基于 Storm 平台的数据迁移合并节能策略

蒲勇霖[1]，于炯[1]，鲁亮[2]，李梓杨[1]，卞琛[3]，廖彬[4]

（1. 新疆大学信息科学与工程学院，新疆 乌鲁木齐 830046；2. 中国民航大学计算机科学与技术学院，天津 300300；
3. 广东金融学院互联网金融与信息工程学院，广东 广州 510521；4. 新疆财经大学统计与信息学院，新疆 乌鲁木齐 830012）

摘 要：针对 Storm 存在低效率、高能耗的问题，通过分析 Storm 平台的基本框架与拓扑结构，设计了资源约束模型、最优线程数据重组原则和节点降压原则，并在此基础上提出了基于 Storm 平台的数据迁移合并节能策略（DMM-Storm），包括资源约束算法、数据迁移合并算法和节点降压算法。其中资源约束算法根据资源约束模型，判断工作节点是否允许数据的迁移；数据迁移合并算法根据最优线程数据重组原则，设计了最优的线程数据迁移方法；节点降压算法根据节点降压限制条件，降低了工作节点的电压。实验结果表明，与现有的节能策略相比，执行 DMM-Storm 在不影响集群性能的前提下，有效降低了能耗。

关键词：大数据；Storm；资源约束；数据迁移；能耗

中图分类号：TP311

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2019226

Energy-efficient strategy for data migration and merging in Storm

PU Yonglin[1], YU Jiong[1], LU Liang[2], LI Ziyang[1], BIAN Chen[3], LIAO Bin[4]

1. School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
2. School of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China
3. College of Internet Finance and Information Engineering, Guangdong University of Finance, Guangzhou 510521, China
4. School of Statistics and Information, Xinjiang University of Finance and Economics, Urumqi 830012, China

Abstract: Storm is suffering the problems of high energy consumption but low efficiency. Aiming at this problem, the resource constraint model, the optimal principle of data reorganization in executors and node voltage reduction principle were proposed based on the analysis of the architecture and topology of Storm, and further the energy-efficient strategy for data migration and merging was put forward in Storm(DMM-Storm), which was composed of resource constraint algorithm, data migration and merging algorithm as well as node voltage reduction algorithm. The resource constraint algorithm estimates whether work nodes are appropriate for data migration according to the resource constraint model. The data migration and merging algorithm designs an optimal method to migrate data according to the the optimal principle of data reorganization in executors. The node voltage reduction algorithm reduces voltage of work nodes according to node voltage reduction principle. The experimental results show that the DMM-Storm can reduce energy consumption efficiently without affecting the performance of cluster compared with the existing researches.

Key words: big data, Storm, resource constraint, data migration, energy consumption

收稿日期：2019–07–04；**修回日期**：2019–10–19

基金项目：国家自然科学基金资助项目（No.61862060, No.61462079, No.61562086, No.61562078）；新疆维吾尔自治区研究生科研创新基金资助项目（No.XJGRI2016028）；新疆大学博士生科技创新基金资助项目（No.XJUBSCX-201902）

Foundation Items: The National Natural Science Foundation of China (No.61862060, No.61462079, No.61562086, No.61562078), The Research Innovation Project of Graduate Student in Xinjiang Uygur Autonomous Region (No.XJGRI2016028), The Doctoral Innovation Program of Xinjiang University (No.XJUBSCX-201902)

1　引言

近年来，随着移动互联网、云计算和物联网的高速发展，数据的产生与积累已达到前所未有的速度，并推动着各行各业进入大数据时代。大数据的相关研究已成为学术界与产业界共同探讨的热点话题，其主要的处理模式为批量处理、流式处理、交互处理和图处理等[1-5]。随着大数据的飞速发展，各种处理模式不断产生，出现了高能耗、高污染的。因此如何有效解决由于新兴技术带来的高能耗，一直是广大学者共同面对的问题。据美国斯坦福大学的调查发现，全球数据中心 2010 年的耗电量为 2 355 亿 kW·h，占据全球电力总消耗的 1.3%，其中美国数据中心的耗电量占据全美国电力总消耗的 2%[6]。我国数据中心的电力消耗同样惊人，截至 2011 年底，我国数据中心的总量达到 43 万个，数据中心的耗电量占据全国电力总消耗的 1.5%，并且所占比例仍在逐年上升[7]。随着大数据时代的到来，海量的能源被用于数据处理，但其能效不断降低。因此如何有效提高能源的利用率，是解决大数据处理能耗问题的关键。

大数据的处理一般被分为实时处理与非实时处理，在 IDC 与希捷（Seagate）公司联合发布的《数据时代 2025》白皮书中指出，2025 年全球数据总量将达到 163 ZB，比 2016 年创造出的数据量提高了 10 倍。其中实时数据所占比例超过 25%[8]。由此可见，实时大数据拥有广泛的应用前景，而在大数据的处理模式中，流式处理具有很高的实时性。流式处理作为新的可容错、高性能的分布式处理平台，存在着高能耗的问题[9]，已经给产业界带来了巨大的能耗开销。因此流式处理平台的节能优化是一个亟待解决的问题。

现有流式处理平台以 Apache Storm 框架[10]为代表。Storm 是一个开源、主从式架构、横向扩展性良好且容错能力强的分布式实时处理平台，其编程模型简单，支持包含 Java 在内的多种编程语言，数据处理高效。相较于不开源的 Puma[11]以及社区冷淡的 S4[12]，Storm 拥有更广阔的发展前景；相较于目前主流的实时处理平台 Flink[13]与 Spark Streaming[14]，Storm 的数据处理实时性能效果更佳；相较于后起之秀 Heron[15]，Storm 更加简单且业界的认可度与成熟度更高。目前 Storm 已经广泛运用到银行金融[16]、临床医疗[17]、社交网络[18]等行业进行实时大数据分析，并广泛运用到机器学习算法、分布式远程调用等领域进行理论研究[19]，被誉为“实时处理领域的 Hadoop”。

