遍. 而对比策略使得休眠 DataNode无法再次启动，将造成数据在 HDFS 集群的存储不均衡问题，并且随着集群运行时间增加，该不平衡度逐步加剧.

图 10 采用集群中各 DataNode 存储量间的标准差 δ 衡量存储负载不均衡度. 实验结果可知，采用本文所提出算法很好的降低了存储负载不均衡度，且随着数据到达率提升，效果更明显. 在每轮达到 32 种数据块时，$\Delta\delta=\delta_{\text{un}}-\delta_{\text{lb}}=1.4229$，当每轮数据块种类达到 256 时，$\Delta\delta=13.4843$.

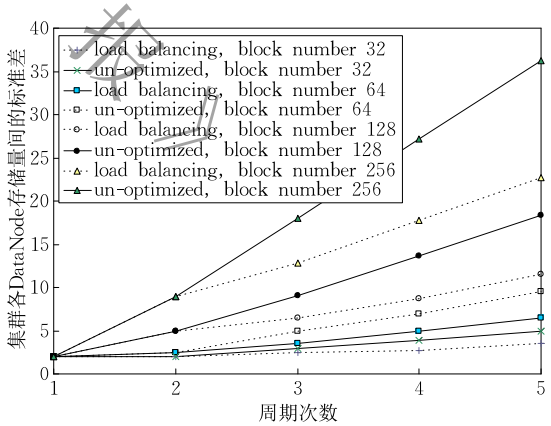

图 10 DataNode 存储不均衡对比

6.4 HDFS 集群能耗分析

由 6.2 节实验可知，在保证数据可用性需求条件下本文的可变 $\tilde{\kappa}$ 覆盖最小集算法能够让集群中更多 DataNode 节点休眠，休眠节点以低功耗运行能够大幅降低集群能耗.

除此之外，我们还将通过实际运行 Hadoop 基准测试程序—WordCount 来测试不同存储方法对数据中心计算节能效果. 为了验证本文提出算法对

于不同架构和规模的 HDFS 集群的节能效果，实验选用 6.1 节中所列 3 类集群架构：Fat-Tree(54 台服务器)、$BCube_2$(64 台服务器)和 $DCell_2$(100 台服务器)．每类集群独立运行 8 组不同 WordCount 任务，其搜索计词规模从 8～1024 个数据块以 2 的阶乘递增．实验仍然设定不同种类数据块的可用性：需保留 1 个活跃副本、2 个活跃副本与 3 个活跃副本的数据块数量之比为 4∶3∶1．保证数据块满足可用性要求的基础上，针对依次递增的任务负荷分别采用恒定 3 重覆盖算法[22]和可变 $\tilde{\kappa}$ 覆盖最小集算法进行存储优化．

由于休眠服务器是最直接节能的方法，因此首先我们统计在三种集群中执行不同规模的 WordCount 任务，两种覆盖集算法最多所能关闭的 DataNode 数目，如图 11 所示．

