DOI:10.13471/j.cnki.acta.snus.2015.06.011

适应节能与异构环境的 MapReduce 数据布局策略*

廖 彬[1,2]，张 陶[3]，于 炯[2]，刘 继[1]，钟 磊[1]，刘 炎[4]
（1. 新疆财经大学统计与信息学院，新疆 乌鲁木齐 830012；
2. 新疆大学软件学院，新疆 乌鲁木齐 830008；
3. 新疆医科大学医学工程技术学院，新疆 乌鲁木齐 830011；
4. 清华大学软件学院，北京 100084）

摘 要：大数据处理过程中产生的高能耗问题亟待解决，尤其是在数据量规模剧增的背景下。在对已有数据布局策略存在问题分析的基础上，分析了与基于存储区域划分的节能模式及与异构 HDFS 集群的不适应、数据块切分算法不灵活、存储节点选择的随机性等几个方面的问题，继而提出面向节能的 MapReduce 数据布局策略。首先，新策略适应将集群划分为不同存储区域（Active-Zone 与 Sleep-Zone）的节能模式；其次，新策略对传统的数据块数计算方法进行了改进，提出作业截止时间约束下的最小任务数计算方法确定数据块数量；最后，新的存储策略增加了对异构集群环境的适应能力，并能根据不同的作业类型进行存储节点的选择。实验结果表明：新的数据布局策略能够适应异构集群环境，达到减小 MapReduce 作业能耗的目的。

关键词：绿色计算；MapReduce；异构环境；数据布局
中图分类号：TP393.09 **文献标志码**：A **文章编号**：0529－6579（2015）06－0055－12

An Energy-Efficient and Heterogeneous Environment Adaptive Data Layout Strategy for MapReduce

LIAO Bin[1,2], *ZHANG Tao*[3], *YU Jiong*[2], *LIU Ji*[1], *ZHONG Lei*[1], *LIU Yan*[4]
（1. College of Statistics and Information, Xinjiang University of Finance and Economics, Urumqi 830012, China;
2. School of Software, Xinjiang University, Urumqi 830008, China;
3. Department of Medical Engineering and Technology, Xinjiang Medical University, Urumqi 830011, China;
4. School of Software, Tsinghua University, Beijing 100084, China）

Abstract: The problem of high energy consumption producing from big data processing is an important issue that needs to be solved, especially under the background of data explosion. Based on analyzing problems of the existing data layout policy, the problems of the in adaptation of energy-saving mode based on storage area division and heterogeneous HDFS cluster, the inflexibility of data block segmentation algorithm, the randomness of storage node selection, proposing a data layout strategy orienting to energy conservation are analyzed. Firstly, the new strategy divides the cluster into two different storage areas to meet the needs of saving energy: Active-Zone and Sleep-Zone; secondly, the new strategy has made im-

* **收稿日期**：2014－12－07
基金项目：国家自然科学基金资助项目（61562078, 61262088, 71261025）；新疆财经大学博士启动基金资助项目（2015BS007）
作者简介：廖彬（1986 年生），男；**研究方向**：绿色计算、数据库系统理论及数据挖掘；E-mail：liaobin665@163.com

provements on traditional data block computing method, proposes a minimum number of jobs calculation method to determine the number of data blocks; at last, the new strategy can increase the adaptability of the heterogeneous cluster environment and can choose the appropriate storage nodes according to different job types. Experimental results show that the new data layout strategy can adapt to the heterogeneous cluster environment and reach the goal of reducing energy consumption for MapReduce jobs.

Key words: green computing; MapReduce; heterogeneous environment; data layout

随着云计算、物联网、移动互联网等技术的不断发展，数据正以前所未有的速度在不断增长和积累。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，同时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]。MapReduce 是最流行的大数据计算模式[2]，而 MapReduce 对大数据的分析与处理离不开底层的分布存储系统提供的数据存储服务，目前比较著名的分布存储系统有 Google 的 GFS（Google File System，谷歌分布式文件系统）[3]，HDFS（Hadoop Distributed File System，Hadoop 分布式文件系统）[4]、Lustre、MooseFs 以及清华大学自主研发的 CarrierFs 等。其中 GFS 管理着 Google 公司百万服务器上的海量数据，基于 GFS 的分布式数据库 BigTable 支撑着 Google 搜索、地图、社交网络等服务[5]。HDFS 作为 Hadoop 底层分布式文件系统，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[6]。

据文献［7］统计，2007 年全球数据量达到 281 EB，而 2007 年到 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系统规模的不断扩大，这使得运营成本不断的提高，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它开销。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。据文献［8］统计，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 8 680 亿度电，占全球总耗电量的 5.3%。纽约时报与麦肯锡经在《纽约时报》上发表了“Power, pollution and the Internet”[9]，调查显示全球的数据中心总用电量约为 3 000 亿 W，相当于 30 座核电厂的出电量，而巨大的能耗中却只有 6% ~ 12% 的能耗被用于相应用户的请求。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大数据快速发展的一个主要瓶颈[1]。

已有对 MapReduce 的数据布局研究工作的目的大多以提高 MapReduce 作业的执行效率为首要目标，并没有考虑到能耗因素。本文在对已有数据布局策略深入研究的基础上，总结了与基于存储区域划分的节能模式及与异构 HDFS 集群的不适应、数据块切分算法不灵活、存储节点选择的随机性等几个方面的问题；针对以上问题分别提出了对应的解决方案。

1 相关工作

从 2012 - 2014 年 Gartner 公布的技术发展趋势报告中可以看出，绿色 IT 技术已经成为十大 IT 关键技术之一。通过优化数据布局策略提升系统能耗效率属于节能的分布式存储系统研究方向之一。所以，本文首先将分布式存储系统的节能研究进行简要介绍，继而针对具体的数据布局相关工作进行阐述。

将分布存储系统中的能耗优化问题根据研究内容进行划分，可分为基于硬件的节能与基于调度的节能两个方面[10]。基于硬件的节能方法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。基于硬件的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。基于调度的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（如：休眠、降频等），以达到节能的目的。由于不需要对现有硬件体系进行改变，基于调度的节能方法是目前分布式存储节能技术的研究热点。当前，基于调度的节能研究主要集中在基于节点调度与数据调度两方面。节点调度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，并让其它节点进入低能耗模式以达到降