在 Storm 集群中，一个流式作业（拓扑）通过有向无环图（DAG, directed acyclic graph）表示，且拓扑内的数据通过轮询（RR, round-robin）调度策略均匀分配到各个工作线程上，而一个工作进程包含多个工作线程。Storm 集群采用轮询调度策略均匀分配数据需要符合 8 种流组模式，其中最重要的是随机分组（shuffle grouping）。但是随机分组存在两大弊端，致使 Storm 集群产生额外资源与能源的浪费。其一是数据流经工作节点时，有些线程数据处理复杂度较低，随机分组致使线程因计算资源过剩而产生浪费；其二是未考虑节能的问题，对于数据处理复杂度较低的线程，并未对存在该线程的节点进行节能处理，造成能源的浪费。因此要解决因随机分组而带来的资源与能源的浪费问题，需要综合考虑集群的路径、工作节点的计算资源、数据的匹配等各方面的因素，寻找最优的数据分配策略，从而在保证性能的前提下实现节约资源与能源的目的。本文的主要贡献如下。

1) 通过分析 Storm 集群的拓扑结构，建立数据分配模型、路径开销模型与资源约束模型，进一步提出最优线程数据重组原则，为数据迁移合并模型的建立奠定了基础。

2) 根据资源约束模型、数据迁移合并模型和节点降压原则，提出基于 Storm 平台的数据迁移合并节能策略（DMM-Storm, energy-efficient strategy for data migration and merging in Storm），该策略包括资源约束算法、数据迁移合并算法和节点降压算法。其中，资源约束算法验证工作节点是否允许数据迁移，数据迁移合并算法根据集群拓扑的实际情况，确定数据迁移的最优方法，并为节点降压算法提供了理论支撑。节点降压算法根据数据迁移合并算法与节点降压原则的特点，降低非关键路径上工作节点的电压以减少集群中的能耗损失。最后通过实验从多角度验证了算法的有效性。

2　相关工作

学术界与产业界针对现有大规模数据处理平台节能方面的研究主要分为 4 类：批量数据处理平台节能算法、流式数据处理平台节能算法、图数据

处理平台节能算法和交互数据处理平台节能算法。其中批量数据处理平台节能算法的核心思想主要是以 Hadoop 为代表进行算法的优化，通常对框架内的磁盘区域进行划分，通过动态组件失活（dynamic component deactivation），即在一段时间内动态关闭集群硬件的部分组件或对磁盘部分区域进行休眠达到节能的目的[20-22]；图数据处理平台节能算法的核心思想主要是以 Pregel 为代表进行算法的优化，通常对图边缘数据的重要性进行判别，弹性调节集群的功耗达到节能的效果[23-25]；交互数据处理平台节能算法的核心思想主要是以 MapReduce 为代表进行算法的优化，通常以优化配置参数[26]、作业迁移调度[27]和任务完成后关闭对应节点[28]等提高能源利用率达到节能的目的[29-31]。这 3 种方案在一定程度上解决了大数据处理平台的能耗问题，但是对于实时性较高的数据处理模式存在着较大的局限性，无法直接作用于流式处理的平台。针对流式大数据处理模式的高能耗问题，现有学者提出从硬件[32]、软件[33]以及两者结合[34]这 3 个方面进行研究。

硬件的节能策略主要通过替换高能耗的电子元件[35]，以达到节能的效果。该方法节能效果显著且操作简单，但其价格高昂，不适合部署于大规模的集群当中。软件的节能策略主要通过任务调度[36]、资源迁移[37]等方法以提高集群的性能，达到节能的目的，但由于节能效果不稳定致使节能效果不佳。软件与硬件结合的节能策略是现在研究的重点，主要通过在任务完成后动态调节集群的电压或电源，实现集群电压或电源的缩放管理[38]，以达到节能的目的。Cordeschi 等[39]针对流式大数据传输的不可控、不稳定以及实时数据量大等特性，在不影响响应时间约束条件的前提下，计算了最小化网络传输的总能耗。Wang 等[38]通过使用动态电压调控技术（DVFS, dynamic voltage frequency scaling）调节集群 CPU 的电压以达到节能的目的。Panda 等[40]通过引入上下文感知数据流执行模型，提高了任务执行的效率，从侧面降低了集群的能耗。综上所述，以上研究都是从流式处理自身特性的角度出发，建立合理的流式处理能耗模型。但针对 Storm 平台的节能策略仍具有较高的研究价值。

Sun 等[9]提出一种流式大数据处理环境下的实时资源调度节能策略（Re-Stream），该策略通过建立集群响应时间、CPU 占用率以及能耗之间的逻辑关系，并根据流式处理框架的基本性质，对整个集群拓扑执行的关键路径进行定义，综合利用拓扑执行关键路径上性能感知的任务调度策略，以及拓扑执行非关键路径上能耗感知的任务整合策略，使任务响应时间和集群能耗均降低到最低值。

Zong 等[41]根据流式大数据处理的自身特性提出 2 种基于副本的调度节能算法——能量感知副本算法（EAD, energy-aware duplication）以及性能和能量均衡副本算法（PEBD, performance-energy balanced duplication），该节能算法的核心思想为集群拓扑不执行任务调度时或数据处理完成后，立刻降低集群节点的电压。该算法不仅保证了集群数据处理的快速执行，同时满足任务执行完成后节约集群能耗的思想。