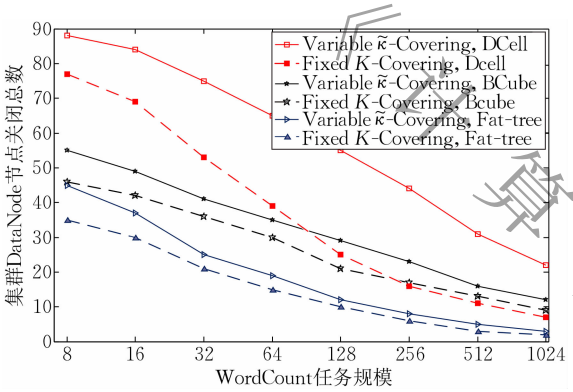

图 11　执行不同规模的任务时，不同集群关闭的节点数

由图 11 实验结果可知，无论是仅搜索 8 数据块的轻计算载荷还是 1024 数据块的繁重计算载荷，可变 $\tilde{\kappa}$ 覆盖最小集算法都比恒定 3 重覆盖算法(Covering Set 3，CS-3)关闭更多的 DataNode 节点，且在 3 种集群结构中都优势稳定．这是因为可变 $\tilde{\kappa}$ 覆盖集算法能够完全符合数据可用性需求而找到 $\tilde{\kappa}$-覆盖子集．同时还注意到，随着集群服务器规模增大，采用可变 $\tilde{\kappa}$ 覆盖集算法比 CS-3 重覆盖算法关闭更多的 DataNode 节点．在执行 1024 数据块的重载荷计算时，Fat-Tree(54 台服务器)集群采用本文算法比 CS-3 重覆盖算法多关闭 12%的服务器，而 $DCell_2$(100 台服务器)集群则可多关闭 15%的服务器．

计算集群能量消耗，服务器功耗由基态功耗与动态功耗组成．基态功耗是服务器在空闲状态的基础功耗，用 P_{idle} 表示，动态功耗与 CPU 使用率密切相关，Fan 给出了非线性模型[28]：

$$P_{\text{sever}}=P_{\text{idle}}+(P_{\max}-P_{\text{idle}})\times(2u-u^r) \tag{7}$$

其中，P_{sever} 是单台服务器的总功耗；$P_{\max}$ 是服务器在满负荷运行下功耗；u 为 CPU 使用率；r 为校准系数，其能够最小化理论与实际功耗的方差，不同服务器 r 不同，由实验测试获得．文献[29-30]等多篇文献表述，服务器的基态功耗占最大功耗的 50%～70%．并且通过对上千台服务器的实验，式(7)能够精确拟合实际服务器能耗，误差仅在 1%～5%．

单个服务器能耗 E_i 和整个集群的能耗 E_{total} 可由如下公式计算：

$$E_i=\int_{T_i^{\text{star}}}^{T_i^{\text{end}}} P_i\cdot t_i\,\mathrm{d}t \tag{8}$$

$$E_{\text{total}}=\sum_{i\in \text{open servers}}\left(\int_{T_i^{\text{star}}}^{T_i^{\text{end}}} P_i\cdot t_i\,\mathrm{d}t\right) \tag{9}$$

其中，P_i 即是第 i 个服务器的总功耗 P_{sever}；t_i 为第 i 个服务器的运行时间$[T_i^{\text{star}},T_i^{\text{end}}]$．

图 12、图 13 和图 14 分别给出三种结构集群执行不同载荷 WordCount 任务时的能耗．由实验数据统计图可知，可变 $\tilde{\kappa}$ 覆盖算法进行配置后，数据中心存储能量损耗在 3 种网架结构中不同的负载的实验场景下始终是最小的．为测试数据中心面对更多样化的任务，本文选用 TeraSort、Grep 和 WordCount 三种任务运行．图 15 给出不同载荷下 CS-3 重覆盖算法和可变 $\tilde{\kappa}$ 覆盖最小集算法的集群节能效率．轻载荷场景下，可变 $\tilde{\kappa}$ 覆盖最小集算法比 CS-3 重覆盖算法的节能效率高 30.15%，比不优化存储节能最高可达 87.01%；重载荷时，比 CS-3 重覆盖算法节能效率高 7.21%，比不优化存储节能最高可达 18.05%，如表 3 所示．

表 3　集群节能效率

	多任务轻载荷		多任务重载荷	
Fat-Tree	30.15%	79.01%	5.36%	12.83%
$BCube_2$	17.79%	83.12%	4.47%	14.22%
$DCell_2$	14.53%	87.01%	7.21%	18.05%

注：每种计算载荷中左侧为可变 $\tilde{\kappa}$ 覆盖算法比 CS-3 重覆盖算法的节能效率，右侧为可变 $\tilde{\kappa}$ 覆盖算法比初始 HDFS 机架感知存储方法的节能效率．