低能耗的目的。节点调度中被关闭节点的选择与数据调度技术紧密相关，而目前已有的数据调度技术主要有基于静态数据放置、动态数据放置与缓存预取三种。其中基于静态数据放置的数据调度根据固定的数据放置策略将数据存储到系统中各节点上后，将不再改变其存储结构[11-15]。基于动态数据放置的数据调度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[16-20]。基于缓存预取的数据调度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[21]。

研究 MapReduce 数据布局策略方面，已有工作主要有 CoHadoop、Hadoop + + 及 CHMJ[22-25]。CoHadoop解决了 Hadoop 无法把与作业相关的数据定位到同一个节点集合下的性能瓶颈，是对Hadoop的轻量级扩展，目的是使得应用层能控制数据的存储。CoHadoop 提高了索引（Indexing）、聚合（Grouping）、聚集（Aggregation）、纵向存储（Columnar Storage）、合并（join）以及 Sessionization 等操作的效率。Hadoop + + 改进思路与 CoHadoop 类似，它将同一个作业产生的两个数据文件放置在一起；但是当有新文件写入系统的时，系统需要对数据进行重新组织。CHMJ 设计了多副本一致性哈希算法，将具有连接关系的表根据其连接属性的哈希值在机群中进行分布，在提升连接查询处理中数据本地性的同时，也保证了数据的可用性。并且，CHMJ 在多副本一致性哈希数据分布的基础上，提出了 HashMapJoin 并行连接查询处理算法，有效地提高了连接查询的处理效率。以上对 MapReduce 的数据布局研究工作的目的都是以提高 MapReduce 作业的执行效率为首要目标，并没有考虑到能耗因素，这是本文与已有工作的最大不同之处，针对已有布局策略有可能导致的能耗问题，本文在第 3 章进行了详细阐述。本文研究节能的 MapReduce 数据布局策略，主要做了如下几个方面的工作。

1）归纳分析了已有的数据布局策略存在：不适应基于存储区域划分的节能模式及异构的 HDFS 集群环境，数据块切分算法缺乏灵活性，存储节点选择的存在随机性等问题。

2）通过返回节点状态矩阵中处于活动状态的节点，对通过将 RACK 划分为 Active-Zone 与 Sleep-Zone 存储区域的节能模式进行了适应。

3）当作业具有截止时间约束时，通过对 MapReduce计算原理及公式的推导，得到了任务截止时间约束下的最小任务计算方法，从而可以有效的确定数据布局前的数据块的切分数量（即 Map 任务的数量）。

4）已有方法采用平均主义及随机方法确定数据块的大小及存储位置，不能很好的适应异构集群及不同 MapReduce 作业的特点。异构集群环境下，本文对各异构节点执行不同 MapReduce 作业类型的计算能力进行评估，提出了适应异构集群的数据块大小切分方法及适应不同作业类型的存储节点选择方法。

2　已有数据布局策略存在的问题

MapReduce 任务执行前首先要将任务处理的数据存储到 HDFS 中，HDFS 提供了分布、高效的数据存储及访问能力。数据文件被切分成固定大小的数据块（Data Block）分布的存储到 DataNode 节点中，其中数据块的大小是可配置的（默认大小为 64 MB）。HDFS 中数据文件只能一次写入，并不允许对已经写入的数据块进行修改。NameNode 节点内存中维护着文件系统的目录信息，即维护着活动的 DataNode 节点列表及每个 DataNode 节点中所有的数据块信息（BlockMap）。HDFS 通过副本机制达到容错及故障恢复的目的，系统默认副本系数为 3，并采用基于机架感知的数据块存储策略（如图 1 所示）将数据块布局到 HDFS 系统中[26]。

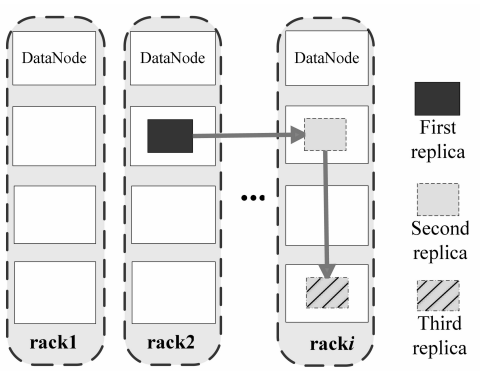

图 1　基于机架感知的数据块存储策略

Fig. 1　The rack-aware block replica placement policy in HDFS

机架感知的数据块存储策略首先将数据块的第 1 个副本存放在创建该数据块的本地 DataNode 上（前提条件是本地 DataNode 节点空间足够），此策略叫就近写（Write Affinity）；第 2 个副本会被存储到同第 1 个副本不同机架（Rack）的 DataNode 上；第 3 个副本会被存储到同第 2 个副本同机架但不同

DataNode 的节点中。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位置具有较大的随机性[27]，这种随机的布局策略虽然能够均衡系统的负载，但是同样存在以下几个方面的问题。

2.1 与基于存储区域划分的节能模式的不适应

文献［27］中将存储区域划分为 Active-Zone 与 Sleep-Zone，并适时的将 Sleep-Zone 中的节点进行休眠处理以达到节能的目的，基于机架感知的数据块存储策略在进行数据布局时并没有对节点的区域（Active-Zone 与 Sleep-Zone）进行区分对待，致使不能很好的适应节能模式的需求。

2.2 与异构 HDFS 集群的不适应

现有的数据放置策略并没有考虑到系统中数据存储节点的异构性问题，基于机架感知的数据块存储策略认为 HDFS 集群中所有节点的服务能力相同，数据按照平均主义的原则进行切分、布局。在这种布局策略下，服务能力强的节点能够较早的完成任务，而服务能力较弱的节点则需要较长的时间完成任务，将引发以下两种情况：第一，当快任务能够容忍慢任务时，将造成“任务等待”，即早完成的任务等待还未完成的任务。第二，当块任务不能容忍慢任务时，将引发 Hadoop 的推测执行（Speculative Execution）机制，推测执行机制是为了防止运行速度慢的任务影响作业的整体执行速度，根据推测算法推测出“拖后腿”的任务，并为该任务启动一个备份任务，并最终选用最先成功运行完成任务的计算结果作为最终结果。由此可见，不论是那种情况，都会造成资源的浪费。