蒲勇霖等[42]针对 Storm 平台在进行数据处理时存在高能耗的问题，提出工作节点内存电压调控节能策略（WNDVR-Storm），该策略根据 Storm 集群实际的数据处理及传输情况，对集群工作节点数据处理能力进行判别，并由集群数据流实际情况，通过对工作节点的内存电压进行动态调节，以达到节能的目的。该节能策略不仅有效降低了集群的能耗，而且在一定程度上对集群的负载均衡进行了优化。但是还存在以下两点不足：1) 对集群工作节点内存电压进行动态调节，实现难度较高且存在一定的偶然性；2) 若集群规模较大且工作节点过多，节能算法可能失效。

本文与文献[9,38-42]的不同之处主要体现在 4 个方面。首先，本文通过分析集群进行数据迁移时，工作节点存在资源约束（CPU、网络带宽和内存）的问题而构建资源约束模型，防止集群由于数据迁移而造成资源溢出的问题。其次，本文根据最优线程数据重组原则与节点降压原则，完成集群的数据迁移与节点降压，该过程不会影响集群的性能，并节约了集群的能耗。第三，本文不仅节约了集群的能耗，还通过数据迁移算法减少了节点之间的通信开销。最后，本文选取 Intel 公司[10]发布在 GitHub 上的基准测试而非自己定义的拓扑，因此更具代表性。

3 问题建模与分析

本节主要从 Storm 拓扑的数据分配模型、路径与成本模型和资源约束模型出发，分析 Storm 集群在节能方面存在的局限性，由此提出非关键

线程中的数据迁移合并与关键线程中的数据迁移合并这 2 种模型，为节能策略的设计与实现提供了理论依据。

3.1　数据分配模型

在 Storm 集群中，一个流式作业由一个拓扑（topology）的有向无环图构成，其中数据源编程单元（spout）和数据处理编程单元（bolt）这 2 类组件（component）共同构成了数据源的顶点，且各组件之间针对不同的流组模式发送和接收数据流，进而构成了有向无环图的弧。为提高拓扑对数据流的并行运算能力，令每个组件都可同时创造多个线程（executor）。提交拓扑之后，数据将分发到集群各工作节点中，并进行数据的处理与传输。设集群工作节点集合为 $N(C)=\{n_1,n_2,\cdots,n_m\}$，线程集合为 $E(C)=\{e_{j1},e_{j2},\cdots,e_{jn}\}$，将工作节点的数据均匀分配到集群的线程上，记工作节点分配给线程 e_{ji} 的数据为 da_{ji}（若线程的并行度为 1，则该线程上的数据为 da_j），则集合 $\mathrm{DA}_n=\{\mathrm{da}_{j1},\mathrm{da}_{j2},\cdots,\mathrm{da}_{jn}\}$ 表示工作节点分配到线程上的数据集合。图 1 为在 3 个工作节点下，Storm 集群使用轮询调度策略后线程数据的分配情况。其中工作节点的集合为 $N=\{n_1,n_2,n_3\}$，线程内数据可表示为 $\mathrm{DA}_{n_1}=\{\mathrm{da}_a,\mathrm{da}_{d1},\mathrm{da}_{f1},\mathrm{da}_h\}$，$\mathrm{DA}_{n_2}=\{\mathrm{da}_b,\mathrm{da}_{d2},\mathrm{da}_{f2},\mathrm{da}_i\}$，$\mathrm{DA}_{n_3}=\{\mathrm{da}_c,\mathrm{da}_{d3},\mathrm{da}_e,\mathrm{da}_j\}$。

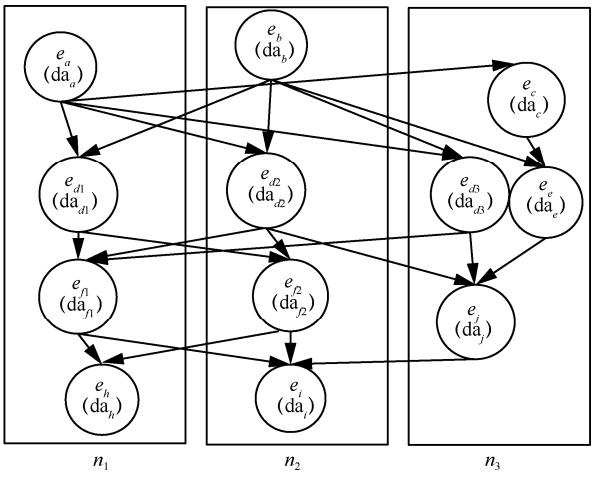


图 1　线程数据分配

为了消除节点内部进程间通信开销，图 1 为每个工作节点仅分配一个进程（worker），因此在拓扑执行过程中只需考虑 2 类通信成本开销，一类为线程数据 da_{d1} 与 da_{f1} 之间，节点内部线程间的通信开销；另一类为线程数据 da_{d2} 与 da_{f1} 之间，节点间通信开销。无论工作节点如何分配数据，都满足以上 2 种通信开销。

此外，令线程 e_a 与 e_b 为线程 $\{e_c,e_{d1},e_{d2},e_{d3},e_e\}$ 的父线程，若线程 $\{e_c,e_{d1},e_{d2},e_{d3}\}$ 为父线程 e_a 下的子线程，则线程 $\{e_c,e_{d1},e_{d2},e_{d3}\}$ 内的线程互为同源线程；若线程 $\{e_{d1},e_{d2},e_{d3},e_e\}$ 为父线程 e_b 下的子线程，则线程 $\{e_{d1},e_{d2},e_{d3},e_e\}$ 内的线程互为同源线程，以此类推，完成父线程与同源线程之间的对应关系。