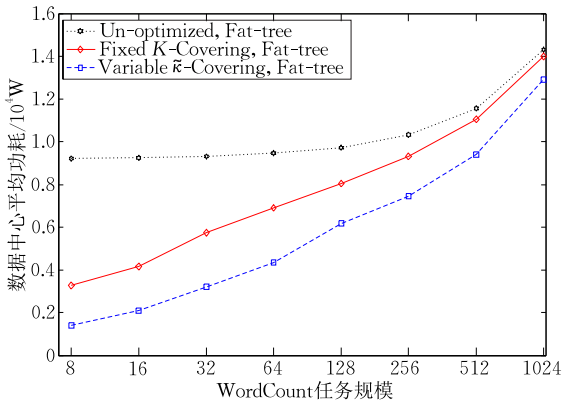

图 12　执行不同规模的任务时，Fat-tree 集群平均功耗

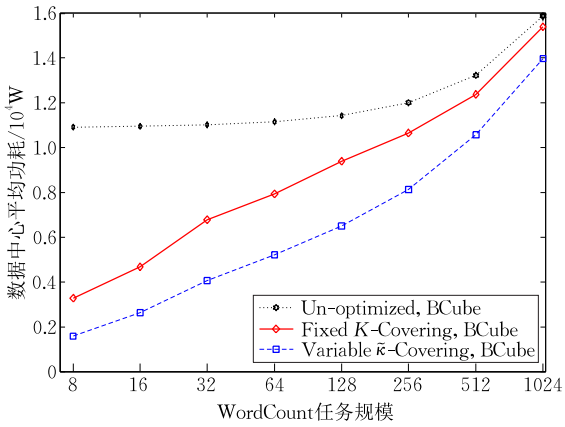

图 13 执行不同规模的任务时，$BCube_2$ 集群平均功耗

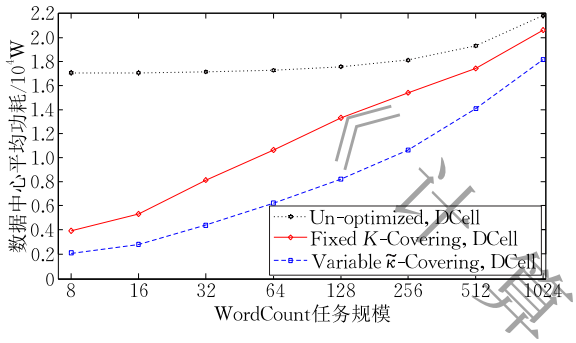

图 14 执行不同规模的任务时，$DCell_2$ 集群平均功耗

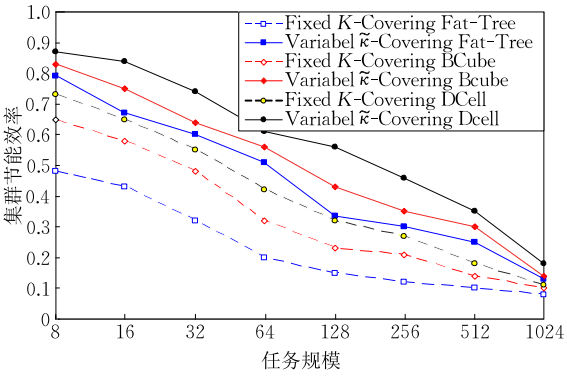

图 15 执行不同规模的多任务时，集群节能效率

6.5 HDFS 集群网络性能测评

对于云计算，数据中心网络传输性能是影响数据在集群中准确读/写的关键因素之一. 可变 $\tilde{\kappa}$ 覆盖集算法优化配置数据的存储，在降低集群能耗的同时，还因为减少了数据的存储位置，即数据交互的源-宿，以及传输通道，可有效缓解数据中心在执行 MapReduce 等任务时的网络拥塞问题. 为此，进行如下实验测试.

实验选择 Fat-Tree 拓扑结构 HDFS 集群进行，集群网络配置如表 4 所示. 以 512 个数据块的 WordCount 为计算用例，运行 MapReduce 任务. 数据块的配置依循可变 $\tilde{\kappa}$ 覆盖算法和 CS-3 重覆盖算法. Map 任务在数据块本地服务器执行，随后通过 shuffle 过程将 Map 结果通过数据中心网络传输到单独服务器执行 Reduce 计算.