2.3 数据块切分算法不灵活

Hadoop 中利用类 InputFormats 来定义文件是如何被切分并由 Map 任务使用，InputFormats 为用户提供了一种扩展的手段，用来通过组装定制化的分片来将一些分布的数据提供给 Map 任务。当任意 MapReduce 任务启动的时，数据文件会被 InputFormats 切分为多个分片（splits），每一个分片对应了 MapReduce 程序中的一个 Map 任务。默认情况下，不同的 FileInputFormat 实现将一个文件分成 64 MB 的 chunk（HDFS 的默认块大小）。Hadoop 调度器尽量将 Map 任务调度给那些包含分片的本地副本的 Datanode。大多数情况下，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。

已有的数据块大小只能由固定的配置参数进行设置，并不能很好的适应不同 MapReduce 任务的特点，更不能根据作业的完成时间需求及集群中各节点的计算能力的不同而进行灵活的设置。

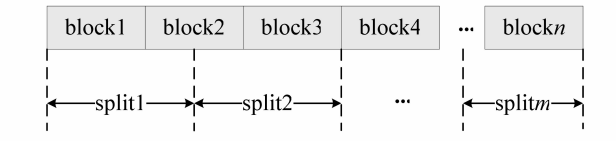

图 2 数据块 block 与 split 之间的对应关系

Fig. 2 The relationship between data block and split

2.4 存储节点选择的随机性

基于机架感知的数据块存储策略在选择存储节点时具有较大的随机性，不能很好的进行作业类型与节点类型之间的匹配。一方面，不同 MapReduce 作业对 CPU、内存、磁盘、网络等资源的需求不同，而异构 HDFS 集群环境下不同节点之间的 CPU、内存、磁盘、网络等资源的服务能力不尽相同；另一方面，MapReduce 作业调度时的本地优先策略容易导致部分任务在不合适的节点上执行（由存储节点选择的随机性造成），从而影响作业的整体进程。由此可见，必须为不同的 MapReduce 作业类型所处理的数据选择合适的存储节点，是适应节能的数据布局算法必须考虑的问题。

针对以上问题，本文第 3 节提出面向节能的 MapReduce 数据布局算法对问题予以解决。

3 面向节能的 MapReduce 数据布局策略

3.1 面向节能的数据布局策略总体流程

如图 3 所示为面向节能的 MapReduce 数据布局策略流程图。

首先，判断是否处在节能模式；如果处于节能模式，则接着判断存储区域的划分状态；如果不处于节能模式，则采用基于机架感知的数据布局策略进行数据的存储。当处于节能存储区域划分模式时，返回 Active-Zone 区域中存储节点（参见 3.2 节）；否则，返回集群中所有节点。当作业具有截止时间约束时，通过 3.3 节中提出的任务截止时间约束下的最小任务计算方法确定数据块的切分数量；否则，按照系统配置参数确定数据块数。当系统为异构集群时，通过 3.4 节中提出的适应异构集群的数据大小切分方法确定具体数据块的大小；如果系统为同构集群，数据块的大小由系统参数配置确定。最后，数据块的存储节点根据 3.5 节适应不同作业类型的节点选择算法进行确定。

3.2 支持节能存储区域划分的数据布局

我们的早期工作[27]将 RACK 划分为 Active-

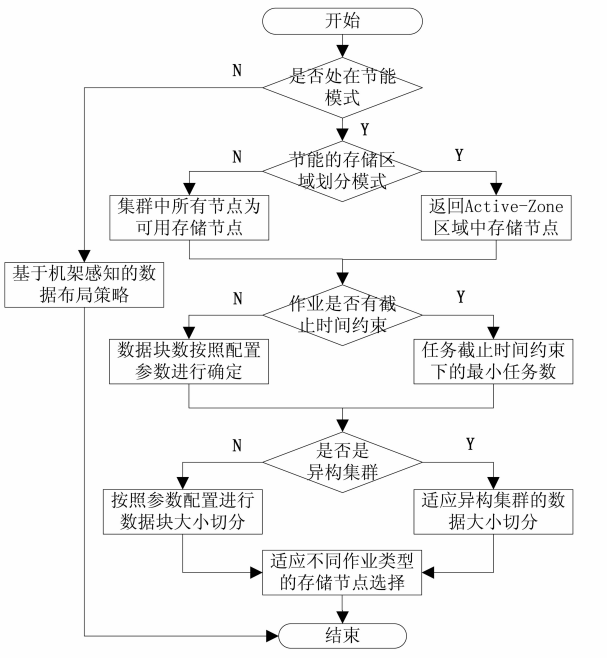

图 3　面向节能的 MapReduce 数据布局策略流程图

Fig. 3　The flow chart for energy saving MapReduce data layout strategy

Zone 与 Sleep-Zone 两个存储区域，根据不同数据的访问频率与规律计算活动因子以配置数据的存储区域，通过数据中心负载规律适时对 Sleep-Zone 区域中的服务器进行休眠处理以达到节能的目的。基于机架感知的数据块存储策略在进行数据布局时并没有对节点的区域（Active-Zone 与 Sleep-Zone）进行区分对待，致使不能很好的适应节能模式的需求。在通过划分 Active-Zone 与 Sleep-Zone 存储区域节能模式下，不是所有的节点都处于可用状态，必须返回处于活动状态的节点，即返回节点状态矩阵[27]（式 1）中 $S_{mn} = 1(1 \leqslant m \leqslant sm, 1 \leqslant n \leqslant i)$ 的节点[27]

$$DS_{sm\times i} = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1i} \\ S_{21} & S_{22} & \cdots & S_{2i} \\ & & & \\ S_{sm1} & S_{sm2} & \cdots & S_{smi} \end{bmatrix} \tag{1}$$

3.3　作业截止时间约束下的最小任务数

通过 4.2 节任务数对任务完成时间及能耗的影响实验我们发现：首先，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；其次，随着任务数量的不断增大，作业能耗不断的增加；即完成相同作业，任务数越大，完成该任务能耗越大。由于 MapReduce 作业大多受到任务完成时间的约束，所以本节主要对满足作业截止时间约束下的 MapReduce 作业最小分解任务数进行建模，试图通过减小任务数量达到节能的目的。