3.2　路径开销模型

令一条路径 $p(e_{ji}, e_{mn})$ 为集合 $B(p(e_{ji}, e_{mn}))$ 的子路径，其中顶点 e_{ji} 与 e_{mn} 表示从 e_{ji} 开始到 e_{mn} 结束。对于 $\exists k$，则 $b_{j,k}\in p(e_{ji},e_{mn})$，$b_{k,i}\in p(e_{ji},e_{mn})$，由此对于 $\forall b_{j,i}\in p(e_{ji},e_{mn})$ 都成立。此外，对于 $\forall b_{k,l}\in p(e_{ji},e_{mn})$，如果 $k\neq j$，则 $\exists m$ 与 $b_{m,k}\in p(e_{ji},e_{mn})$；如果 $i\neq j$，则 $\exists m$ 与 $b_{l,m}\in p(e_{ji},e_{mn})$。

路径开销 $l_p(e_{ji},e_{mn})$ 表示从顶点 e_{ji} 到 e_{mn} 所有线程与有向边的开销之和，则有

$$l_p(e_{ji},e_{mn})=\sum_{e_{ji}\in E(p(e_{ji},e_{mn}))}e_{ji}+\sum_{b_{j,i}\in B(p(e_{ji},e_{mn}))}b_{j,i}\quad(1)$$

令路径的计算开销 $W^{\text{executor}}_{\text{computation cost}}$ 为线程的计算时间，路径的传输开销 $W^{\text{edge}}_{\text{communication cost}}$ 为线程之间的传输时间，则路径的总开销 W_{cost} 为

$$W_{\text{cost}}=W^{\text{executor}}_{\text{computation cost}}+W^{\text{edge}}_{\text{communication cost}}\quad(2)$$

其中，路径的传输开销 $W^{\text{edge}}_{\text{communication cost}}$ 由节点间通信开销、节点内部进程间通信开销与节点内部线程间的通信开销组成，根据 3.1 节可知，节点内部进程间通信开销为 0，则

$$W_{\text{cost}}=W^{\text{executor}}_{\text{computation cost}}+W^{\text{edge}}_{\text{executor cost}}+W^{\text{edge}}_{\text{node cost}}\quad(3)$$

其中，$W^{\text{edge}}_{\text{node cost}}$ 表示节点间通信开销，$W^{\text{edge}}_{\text{executor cost}}$ 表示节点内部线程间通信开销。此外，如果线程 e_{ji} 和 e_{mn} 位于相同的工作节点，则线程间的通信开销可忽略不计，则

$$W_{\text{cost}}=W^{\text{executor}}_{\text{computation cost}}+W^{\text{edge}}_{\text{node cost}}\quad(4)$$

令整个拓扑存在 m 条路径，则拓扑执行关键路径 $l(G_p)$ 为

$$l(G_p)=\max\{l_{p_1}(e_{ji},e_{mn}),l_{p_2}(e_{ji},e_{mn}),\cdots,l_{p_m}(e_{ji},e_{mn})\} \tag{5}$$

此外，在确保 Storm 集群性能不变的前提下，根据数据在拓扑关键路径处理及传输时间，确定位于拓扑执行关键路径上的工作节点与位于拓扑执行非关键路径上的工作节点。将位于拓扑执行关键路径上的工作节点称作关键节点，位于拓扑执行非关键路径上的工作节点称作非关键节点，位于拓扑执行关键路径上的线程称作关键线程，位于拓扑执行非关键路径上的线程称作非关键线程。

以图 2 为例，定义一条拓扑执行关键路径为 $e_a \to e_{d1} \to e_{f2} \to e_h$，则工作节点 n_3 上的所有线程为非关键节点上的非关键线程，工作节点 n_1 与 n_2 上的线程分为 2 类：一类为关键节点上的关键线程，另一类为关键节点上的非关键线程。此外，线程 e_c 与线程 $\{e_{d1},e_{d2},e_{d3}\}$ 互为同源线程，以此类推，完成非关键节点上非关键线程与关键节点上线程的对应关系。