表 4 HDFS 集群网络配置参数

参数	设定值
数据中心结构	Fat-Tree 结构，包含 45 个 6 口交换机，54 个 DataNodes 节点
交换机输出端口缓冲区大小	500 报文
流量生成器	CBR
链路带宽	接入层：200 Mbps 汇聚层和核心层：1000 Mbps
一个 Map 计算任务输出数据量	324.57 kB
全部 Map 计算任务输出数据量	270.049 MB
CBR 产生率	2 Mbps
报文大小	1000 Bytes

传输端到端延时是衡量网络服务质量的重要指标之一，图 16 统计了在不同存储策略下数据传递过程中最大延时和最小延时. 由实验结果可知：采用可变 $\tilde{\kappa}$ 覆盖集算法优化配置后集群的最大传输时延为 0.06128 s，对比算法的最大传输时延为 0.06152 s，减少 0.39%，最小传输时延为 0.06024 s，对比算法为 0.060416 s，减少 0.29%. 这是因为采用本文算法减少了集群中执行 Map 任务的 DataNode 数量，即数据流读取的源端数量，则对于多对一传输模式（Map 到 Reduce 传输过程）使用的交换机和通信链路减少，从而缓解了数据中心网络特有的 incast 拥塞问题，提升了传输及时性.

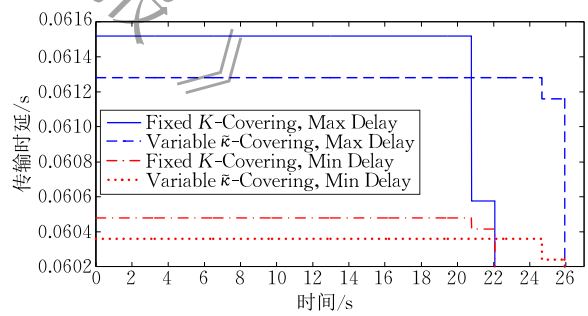

图 16 数据包的端到端传输时延比较

带宽占用是衡量网络运行性能的另一个重要指标，图 17 所示为采用可变 $\tilde{\kappa}$ 覆盖集算法和对比算法进行存储配置后，数据被读写操作的网络带宽占用率. 采用本文提出算法配置的 Reducer 端口带宽占用均值为 83156.61 Kbps，对比算法为 100703.56 Kbps，本算法节省 17.42%. 虽然都没有达到端口速率 200 Mbps，但该实验场景仅是轻载任务，随着任务的增多，对比算法的 Reducer 端口有可能由于传输数据量大于端口带宽而出现网络拥塞.

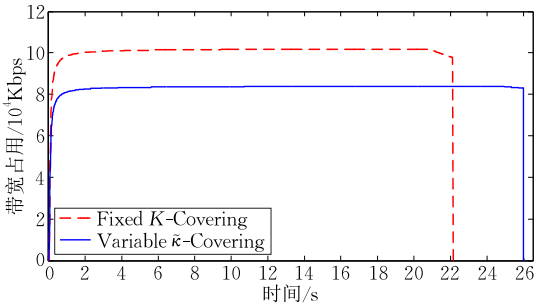

图 17　Reducer 端口带宽占用对比

最后，通过增加传输负荷测试在重负载情况下网络性能. 实验中同时执行三组本节 MapReduce 任务，每组实验在求解覆盖子集和进行中间数据传输时均是相互独立的，统计网络传输时的丢包率. 实验统计结果如表 5 所示：在实验重载情况下，采用 CS-3 重覆盖算法配置的集群在数据传输过程中出现了不同程度的丢包，丢包率在 1.9%左右，而采用本文算法能够有效避免网络传输时出现丢包，集群丢包率均为 0.