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，能够通过表 1 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 与 N_R^S 个 Map 与 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 $T_M(i)$，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

$$T_M(i) = f(C, D_M(i)) \tag{2}$$

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

$$T_S(j) = g(C, D_S(j)) \tag{3}$$

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

$$T_R(j) = h(C, D_R(j)) \tag{4}$$

其中，式（2）－（4）中的 C 表示任意节点 dn_{ij} 处理作业 J 时的计算能力（其值可通过训练获得）。

5）计算 Map 阶段的采样任务平均完成时间为

$$T_M^\Lambda = \frac{\sum_{i=1}^{N_M^S} f(C, D_M(i))}{N_M^S} \tag{5}$$

6）计算 Shuffle&Sort 阶段的采样任务平均完成时间为

$$T_S^\Lambda = \frac{\sum_{j=1}^{N_M^R} g(C, D_R(j))}{N_M^R} \tag{6}$$

7）计算 Reduce 阶段的采样任务平均完成时间为

$$T_{\mathrm{R}}^{\mathrm{A}} = \frac{\sum_{j=1}^{N_{\mathrm{M}}^{\mathrm{R}}} h(C, D_{\mathrm{R}}(j))}{N_{\mathrm{M}}^{\mathrm{R}}} \tag{7}$$

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

$$T_{\mathrm{M}}^{\mathrm{avg}} \approx \left[\frac{N_{\mathrm{M}}}{R_{\mathrm{M}}}\right] T_{\mathrm{M}}^{\mathrm{A}} \approx \left[\frac{N_{\mathrm{M}}}{R_{\mathrm{M}}}\right] \times \frac{\sum_{i=1}^{N_{\mathrm{M}}^{\mathrm{S}}} f(C, D_{\mathrm{M}}(i))}{N_{\mathrm{M}}^{\mathrm{S}}} \tag{8}$$

同样方法，可预测 Reduce 阶段（包含 Shuffle&Sort 阶段）任务的平均完成时间为

$$T_{\mathrm{R}}^{\mathrm{avg}} \approx \left[\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}} - 1\right] T_{\mathrm{S}}^{\mathrm{A}} + \left[\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}\right] T_{\mathrm{R}}^{\mathrm{A}} \approx \left[\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}} - 1\right] \times \frac{\sum_{j=1}^{N_{\mathrm{M}}^{\mathrm{R}}} g(C, D_{\mathrm{R}}(j))}{N_{\mathrm{M}}^{\mathrm{R}}} + \left[\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}\right] \times \frac{\sum_{j=1}^{N_{\mathrm{M}}^{\mathrm{R}}} h(C, D_{\mathrm{R}}(j))}{N_{\mathrm{M}}^{\mathrm{R}}} \tag{9}$$

值得注意的是，由于作业 J 在集群中运行 Shuffle&Sort 的第一次操作与 Map 阶段有重叠部分，所以式（8）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（8）与式（9）的基础上，总完成时间可用式（10）进行计算

$$T_{\mathrm{avg}} \approx T_{\mathrm{M}}^{\mathrm{avg}} + T_{\mathrm{R}}^{\mathrm{avg}} \approx \left[\frac{N_{\mathrm{M}}}{R_{\mathrm{M}}}\right] T_{\mathrm{M}}^{\mathrm{A}} + \left[\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}} - 1\right] T_{\mathrm{S}}^{\mathrm{A}} + \left[\frac{N_{\mathrm{R}}}{R_{\mathrm{R}}}\right] T_{\mathrm{R}}^{\mathrm{A}} \tag{10}$$

对式（10）进行变换，可得到式（11）

$$T_{\mathrm{avg}} + T_{\mathrm{S}}^{\mathrm{A}} \approx \frac{N_{\mathrm{M}} \times T_{\mathrm{M}}^{\mathrm{A}}}{R_{\mathrm{M}}} + \frac{N_{\mathrm{R}} \times (T_{\mathrm{S}}^{\mathrm{A}} + T_{\mathrm{R}}^{\mathrm{A}})}{R_{\mathrm{R}}} \tag{11}$$

在式（11）中，设 $A_{\mathrm{J}} = N_{\mathrm{M}} \times T_{\mathrm{M}}^{\mathrm{A}}$，$B_{\mathrm{J}} = N_{\mathrm{R}} \times (T_{\mathrm{S}}^{\mathrm{A}} + T_{\mathrm{R}}^{\mathrm{A}})$，$C_{\mathrm{J}} = T_{\mathrm{avg}} + T_{\mathrm{S}}^{\mathrm{A}}$，可将式（11）简化表示为

$$C_{\mathrm{J}} = \frac{A_{\mathrm{J}}}{R_{\mathrm{M}}} + \frac{B_{\mathrm{J}}}{R_{\mathrm{R}}} \tag{12}$$

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

$$\begin{cases} f(R_{\mathrm{M}}, R_{\mathrm{R}}) = R_{\mathrm{M}} + R_{\mathrm{R}}, \\ \text{s.t. } C_{\mathrm{J}} = \dfrac{A_{\mathrm{J}}}{R_{\mathrm{M}}} + \dfrac{B_{\mathrm{J}}}{R_{\mathrm{R}}}. \end{cases} \tag{13}$$

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

$$\begin{cases} R_{\mathrm{M}} = \dfrac{\sqrt{A_{\mathrm{J}}}(\sqrt{A_{\mathrm{J}}} + \sqrt{B_{\mathrm{J}}})}{C_{\mathrm{J}}}, \\ R_{\mathrm{R}} = \dfrac{\sqrt{B_{\mathrm{J}}}(\sqrt{A_{\mathrm{J}}} + \sqrt{B_{\mathrm{J}}})}{C_{\mathrm{J}}}. \end{cases} \tag{14}$$

分别将 $A_{\mathrm{J}} = N_{\mathrm{M}} \times T_{\mathrm{M}}^{\mathrm{A}}$，$B_{\mathrm{J}} = N_{\mathrm{R}} \times (T_{\mathrm{S}}^{\mathrm{A}} + T_{\mathrm{R}}^{\mathrm{A}})$，$C_{\mathrm{J}} = T_{\mathrm{avg}} + T_{\mathrm{S}}^{\mathrm{A}}$，代入式（14）可得

$$\begin{cases} R_{\mathrm{M}} = \dfrac{\sqrt{N_{\mathrm{M}} \times T_{\mathrm{M}}^{\mathrm{A}}}(\sqrt{N_{\mathrm{M}} \times T_{\mathrm{M}}^{\mathrm{A}}} + \sqrt{N_{\mathrm{R}} \times (T_{\mathrm{S}}^{\mathrm{A}} + T_{\mathrm{R}}^{\mathrm{A}})})}{T_{\mathrm{avg}} + T_{\mathrm{S}}^{\mathrm{A}}}, \\ R_{\mathrm{R}} = \dfrac{\sqrt{N_{\mathrm{R}} \times (T_{\mathrm{S}}^{\mathrm{A}} + T_{\mathrm{R}}^{\mathrm{A}})}(\sqrt{N_{\mathrm{M}} \times T_{\mathrm{M}}^{\mathrm{A}}} + \sqrt{N_{\mathrm{R}} \times (T_{\mathrm{S}}^{\mathrm{A}} + T_{\mathrm{R}}^{\mathrm{A}})})}{T_{\mathrm{avg}} + T_{\mathrm{S}}^{\mathrm{A}}} \end{cases} \tag{15}$$