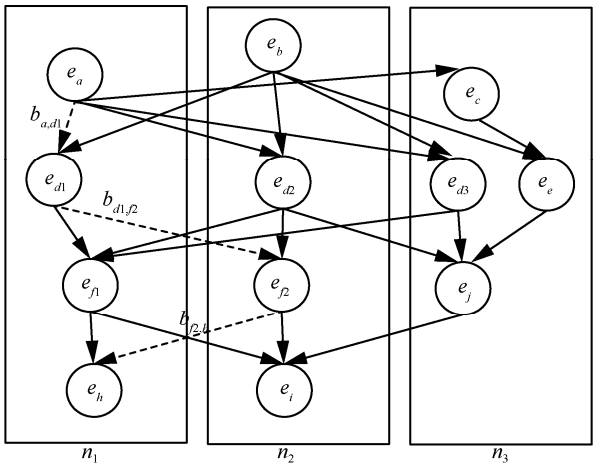


图 2 拓扑执行关键路径的数据传输及处理情况

3.3 资源约束模型

根据 Storm 集群进行数据迁移的特点，令工作节点分配给线程的资源集合为 $\mathrm{RE}_N=\{\mathrm{re}_{n_1},\mathrm{re}_{n_2},\cdots,\mathrm{re}_{n_m}\}$，且计算资源主要包括 CPU、内存及网络带宽 3 类资源的占用率。令工作节点内 3 类资源占用的极限为 $\mathrm{RE}_N=(\mathrm{RE}_N^{\mathrm{C}},\mathrm{RE}_N^{\mathrm{M}},\mathrm{RE}_N^{\mathrm{B}})$。其中 $\mathrm{RE}_N^{\mathrm{C}}$ 表示工作节点 CPU 资源占用的极限，$\mathrm{RE}_N^{\mathrm{C}}=\{\mathrm{re}_{n_1}^{\mathrm{C}},\mathrm{re}_{n_2}^{\mathrm{C}},\cdots,\mathrm{re}_{n_m}^{\mathrm{C}}\}$；$\mathrm{RE}_N^{\mathrm{M}}$ 表示工作节点内存资源占用的极限，$\mathrm{RE}_N^{\mathrm{M}}=\{\mathrm{re}_{n_1}^{\mathrm{M}},\mathrm{re}_{n_2}^{\mathrm{M}},\cdots,\mathrm{re}_{n_m}^{\mathrm{M}}\}$；$\mathrm{RE}_N^{\mathrm{B}}$ 表示工作节点网络带宽资源占用的极限，$\mathrm{RE}_N^{\mathrm{B}}=\{\mathrm{re}_{n_1}^{\mathrm{B}},\mathrm{re}_{n_2}^{\mathrm{B}},\cdots,\mathrm{re}_{n_m}^{\mathrm{B}}\}$。原工作节点 CPU 资源占用为 $\mathrm{ore}_{n_i}^{\mathrm{C}}$（单位为 Hz），内存资源占用为 $\mathrm{ore}_{n_i}^{\mathrm{M}}$（单位为 B），网络带宽资源占用为 $\mathrm{ore}_{n_i}^{\mathrm{B}}$（单位为 bit/s）。由于 Storm 集群环境中数据源源不断产生，且拓扑一旦提交将持续运行下去，因此为了保证集群的高效运行，且工作节点的资源不会溢出，这 3 类资源需要满足如下条件，即

$$\sum_{\forall \mathrm{re}_{n_i}\in \mathrm{RE}_N}\mathrm{ore}_{n_i}^{\mathrm{C}}\leqslant \mathrm{re}_{n_i}^{\mathrm{C}} \tag{6}$$

$$\sum_{\forall \mathrm{re}_{n_i}\in \mathrm{RE}_N}\mathrm{ore}_{n_i}^{\mathrm{M}}\leqslant \mathrm{re}_{n_i}^{\mathrm{M}} \tag{7}$$

$$\sum_{\forall \mathrm{re}_{n_i}\in \mathrm{RE}_N}\mathrm{ore}_{n_i}^{\mathrm{B}}\leqslant \mathrm{re}_{n_i}^{\mathrm{B}} \tag{8}$$

为了满足集群运行的可靠性，集群工作节点不能满负荷运行，本文将符合 CPU 资源的正常计算称为满足 CPU 资源临界原则，符合内存资源的正常计算称为满足内存资源临界原则，符合网络带宽资源的正常传输称为满足网络带宽资源临近原则。

当线程准备迁入数据时，该工作节点资源满足 CPU 资源临界原则、内存资源临界原则以及网络带宽资源临近原则时，允许线程迁入数据。即数据迁入原则 tr 需要满足如下条件

$$\mathrm{re}_{n_i}^{\mathrm{C}'}+\sum_{\forall \mathrm{re}_{n_i}\in \mathrm{RE}_N}\mathrm{ore}_{n_i}^{\mathrm{C}}\leqslant \mathrm{re}_{n_i}^{\mathrm{C}} \tag{9}$$

$$\mathrm{re}_{n_i}^{\mathrm{M}'}+\sum_{\forall \mathrm{re}_{n_i}\in \mathrm{RE}_N}\mathrm{ore}_{n_i}^{\mathrm{M}}\leqslant \mathrm{re}_{n_i}^{\mathrm{M}} \tag{10}$$

$$\mathrm{re}_{n_i}^{\mathrm{B}'}+\sum_{\forall \mathrm{re}_{n_i}\in \mathrm{RE}_N}\mathrm{ore}_{n_i}^{\mathrm{B}}\leqslant \mathrm{re}_{n_i}^{\mathrm{B}} \tag{11}$$

其中，$\mathrm{re}_{n_i}^{\mathrm{C}'}$ 表示迁入数据后增加的工作节点 CPU 资源占用率，$\mathrm{re}_{n_i}^{\mathrm{M}'}$ 表示迁入数据后增加的工作节点内存资源占用率，$\mathrm{re}_{n_i}^{\mathrm{B}'}$ 表示迁入数据后增加的工作节点网络带宽资源占用率。

为了在保证拓扑高效执行的同时，提高资源的占用率，且在一定程度上优化集群的负载分配，资源约束模型为后续数据重分配原则提供了理论依据。

3.4 拓扑非关键线程中的数据迁移合并模型

根据 3.3 节资源约束模型，提出最优线程数据重组原则；本节针对存在关键节点上的非关键线程，提出非关键线程中的数据迁移合并模型。此外，根据集群数据传输及处理总时间，确定了拓扑执行

非关键路径工作节点电压的最低值。

定义 1　最优线程数据重组原则。根据 3.1 节可知，$E'(C)\in E(C)$ 且 $E'(C)=\{e_{ji},e_{mn},\cdots,e_{yz}\}$ 为同源线程集合。同源线程之间数据的迁移合并可通过父线程进行数据的重分配，而非同源线程之间进行数据的迁移合并可能会出现数据不匹配问题，因此线程之间数据的迁移合并需要选择互为同源的线程。定义父线程 e_{ab} 传入子线程 e_{ji} 与 e_{mn} 数据大小分别为 $\mathrm{da}_{ab,ji}$ 与 $\mathrm{da}_{ab,mn}$，执行最优线程数据重组原则后，父线程对子线程的数据分配出现改变，类似数据由线程 e_{mn} 迁入线程 e_{ji}，且迁入的数据大小 $\mathrm{da}_{ji_{\mathrm{input}}}$ 为

$$\mathrm{da}_{ji_{\mathrm{input}}}=\sum_{f(e_{mn})\to f(e_{ji})}\mathrm{da}_{mn,ji} \tag{12}$$