表 5　算法性能比较

		发包数量	丢包数量	丢包率%
恒定 3 覆盖算法	任务 1	270049	5158	1.91
	任务 2	270049	5160	1.91
	任务 3	270048	5116	1.89
可变 $\tilde{\kappa}$ 覆盖算法	任务 1	270049	0	0
	任务 2	270048	0	0
	任务 3	270049	0	0

6.6　可变 $\tilde{\kappa}$ 覆盖集求解算法最优性分析

本节通过实验讨论采用贪心萤火虫算法求解 HDFS 可变 $\tilde{\kappa}$ 覆盖集的最优性，并通过实际计算时间分析算法收敛性.

首先通过实验测试，验证采用贪心 FA 算法可找到 HDFS 可变 $\tilde{\kappa}$ 覆盖的最小子集. 对于解空间有界的 0-1 整数规划问题，枚举法总能够找到最优解. 因此我们通过实验对比本文提出的贪心萤火虫算法和枚举法的求解结果，以证明贪心 FA 算法求解 HDFS 可变 $\tilde{\kappa}$ 覆盖的最优性. 考虑实验场景多样性，选取需存储 512 种数据块的轻载荷和需存储 1024 种数据块的重载荷两个计算用例. 数据块可用性需求设定遵循：需保留 1 个活跃副本、2 个活跃副本与 3 个活跃副本的数据块数量之比为 4∶3∶1. 即轻载荷情况下，512 种数据块中 256 种数据块至少要有 1 个活跃副本，192 种数据块至少要有 2 个活跃副本，64 种数据块至少要有 3 个活跃副本；重载荷情况下 1024 种数据块中有 512 种数据块要保留 1 个活跃副本，384 种数据块要有 2 个活跃副本，128 种数据块要有 3 个活跃副本. 表 6 和表 7 给出了分别采用枚举法和贪心萤火虫算法求解可变 $\tilde{\kappa}$ 覆盖最小集的结果和算法执行时间.

表 6　轻载荷(512 种数据块)算法执行比对

	枚举法		贪心萤火虫算法	
	执行时间/s	覆盖集/台	执行时间/s	覆盖集/台
Fat-Tree	12440	42	0.3150	42
$BCube_2$	35862	48	0.9570	48
$DCell_2$	65073	69	1.8620	69

表 7　重载荷(1024 种数据块)算法执行比对

	枚举法		贪心萤火虫算法	
	执行时间/s	覆盖集/台	执行时间/s	覆盖集/台
Fat-Tree	26870	45	0.5780	45
$BCube_2$	79692	52	1.8860	52
$DCell_2$	124709	78	5.3440	78

由实验结果可知，基于贪心 FA 算法求解可变 $\tilde{\kappa}$ 覆盖最小集与枚举法求出的全局最优解相同，即贪心 FA 算法可寻优至全局最优解. 并且注意到贪心 FA 算法执行时间是枚举法的万分之一($1/10^5$ 级)，因此计算代价更低，更适用于大规模云计算数据中心应用.

7　结　论

突破 HDFS 机架感知存储策略的数据块副本个数恒定的局限，提出在保证数据块可用性前提下的灵活可变数据块副本存储策略，并将该问题数学表示为求解可变 $\tilde{\kappa}$ 覆盖最小集问题. 论文首先建立了数据中心 Hadoop 架构的数据存储超图模型，准确表述了文件-数据块和 DataNode 节点-机架间的复杂多对多关联关系，进而将求解可变 $\tilde{\kappa}$ 覆盖最小集问题映射为 $\tilde{\kappa}$-横贯超边计算，并给出一种贪心 FA 算法加以求解. 通过对不同规模的数据存储和实际运行 MapReduce 应用，论文所提出算法比现有恒定数据块副本配置算法能够更多的休眠 DataNode 节点，实现数据中心节能.