由于 MapReduce 作业的类型是有限的，相同类型的作业 Map、Shuffle&Sort 与 Reduce 的数据量与完成时间的函数（式（5）-（7））相同。所以对于相同类型的 MapReduce 作业，函数式（5）-（7）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 3.4 节的方法可确定任意节点 dn_i 的处理作业 J 时的计算能力。所以，式（15）可解。同样，当参数 R_{M} 与 R_{R} 值已知时，可利用式（15）确定参数 N_{M} 与 N_{R} 的值，即当可用 Map 资源 slot 及 Reduce 资源 slot 数已知的情况下，可对满足作业完成时间 deadline 约束条件下的最小 Map 任务数及 Reduce 任务数进行计算。而本文中则主要取 Map 任务数的值，因为 Map 任务数决定了数据布局过程中数据块的数量。

3.4 适应异构集群的数据大小切分

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之间的差异，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的解决异构集群中各个节点处理能力的差异性问题，保证同一个作业的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

$$C_{ij} = \frac{\mathrm{DataVol}_{ij}}{\mathrm{Time}_{ij}} \tag{16}$$

其中 Time_{ij} 表示节点 dn_i 完成任务 task_j 所花的时间，$\mathrm{DataVol}_{ij}$ 表示任务所处理数据量的大小。不论是异构还是同构 Hadoop 集群，集群每个节点对于不同作业类型的计算能力可通过大量的训练得到，本文中设 C_{ij} 值为已知。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。

表 1　节点计算能力记录表

Table 1　DataNode's computing capability for each task

节点	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 被分解为 n 个 Map 任务（或由 3.3 节中任务截止时间约束下的最小切分方法确定），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，即适应异构的集群环境，数据的切分规则应满足（16）式

$$\begin{cases} \dfrac{C_1}{D_1} = \dfrac{C_2}{D_2} = \ldots = \dfrac{C_n}{D_n}, \\ D = D_1 + D_2 + \ldots + D_n. \end{cases} \tag{17}$$

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

3.5　适应不同作业类型的存储节点选择

不同 MapReduce 作业对 CPU、内存、磁盘、网络等资源的需求不同，而异构 HDFS 集群环境下不同节点之间的 CPU、内存、磁盘、网络等资源的服务能力不尽相同。解决已有数据布局策略的随机性，须为不同的 MapReduce 作业类型所处理的数据选择合适的存储节点（任务执行节点）。

某 MapReduce 任务数据块存储点选择由式（18）与式（19）计算结果共同决定，其中式（19）为 MapReduce 任务对节点的评价函数，式（19）为节点选择函数。

$$\text{Capacity}_{dn_i} = (1-\alpha) \cdot \sum_{j=1} \theta_j \cdot r_j \tag{18}$$

其中 dn_i 表示集群中某节点，Capacity_{dn_i} 表示节点 dn_i 对 MapReduce 作业 J 的服务能力；r 表示节点各资源（如 CPU、内存、磁盘 I/O、网络等）的服务能力。θ 表示 MapReduce 作业 J 对于资源的依赖系数，且 $\sum_{j=1} \theta_j = 1$。$\alpha \in [0,1]$ 表示节点的繁忙程度，节点越繁忙，α 值越大。

由式（19）选择服务评价值最高的节点为数据存储节点。

$$\text{Max}(\text{Capacity}_{dn_1}, \text{Capacity}_{dn_2}, \ldots, \text{Capacity}_{dn_y}) \tag{19}$$

以上方法需要进行大量的计算且参数难以获得，也可通过 3.4 节中提出的方法，通过大量的训练得到表 1 所示的数据记录形式，对与不同的 MapReduce作业可通过查询已有数据，并通过式（19）决定存储节点。

4　实验及结果分析

4.1　实验环境配置

为了对本文提出的 MapReduce 数据布局策略进行实验分析，项目组搭建了拥有 22 个节点的 Hadoop 集群；其中 NameNode 与 SecondNameNode 分别独立为一个节点，其余 20 节点为 DataNode（5RACK ×4 DataNode）。实验环境拓扑结构如图 4 所示。

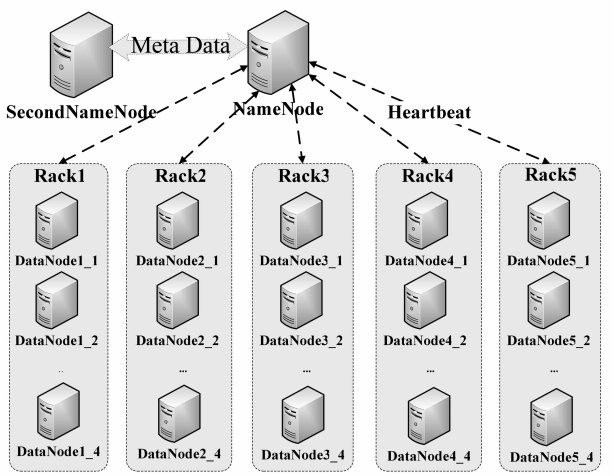

图 4　实验环境拓扑结构图

Fig. 4　Topology diagram of experimental environment

管理节点 2 个，DataNode 节点 20 个

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，特将数据块默认的分块大小配置为 128 MB，即 dfs. block. size = 128 MB。单个 DataNode 节点上 Map 与 Reduce 任务 Slot 资源槽数设置为 1，即配置项为

mapred. tasktracker. map. tasks. maximum = 1

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 1s/次，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。实验总体环境描述见表 2。