由 3.3 节可知，线程之间数据的迁移合并，被迁入数据的线程存在工作节点的资源约束问题，则为满足资源约束条件，存在

$$\mathrm{re}_{n_{i_{\mathrm{input}}}}+\mathrm{ore}_{n_i}\leqslant\mathrm{re}_{n_i} \tag{13}$$

其中，$\mathrm{re}_{n_{i_{\mathrm{input}}}}$ 表示线程迁入数据后增加的工作节点资源占用情况，ore_{n_i} 表示原工作节点资源的占用情况，re_{n_i} 表示工作节点资源占用的临界值，式(13)表示被迁入数据的线程满足 tr。

根据最优线程数据重组原则设计数据迁移合并模型，为非关键节点的降压奠定了基础。针对是否存在关键节点上的非关键线程，提出 2 种线程中的数据迁移合并模型，当集群存在关键节点上的非关键线程时，集群实施拓扑非关键线程中的数据迁移合并模型，且该模型不改变集群性能；当集群不存在关键节点上的非关键线程时，集群实施拓扑关键线程中的数据迁移合并模型，且该模型对系统性能造成一定的影响需要进行相应的评估。

定义 2　节点降压原则。为计算集群实施节能策略后非关键节点电压的最低值，需要注意以下两点：其一，确定非关键节点电压的限制条件；其二，确定非关键节点电压的改变量。

集群实施非关键线程中的数据迁移合并模型，需要令非关键节点的集合为 $N'(C)=\{n'_1,n'_2,\cdots,n'_m\}$，其中 $\forall n'_i\in N'(C)$，模型以不改变 Storm 集群性能为前提，因此不能改变拓扑执行关键路径，由此可知模型的执行存在一个限制条件，即数据传输及处理的总时间不会发生改变。根据 3.3 节可知，线程迁入数据存在制约条件，即线程 e_{ji} 迁入数据需要满足 tr。由于线程迁入数据为 $\mathrm{da}_{n'_i}$，关键路径的时间 t 不发生改变，则非关键节点 n'_i 的电压为 $\mathrm{ve}_{n'_i}$，此时调节电压 $\mathrm{ve}_{n'_i}$ 保证关键路径的时间不发生改变，获得改变后的电压为 $\mathrm{ve}'_{n'_i}$。由此存在一个函数关系 $F_{n'_i}(\mathrm{ve}_{n'_i},\mathrm{da}_{n'_i},\mathrm{ve}'_{n'_i})$ 为

$$F_{n'_i}(\mathrm{ve}_{n'_i},\mathrm{da}_{n'_i},\mathrm{ve}'_{n'_i})=\phi\mathrm{ve}'_{n'_i}-\mathrm{ve}_{n'_i} \tag{14}$$

其中，ϕ 表示数据迁移合并过程中资源的损耗率，且 $0<\phi<1$。电压的改变过程符合贪心原则，未达到贪婪状态则无法确定电压调节的最低值，因此为计算电压调节的改变量 vc，引入梯度下降的方法

$$\mathrm{vc}=\alpha(t(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}'_{i+1})+\phi\mathrm{ve}_{i+1}-\mathrm{ve}_i)\cdot\sum_{s=0}^{i}(\phi\varepsilon)^{i-s}\frac{\partial}{\partial\mathrm{vc}}\mathrm{ve}_s \tag{15}$$

其中，ε 表示电压改变轨迹；$t(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}'_{i+1})$ 表示关键路径时间不发生改变；α 表示训练率，且 $0<\alpha<1$。当满足上述条件时，非关键节点 n'_i 的电压以 vc 为步长逐渐降低，直至达到条件改变的临界点，由此可得集群实施非关键线程中的数据迁移合并模型后，非关键节点电压的最低值。由于电压调节过程中集群关键路径不发生改变，则集群数据传输与处理的总时间不发生改变，因此集群的性能不发生改变。

3.5　拓扑关键线程中的数据迁移合并模型

本节针对不存在关键节点上的非关键线程，提出关键线程中的数据迁移合并模型。通过定义能效比（EER, energy efficiency ratio），确定集群性能与能耗之间的关系，并根据能效比，确定拓扑执行关键路径工作节点电压的最低值。

模型以不改变 Storm 集群的性能为前提，即不改变数据传输及处理的时间。关键线程中的数据迁移合并模型对集群数据的处理及传输造成一定的影响，因此需要对集群的性能与能耗进行评估。

定义 3　能效比。通过 3.2 节可知，集群数据处理及传输的总时间为 t（单位为 s），定义集群数据处理及传输的总能耗为 E（单位为 J），则建立能效比模型，即集群数据处理及传输总时间与总能耗乘积的倒数为能效比 R（单位为 $(\mathrm{s}\cdot\mathrm{J})^{-1}$），即

$$R=\frac{\mathrm{C}}{tE} \tag{16}$$

其中，C 表示能效比误差常数，且 $0<\mathrm{C}<1$。此外，

能效比越大表示节能策略的实用价值越高。

定理 1 Storm 集群在实际拓扑执行过程中，数据处理及传输总时间与总能耗的能效比 R 为

$$R=\frac{\mathrm{C}}{E\sum_{i=1}^{n-1}(W_{\text{computation cost}}^{\text{executor}}+W_{\text{node cost}}^{\text{edge}})} \tag{17}$$