参　考　文　献

[1] Guo H, Wang L, Chen F, et al. Scientific big data and Digital Earth. Chinese Science Bulletin, 2014, 59(35): 5066-5073

[2] White T, Cutting D. Hadoop: The Definitive Guide. O'reilly Media Inc Gravenstein Highway North, 2012, 215(11): 1-4

[3] Feng Deng-Guo, Zhang Min, Li Hao, Big data security and privacy protection. Chinese Journal of Computers, 2014, 37(1): 246-258(in Chinese)

(冯登国，张敏，李昊. 大数据安全与隐私保护. 计算机学报，2014，37(1)：246-258)

[4] Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications, 2011, 34(1): 1-11

[5] Armbrust M, Fox A, Griffith R, et al. Above the clouds: A Berkeley view of cloud computing. EECS Department University of California Berkeley, 2009, 53(4): 50-58

[6] Koomey J G. Growth in Data Center Electricity Use 2005 to 2010. Oakland, USA: Analytics Press, 2011

[7] Gu Li-Jing, Zhou Fu-Qiu, Meng Hui, Research on energy consumption and energy efficiency of data center in China, Energy of China, 2010, 32(11): 42-45(in Chinese)
(谷立静，周伏秋，孟辉. 我国数据中心能耗及能效水平研究. 中国能源，2010，32(11)：42-45)

[8] Data Center Energy Efficiency Assessment Guide. Cloud Computing Development and Policy Forum Technical Report, 2012.3.16(in Chinese)
(数据中心能效测评指南."云计算发展与政策论坛"技术报告，2012.3.16)

[9] Elomari A, Maizate A, Hassouni L. Data storage in big data context: A survey//Proceedings of the IEEE International Conference on Systems of Collaboration (SysCo). Casablanca, Morocco, 2016: 1-4

[10] Song Bao-Yan, Wang Jun-Lu, Wang Yan. Optimized storage strategy research of HDFS based on Vandermonde code. Chinese Journal of Computers, 2015, 38(9): 1825-1837(in Chinese)
(宋宝燕，王俊陆，王妍. 基于范德蒙码的 HDFS 优化存储策略研究. 计算机学报，2015，38(9)：1825-1837)

[11] Di S, Kondo D, Cappello F. Characterizing cloud applications on a Google data center//Proceedings of the 42nd International Conference on Parallel Processing (ICPP). Lyon, France, 2013: 468-473

[12] Chen D, Chen Y, Brownlow B N, et al. Real-time or near real-time persisting daily healthcare data into HDFS and ElasticSearch index inside a big data platform. IEEE Transactions on Industrial Informatics, 2017, 13(2): 595-606

[13] Chen S, Pedram M. Efficient peak shaving in a data center by joint optimization of task assignment and energy storage management//Proceedings of the 10th IEEE International Conference on Cloud Computing. Honolulu, USA, 2017: 77-83

[14] Cheng Z, Luan Z, Meng Y, et al. ERMS: An elastic replication management system for HDFS//Proceedings of the 2012 IEEE International Conference on Cluster Computing Workshops (CLUSTER WORKSHOPS). Beijing, China, 2012: 32-40

[15] Abad C L, Lu Y, Campbell R H. DARE: Adaptive data replication for efficient cluster scheduling//Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER). Austin, USA, 2011: 159-168

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

[17] 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 2010 IEEE Second International Conference on Cloud Computing Technology and Science (IEEE CloudCom). Indiana University, USA, 2010: 274-287

[18] Maheshwari N, Nanduri R, Varma V. Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. Future Generation Computer Systems, 2012, 28(1): 119-127

[19] Liao B, Yu J, Zhang T, et al. Energy-efficient algorithms for distributed storage system based on block storage structure reconfiguration. Journal of Network & Computer Applications, 2015, 48(1): 71-86

[20] Harnik D, Naor D, Segall I. Low power mode in cloud storage systems//Proceedings of the IEEE International Symposium on Parallel & Distributed Processing. Chengdu, China, 2009: 1-8

[21] Kim J, Chou J, Rotem D. Energy proportionality and performance in data parallel computing clusters//Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM 2011). Portland, USA, 2011: 1762-1774