表 2 总体实验环境描述

Table 2 Description of experimental environment

项目	描述
操作系统	Debian 7.0
Java 版本	1.6 for Linux
Hadoop	1.0.4
能耗数据测量	北电电力监测仪(USB 智能版),标准为 GB/T17215 -2003,功率误差值(±0.01 ~ 0.1)W,采样频率为 1.5 ~3 s 之间,单位为 kWh
能耗数据采集	电力监测仪用电监测管理系统 V1.0.1
能耗相关单位	功率:W,能耗:J
数据采样频率	1 s 采集数据 1 次
节点 CUP	Intel core2 duo E 8400 3.00 GHz
节点内存	2GB- DDR2-800 MHz
节点硬盘	Hitachi HDP 725032 GLA 380(320 G,7 200 转/s)
网卡信息	Realtek RTL 8168/8111 PCI-E Gigabit Ethernet NIC- 100 Mbps

4.2 任务数对任务完成时间及能耗的影响

本实验采用 Terasort 作业，运行过程中对 Map 与 Reduce 任务进行不同的设置，记录不同条件下作业的执行时间及能耗。其中按数据量的不同分为 3 组：第一组，数据量为 1 907.4 MB 并设置 Map 数量为 40，每个 split 大小为 47.685 MB；第二组，数据量为 2 861 MB 并设置 Map 数量为 60，每个 split 大小为 47.683 MB；第三组，数据量为 4 768.4 MB并设置 Map 数量为 80，每个 split 大小为 59.605 MB。MapReduce 任务分解数与作业完成时间之间的关系如表 3 所示。

表 3 任务数与作业完成时间之间的关系

Table 3 The relationship between job completion time with different task number s

组别	1	2	4	6	8	10	12	14
第一组	410	235	193	188	154	151	122	120
第二组	613	354.5	249	246	229.5	213	222	216
第三组	1 031.3	536	388	348	326	307	255	256
组别	16	18	20	22	24	26	28	30
第一组	106	118	94	92	86	91	82	88
第二组	228	153	159.5	153	147	152	142.5	146
第三组	248	252	239.3	235	226.7	231	218	221

从图 5 可以看出：三组实验作业完成时间都随着任务数的增大而减小，说明可以通过适当的增大任务数量来满足作业的 deadline 需求。另一方面，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（减小效果不明显）。

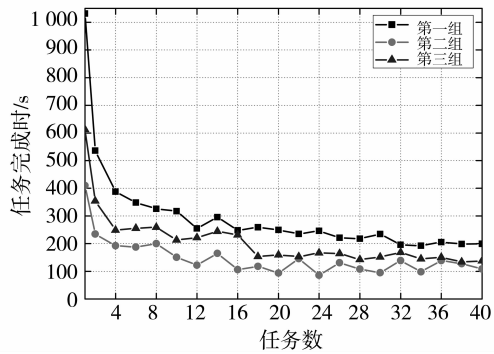

图 5 任务数与作业完成时间之间的关系

Fig. 5 The relationship between task number and job complete time

由图 6 趋势与表 4 中的数据可以看出，随着任务数量的不断增大，作业能耗不断的增加；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成本增加；任务之间的关联变得复杂，任务之间的协调成本增加，更多的任务启动操作及任务完成后的清理动作，以上几方面都导致了能耗的增加。图 6 表明：作业被分解的任务数越少，完成相同作业所需能耗越小，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 3.4 节提出的任务截止时间约束下的最小任务切分方法能够计算出能耗最优的作业分解方案，即能够在满足作业完成时间约束前提条件下，最小化作业执行能耗。

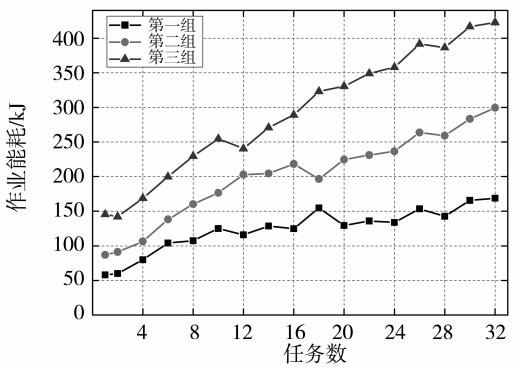

图 6 任务数与作业能耗之间的关系

Fig. 6 The relationship between task number and energy consumption

4.3　综合对比实验

为构造出异构集群环境，本实验将一组 RACK 中的4 台机器替换为性能较差的节点（配置为单核 Intel Pentium 4 1.5GHz，512 MB 内存及 40 GB 硬盘），即本实验中集群环境由两种不同配置的节点组成（设配置高节点类型为 A，配置低节点类型为 B）。将表 5 中配置的 MapReduce 作业运行 50 次，记录各作业的运行结果，整理出不同节点类型对于不同作业的计算能力，即 3.4 节中节点计算能力记录表（表 1）。

表 4　任务数与作业能耗之间的关系

Table 4　The relationship between energy consumption with different task number　　kJ

组别	1	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
第一组	58.0	60.0	80.0	104.0	107.5	125.0	116.0	128.6	124.7	154.8	129.3	135.9	133.9	153.6	142.5	165.7
第二组	87.0	91.0	106.5	138.3	160.2	176.6	202.8	204.6	218.3	196.4	224.6	231.2	236.6	263.8	259.0	283.3
第三组	145.5	142.0	168.7	199.8	229.6	254.6	240.5	270.6	289.1	322.9	330.4	348.9	358.0	391.6	386.5	416.7

表 5　作业类型说明

Table 5　Description of MapReduce Jobs

名称	参数配置
WordCount	数据总量为 9 759.6 MB，Map 任务数为 10，Reduce 任务数为 10
TeraSort	数据总量为 9 536.7 MB，Map 任务数为 10，Reduce 任务数为 10
NutchIndex	Page 数量为 1 000 000，Map 任务数为 80，Reduce 任务数为 10
K – means	Cluster 数为 5，Sample 数为 40 000 000，每个 Input 文件中的 Sample 数为 4 000 000，dimensions 大小为 20，最大迭代次数为 1，Map 任务数为 10，Reduce 任务数为 1
Bayes	Page 数目为 50 000，分类数目为 100，参数 ngrams = 3，Map 任务数为 10，Reduce 任务数为 1
PageRank	Page 数目为 3 000 000，迭代次数设置为 3，参数 Block 与 Block_width 分别设置为 0 与 16，Map 任务数为 10，Reduce 任务数为 1