证明 通过定义 3 可知，集群数据处理及传输总时间与总能耗的比值 $R=\frac{\mathrm{C}}{tE}$，则 $\frac{1}{R}=\frac{tE}{\mathrm{C}}$。如果将总时间划分为 n 等份，t 为 $[t_0,t_1,\cdots,t_{n-1}]$，则有

$$\frac{\mathrm{C}}{R}=\int_{t_0}^{t_{n-1}}E\mathrm{d}t \tag{18}$$

化简式(18)得

$$\frac{\mathrm{C}}{R}=E(t_{n-1}-t_0) \tag{19}$$

因此，获得

$$R=\frac{\mathrm{C}}{(t_{n-1}-t_0)E} \tag{20}$$

通过定理 1 可知，集群数据处理及传输的总时间 $t=W_{\text{computation cost}}^{\text{executor}}+W_{\text{node cost}}^{\text{edge}}$。当时间间隔被划分成 n 等份时，有 $\forall t_i\in t$，t=$[t_0,t_1,\cdots,t_{n-1}]$，其中 t_i 表示总时间 t 内的一段时间间隔，则获得式(17)，计算了集群的能效比。

证毕。

为计算集群实施关键线程中的数据迁移合并模型后的总能耗，需要计算集群非关键节点电压的最低值。根据定义 2 可知，存在 2 个制约条件，其中实施关键线程中的数据迁移合并模型后，系统的能效比与原系统能效比之间的关系为一个制约条件。已知集群非关键节点的集合为 $N'(C)=\{n_1',n_2',\cdots,n_m'\}$，通过 3.3 节可知，线程迁入数据存在制约条件，则线程数据 da 迁入需要满足 tr。由于线程迁入数据 $\mathrm{da}_{n_{i'}'}$，由定义 3 可知集群的能效比为 R，调节此时非关键节点 $n_{i'}'$ 的电压为 $\mathrm{ve}_{n_{i'}'}$。同理，可获得函数 $F_{n_{i'}'}(\mathrm{ve}_{n_{i'}'},\mathrm{da}_{n_{i'}'},\mathrm{ve}'_{n_{i'}'})$ 为

$$F_{n_{i'}'}(\mathrm{ve}_{n_{i'}'},\mathrm{da}_{n_{i'}'},\mathrm{ve}'_{n_{i'}'})=\phi\mathrm{ve}'_{n_{i'}'}-\mathrm{ve}_{n_{i'}'} \tag{21}$$

集群实施关键线程中的数据迁移合并模型电压的改变量 vc′ 为

$$\mathrm{vc}'=\alpha(R(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}'_{i+1})+\phi\mathrm{ve}_{i+1}-\mathrm{ve}_i)\cdot\sum_{s=0}^{i}(\phi\varepsilon)^{i-s}\frac{\partial}{\partial\mathrm{vc}}\mathrm{ve}_s \tag{22}$$

其中，$R(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}_{i+1})$ 表示集群的能效比。当满足上述条件时，非关键节点 $n_{i'}'$ 的电压以 vc′ 为步长逐渐降低，直至达到条件改变的临界点，由此可得集群实施关键线程中的数据迁移合并模型后非关键节点电压的最低值。

4 数据迁移合并节能策略

本节主要介绍基于 Storm 平台的数据迁移合并节能策略，该策略在不影响集群性能的前提下，根据非关键节点非关键线程上数据的迁移情况，对非关键节点的电压进行调节，以此达到节能的目的。节能策略主要分为以下 5 个步骤，其执行流程如图 3 所示。

步骤 1 通过监控器获得原系统拓扑执行关键路径的基本信息。

步骤 2 判断是否存在关键节点非关键线程。

步骤 3 计算被迁入数据的线程是否满足 tr。

步骤 4 计算非关键节点电压的最低值。

步骤 5 根据不同的节能算法计算集群总能耗。

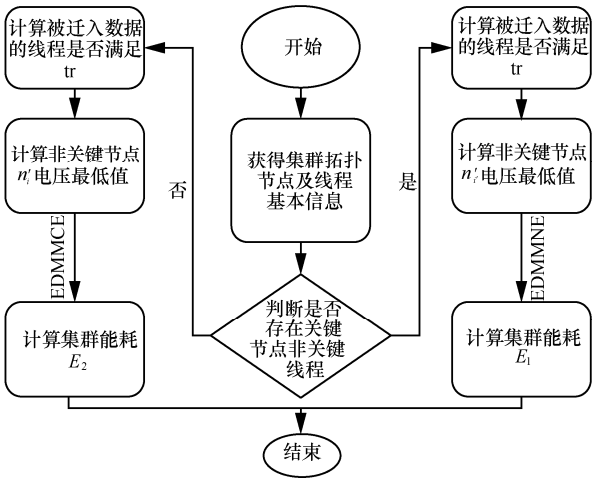


图 3 节能策略流程

4.1 资源约束算法

根据 3.3 节可知，线程被迁入数据首先应该考虑工作节点是否会出现资源溢出现象，因此需要对工作节点的资源总量进行评估。

当关键节点的某类资源占用率达到极限时，将该节点设置为极限节点，表示不能再向该节点内线程迁入数据，需要重新选择关键节点，且被选节点必须满足数据迁入 3 条原则。针对线程被迁入数据存在节点资源约束的问题，提出算法 1，确定非关键节点非关键线程上的数据可以顺利迁出。具体的

算法过程如下所示。

算法 1　资源约束算法

输入

$\mathrm{RE}_N \leftarrow (\mathrm{RE}_N^{\mathrm{C}}, \mathrm{RE}_N^{\mathrm{M}}, \mathrm{RE}_N^{\mathrm{B}})$; /*集群内工作节点 3 类资源的极限集合*/