[22] Kim J, Chou J, Rotem D. iPACS: Power-aware covering sets for energy proportionality and performance in data parallel computing clusters. Journal of Parallel & Distributed Computing, 2014, 74(1): 1762-1774

[23] Yazd S A, Venkatesan S, Mittal N. Boosting energy efficiency with mirrored data block replication policy and energy scheduler. ACM SIGOPS Operating Systems Review, 2013, 47(2): 33-40

[24] Singh K, Kaur R. Hadoop: Addressing challenges of big data //Proceedings of the IEEE Advance Computing Conference. Gurgaon, India, 2014: 686-689

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

[26] Jelassi M N, Largeron C, Yahia S B. Concise representation of hypergraph minimal transversals: Approach and application on the dependency inference problem//Proceedings of the IEEE International Conference on Research Challenges in Information Science, Aitolia-Akarnania, Greece, 2015: 434-444

[27] Reiss C, Wilkes J, Hellerstein J L. Google cluster-usage traces: Format+schema. Google Inc., White Paper, 2011

[28] Fan X, Weber W D, Barroso L A. Power provisioning for a warehouse-sized computer. ACM SIGARCH Compute Architect News 2007; 35(2): 13-23

[29] Verma A, Ahuja P, Neogi A. pMapper: Power and migration cost aware application placement in virtualized systems// Proceedings of the Middleware 2008. New York, USA: Springer, 2008: 243-264

[30] Kusic D, Kandasamy N, Jiang G. Combined power and performance management of virtualized computing environments serving session-based workloads. IEEE Transactions on Network & Service Management, 2011, 8(3): 245-258

YANG Ting, born in 1979, Ph. D., professor. His research interests include high performance computing and cloud computing.

WANG Meng, born in 1994, M. S. candidate. Her research interests include cloud computing and data storage.

ZHANG Ya-Jian, born in 1991, Ph. D. candidate. His research interest is cloud computing.

ZHAO Ying-Jie, born in 1993, M. S. candidate. His research interest is cloud computing.

PEN Hai-Bo, born in 1989, Ph. D., postdoctor. His research interests include data center resource management.

Background

Data center is the key infrastructure for Cloud Computing to provide the huge computing and storage resources for various applications. Hadoop Distributed File System (HDFS), as one of most effective distributed storage systems, have been widely used for handling large amounts of data. It has excellent performance in terms of fault tolerance, reliability and scalability.

HDFS usually adopts the same replication and storage strategy to guarantee data availability, i. e. creating the same number of replicas for all data sets and randomly storing them across data nodes. Such strategies do not fully consider the difference requirements of data availability on different data sets. More servers than necessary should thus be used to store replicas of rarely-used data, which will lead to increased energy consumption. To address this issue, this paper studies the HDFS differential storage energy-saving optimal algorithm applying in Cloud Data center. Breaking through the limitation of the constant number of replicas in existing storage methods, this paper propose a variable number of active replicas storage strategy for each data block according to user requirements of data availability. Firstly, a novel hypergraph-based storage model for Cloud datacenters is developed, which can precisely represent the many-to-many relationship among files, data blocks, data racks, and data nodes. Based on the hypergraph-based storage model, a $\tilde{\kappa}$-transverse hyperedge algorithm is proposed to calculate the minimum set of data nodes variable $\tilde{\kappa}$ covering. Because of just running the minimum number of required data nodes, it can not only save energy for the datacenter, but also maintain full functionality.

The algorithm has also been implemented in a HDFS based prototype datacenter with WordCount, TeraSort, and Grep cloud computing cases for performance evaluation. Experimental results show that the variable hypergraph coverage based strategy can not only reduce energy consumption, but can also relieve the delivery congestion problem in data center network.

This research is supported by the National Natural Science Foundation of China No. 61571324, the Natural Science Foundation of Tianjin No. 16JCZDJC30900, and the National Program of International S&T Cooperation No. 2013DFA11040.