实验结果如表 6 所示。

数据布局阶段，首先按照机架感知的数据块布局策略将 6 种不同的作业数据进行分布存储，数据块大小相同，最后记录各作业的运行时间及能耗；再次，按照本文中提出的作业截止时间约束下的最小任务数确定 Map 与 Reduce 的任务数量，再适应异构环境下的数据切分（3.4 节）及存储节点选择方法（3.5 节）进行数据的布局，记录各作业的运行时间及能耗。两种布局策略下 6 种作业（作业配置参数如表 5 所示）的能耗及完成时间（实验 20 次平均值）对比如表 7 所示。

如表 7 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，本文的数据布局策略能够分别提高 WordCount、TeraSort、NuthIndex、K-means、PageRank 及 Bayes 作业完成时间 24.5、23.6、38.4、25.7、47.4 及 71.9 s，提升作业完成时间 3.55%、5.00%、6.22%、7.59%、10.14% 和 17.17%；分别节能 33.31、34.41、42.29、33.47、59.7 和 76.14 kJ，节能效率分别为 3.650%、5.185%、5.186%、7.723%、10.423% 和 15.100%。通过分析可以发现本文提出的数据布局策略对于 WordCount、TeraSort、NuthIndex 及 K-means 四种作业完成时间及节能效率提升有限（5% 左右），而 PageRank 及 Bayes 有较为明显的提升（10% 以上）。分析 PageRank 与 Bayes 作业的执行过程，发现 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；这是由于 PageRank 与 Bayes 作业的 Map 与 Reduce 的任务存在强的先后依赖关系，Reduce 任务必须等待所有的 Map 任务完成后才能开始。而与 PageRank 与 Bayes 作业不同的是，WordCount、TeraSort、NuthIndex 及 K-means 四种作业完成部分 Map 任务后 Reduce 任务就能够开始执行，所以等待时延较短。

表 6 作业 Map 与 Reduce 任务的平均功耗及计算能力

Table 6 Map and Reduce task's average power consumption and computing capability

作业名称	阶段性任务	A 节点		B 节点	
		Map	Reduce	Map	Reduce
WordCount[1)]	N/A	1.31	82.4	0.248	14.65
TeraSort[1)]	N/A	5.98	5.49	1.085	1.02
NutchIndex[2)]	N/A	436.9	253.5	78.56	42.43
K-means[3)]	Cluster Iterator	5 787.4	7 138	1 045.5	1 283
	Cluster Classification	7 645.0	N/A	1 378.0	N/A
PageRank[2)]	Stage1	3 334	1 687.5	605	305.82
	Stage2	3 335	7 694.8	598.6	1 384.6
Bayes[2)]	DocumentProcessor：DocumentTokeuizer	683.0	1 131	122.7	201.5
	CollocDriver. generateCollocations	12.88	156.8	2.3	28.6
	CollocDriver. computeNGrams	438.6	834.5	78.3	150.2
	DictionaryVectorizer：MakeParitialVectors	1 265.0	1 121	225.0	200.4
	ParticalVectorMerger：MergePartialVectors	4 764.0	2 446	857.2	439.0
	VectorTfld Document Frequency Count	4 967.8	2 473	894.0	446.0
	MakePratialVectors	4 865.5	2 476	875.2	445.6
	ParticalVectorMerger：MergePartialVectors	4 886.8	2 468	877.6	443.8
	TrainNaiveBayesJob IndexInstancesMapper-Reducer	4 951.5	2 018	891.5	362.7
	TrainNaiveBayesJob WeightsMapper-Reducer	4 868.8	2 584	874.5	461.5

1）单位为 MB/s；2）单位为 page/s；3）单位为 sample/s

表 7 不同数据布局策略下作业完成时间及能耗对比

Table 7 The comparison of job completion time and energy consumption with different data layout strategy

作业名称	时间/s			能耗/kJ		
	同构环境	异构环境[1)]	异构环境[2)]	同构环境	异构环境[1)]	异构环境[2)]
WordCount	578.50	689.25	664.78	844.848	912.436	879.124
TeraSort	391.95	471.75	448.16	614.537	663.700	629.286
NutchIndex	510.90	618.00	579.56	755.187	815.600	773.310
K-means	282.75	338.25	312.57	401.295	433.400	399.927
PageRank	352.95	467.25	419.87	530.332	572.760	513.060
Bayes	314.60	418.50	346.65	466.773	504.120	427.977

1）机架感知策略；2）本文策略

5 结论及下一步工作

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大数据快速发展的一个主要瓶颈，所以研究节能的 MapReduece 数据布局策略具有重要意义。已有对 MapReduce 的数据布局研究工作的目的大多以提高 MapReduce 作业的执行效率为首要目标，并没有考虑到能耗因素。本文在对已有数据布局策略深入研究的基础上，总结了与基于存储区域划分的节能模式及与异构 HDFS 集群的不适应、数据块切分算法不灵活、存储节点选择的随机性等几个方面的问题；针对以上问题分别提出了对应的解决方案。首先，新策略适应将集群划分为 Active-Zone 与 Sleep-Zone 存储区域的节能模式；其次，新策略对传统的数据块数计算方法进行了改进，提出作业截止时间约束下的最小任务数计算方法确定数据块数量；最后，新的存储策略增加了对异构集群环境的适应能力，并能根据不同的作业类型进行存储节点的选择。为了对新的数据布局算法的有效性进行验证，本文搭建了

异构集群环境，并对不同作业的完成时间及能耗进行了测量，通过实验数据表明：新的数据布局策略能够适应异构集群环境，达到减小 MapReduce 作业能耗的目的（但不同作业之间存在着差异）。

下一步主要工作是研究 MapReduce 作业能耗模型。MapReduce 能耗模型是将来开发 MapReduce 作业能耗监控及优化软件的理论基础及关键技术，因为能耗模型能够实现在作业执行前对能耗进行预测，执行过程中为节能调度系统提供调度依据，执行后对作业进行能耗计算。

参考文献：

[1] 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] DEAN J, GHEMAWAT S. MapReduce: Simplifed data processing on large clusters [C] // Proceedings of the Conference on Operating System Design and Implementation (OSDI), New York: ACM, 2004: 137 – 150.

[3] GHEMAWAT S, GOBIOFF H, LEUNG S T. The google gile system [C] // Proceedings of 19th ACM Symposium on Operating System Principles, New York:ACM, 2003: 29 – 43.

[4] BORTHAKU D. The hadoop distributed file system: Architecture and design [EB/OL]. (2007 – 07 – 01) [2011 – 2 – 12], http: // hadoop. apache. org/common/docs/r0. 18. 2/hdfs_design. pdf.

[5] CHANG F, DEAN J, GHEMAWAT S, et al. Bigtable: A Distributed Storage System for Structured Data [C] // Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI), Seattle, WA, USA, 2006: 205 – 218.

[6] 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.

[7] 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]. [2013 – 5 – 25], http: // wwww. ifap. ru/ library/book268. pdf.

[8] Global Action Plan. An inefficient truth [EB/OL]. Global action plan report, 2007 [2011 – 02 – 12], http: //globalactionplan. org. uk.

[9] TIMES N Y. Power, Pollution and the Internet [EB/OL]. [2013 – 5 – 20], http: // www. nytimes. com/2012/09/23/ technology/ data-ceneters-waste-vast-amounts-of-energy-belying-industry-image. html.

[10] 于炯,廖彬,张陶,等. 云存储系统节能研究综述[J]. 计算机科学与探索, 2014, 8(9): 1025 – 1040.

[11] 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] // Proceedings of the 4th Workshop on Hot Topics in Syetms Dependability (HotDep '08), San Diego, USA, 2008.

[12] WEDDLE C, OLDHAM M, QIAN J, et al. A gear-shifting power-aware raid [J]. ACM Transactions on Storage, 2007, 3(3): 1553 – 1569.

[13] LI D, WANG J. Conserving energy in conventional disk based RAID systems [C] // Proceedings of the 3rd International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI'05). Piscataway, NJ: IEEE, 2005: 65 – 72.

[14] YAO X Y, WANG J. Rimac: A novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems [C] // Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys 06), Leuven, Belgium, 2006. New York, USA: ACM, 2006: 249 – 262.

[15] PINHEIRO E, BIANCHINI R, DUBNICKI C. Exploiting redundancy to conserve energy in storage systems [C] // Proceedings of the joint international conference on Measurement and modeling of computer systems (SIGMetrics'06/Performance '06), Saint Malo, France, 2006. New York, USA: ACM, 2006: 15 – 26.

[16] COLARELLI D,GRUNWALD D. Massive arrays of idle disks for storage archives [C] // Proceedings of the 2002 ACM/IEEE conference on Supercomputing (SC'02), Baltimore, USA, 2002. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002: 1 – 11.

[17] NARAYANAN D, DONNELLY A, ROWSTRON A. Write off-loading: Practical power management for enterprise storage [J]. ACM Transactions on Storage, 2008, 4(3): 253 – 267.

[18] STORER M, GREENAN K, MILLER E, et al. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage [C] // Proceedings of the 6th USENIX Conference on File and Storage Technologies (2008), San Jose, USA, 2008. Berkeley, CA, USA: USENIX Association, 2008: 1 – 16.

[19] ZHU Q B, CHEN Z F, TAN L, et al. Hibernator: Helping disk arrays sleep through the winter [C] // Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05), Brighton, UK, 2005. New York, NY, USA: ACM, 2005: 177 – 190.

[20] VASIC N, BARISITS M, SALZGEBER V. Making cluster applications energy-aware [C] // Proceedings of the 1st Workshop on Automated Control for Datacenters

and Clouds (ACDC'09), Barcelona, Spain, 2004. New York, NY, USA: ACM, 2009: 37 - 42.

[21] ZHU Q B, DAVIDF M, DEVARAJ C F, et al. Reducing energy consumption of disk storage using power-aware cache management [C]// Proceedings of the 10th International Conference on High-Performance Computer Architecture (HPCA'04), Madrid, Spain, Piscataway, NJ, USA: IEEE, 2004: 118 - 129.

[22] MOHAMED Y, TIAN Y Y, OZCAN F, et al. CoHadoop: flexible data placement and its exploitation in Hadoop [C]// Proceedings of the VLDB Endowment VLDB 2011, 2011, 4(9): 575 - 585.

[23] DITTRICH J, QUIANE-RUIZ J A, JINDAL A, et al. Hadoop + +: Making a yellow elephant run like a cheetah(without it even noticing) [C]// Proceedings of the PVLDB 2010, 2010, 3(1/2): 518 - 529.

[24] ZHAO Y R, WANG W P, MENG D, et al. A data locality optimization algorithm for large-scale data processing in Hadoop [C]// Processings of the ISCC 2012, 2012: 655 - 661.

[25] 赵彦荣,王伟平,孟丹,等. 基于 Hadoop 的高效连接查询处理算法 CHMJ[J]. 软件学报, 2012, 23(8): 2023 - 2041.

[26] 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.

[27] 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.

(上接第 40 页)

参考文献:

[1] 刘军. 科学计算中的蒙特卡罗策略[M]. 北京:高等教育出版社,2009.

[2] 洪志敏. 基于 Monte-Carlo 技术的积分(微分)方程数值求解方法研究[D]. 内蒙古:内蒙古工业大学博士学位论文.

[3] 左应红,王建国. 蒙特卡罗方法在解微分方程边值问题中的应用[J]. 强激光与粒子束,2012, 24(12): 3023 - 3026.

[4] 周泉,陈植武,詹杰民. 蒙特卡罗法的改进:一种新型的快速算法[C]//第二十一届全国水动力学研讨会暨第八届全国水动力学学术会议暨两岸船舶与海洋工程水动力学研讨会文集, 2008.

[5] 林建国,周俊陶. 蒙特卡罗法的一种快速计算:在势流问题中的应用[J]. 水动力学研究与进展 A 辑,2005, 20(3): 405 - 410.

[6] 史慧会,曲小钢. 蒙特卡罗方法在热传导方程中的应用[J]. 重庆工学院学报:自然科学版,2008, 22(11): 101 - 103.

[7] VAJARGAH B F, VAJARGAH K F. Monte Carlo method for finding the solution of Dirichlet partial differential equations [J]. Applied Mathematical Sciences, 2007, 1(10): 453 - 462.

[8] LAUDU D P, BINDER K. A guild to Monte Carlo simulations in statistical physics [M]. Cambridge University Press, 2009.

[9] JUN S, SUNGHWAN Y, NAM Z C. A Monte Carlo method for solving heat conduction problems with complicated geometry [J]. Nuclear Engineering and Technology, 2007, 29(3): 207 - 214.