$\mathrm{re}_{n_i} \leftarrow (\mathrm{ore}_{n_i}^{\mathrm{C}}, \mathrm{ore}_{n_i}^{\mathrm{M}}, \mathrm{ore}_{n_i}^{\mathrm{B}})$; /*关键节点 3 类资源的初始值为 $\mathrm{ore}_{n_i}^{\mathrm{C}}$, $\mathrm{ore}_{n_i}^{\mathrm{M}}$, $\mathrm{ore}_{n_i}^{\mathrm{B}}$ */

$\mathrm{re}'_{n_i} \leftarrow (\mathrm{re}_{n_i}^{\mathrm{C}'}, \mathrm{re}_{n_i}^{\mathrm{M}'}, \mathrm{re}_{n_i}^{\mathrm{B}'})$; /*线程 e_{ji} 被迁入数据后工作节点增加的 3 类资源占用为 $\mathrm{re}_{n_i}^{\mathrm{C}'}$, $\mathrm{re}_{n_i}^{\mathrm{M}'}$, $\mathrm{re}_{n_i}^{\mathrm{B}'}$ */

输出

tr/*表示关键节点上线程允许迁入数据*/

初始化

$E(C)=\{e_{j1}, e_{j2}, \cdots, e_{jn}\}, \forall e_{ji} \in E(C)$; /*集群线程集合，且任意关键节点上线程 e_{ji} 都属于 $E(C)$ */

$N''(C)=\{n_1'', n_2'', \cdots, n_m''\}$; /*集群极限节点集合*/

1) if $\mathrm{ore}_{n_i}^{\mathrm{C}} \geqslant \mathrm{re}_{n_i}^{\mathrm{C}}$ or $\mathrm{ore}_{n_i}^{\mathrm{M}} \geqslant \mathrm{re}_{n_i}^{\mathrm{M}}$ or $\mathrm{ore}_{n_i}^{\mathrm{B}} \geqslant \mathrm{re}_{n_i}^{\mathrm{B}}$ then

2)　$N''(C)=n_i$;

/*节点 n_i 为极限节点，线程不能迁入数据*/

3) else

4)　while e_{ji} = tr do

5)　if $\mathrm{re}_{n_i}^{\mathrm{C}'} + \mathrm{ore}_{n_i}^{\mathrm{C}} \leqslant \mathrm{re}_{n_i}^{\mathrm{C}}$ then

6)　n_i.put($\mathrm{re}_{n_i}^{\mathrm{C}'}$,"CPU");

/*节点 n_i 满足 CPU 资源临界原则*/

7)　end if

8)　if $\mathrm{re}_{n_i}^{\mathrm{M}'} + \mathrm{ore}_{n_i}^{\mathrm{M}} \leqslant \mathrm{re}_{n_i}^{\mathrm{M}}$ then

9)　n_i.put($\mathrm{re}_{n_i}^{\mathrm{M}'}$,"DRAM");

/*节点 n_i 满足内存资源临界原则*/

10)　end if

11)　if $\mathrm{re}_{n_i}^{\mathrm{B}'} + \mathrm{ore}_{n_i}^{\mathrm{B}} \leqslant \mathrm{re}_{n_i}^{\mathrm{B}}$ then

12)　n_i.put($\mathrm{re}_{n_i}^{\mathrm{B}'}$,"Network Bandwidth");

/*节点 n_i 满足网络带宽临界原则*/

13)　end if

14)　end while

15) end if

算法 1 为工作节点的资源约束模型。当工作节点满足 tr 时，允许线程迁入数据。步骤 1)和步骤 2)表示判断节点 n_i 是否为极限节点，若为极限节点则该节点上线程不能被迁入数据，否则需要判断工作节点数据迁入是否满足 3 条原则；步骤 5)~步骤 7)表示判断关键节点是否满足 CPU 资源临界原则；步骤 8)~步骤 10)表示在满足 CPU 资源临界原则后，判断关键节点是否满足内存资源临界原则；步骤 11)~步骤 13)表示在满足之前的两条原则后，判断关键节点是否满足网络带宽资源临近原则。

Storm 集群节能策略首先需要考虑算法的时间复杂度，原集群数据的处理及传输为轮询调度算法，其时间复杂度为 $O(n)$。首先，算法 1 需要判断节点是否为极限节点，其时间复杂度为 $O(1)$；其次，算法 1 的本质为依次判断数据迁入节点是否满足 3 条原则，其时间复杂度为 $3O(1)$；最后，由于需要循环查找满足 3 条原则的关键节点，类似于轮询调度算法，因此时间复杂度为 $O(n)$。则算法 1 的时间复杂度 $T(A)$为

$$T(A) = O(1) + 3O(1) + O(n) = O(n) \tag{23}$$

4.2　数据迁移合并算法

根据 3.4 节可知，需要通过最优线程数据重组原则判断数据迁移是否对集群性能造成影响。

在满足算法 1 的前提下，当数据迁入线程时，首先需要判断对 Storm 集群的性能是否构成影响，因此线程的选择非常重要。为尽可能减少对集群性能的影响，应该考虑以下 2 个条件：1)由于非同源线程之间进行数据的迁移合并可能会出现数据不匹配的问题，因此应该选择非关键节点非关键线程的同源线程；2)数据迁入线程首先需要考虑对集群性能的影响，因此应尽可能查找关键节点非关键线程作为数据迁入的线程。针对数据迁入线程的选择问题，提出算法 2，确定线程数据迁移的最优方法。此外，算法 2 根据是否存在关键节点非关键线程分为关键线程中的数据迁移合并算法（DMMCE, data migration and merging among critical executor）与非关键线程中的数据迁移合并算法（DMMNE, data migration and merging among non-critical executor），具体的算法过程如下所示。

算法 2　数据迁移合并算法

输入

$\mathrm{DA}_n=\{\mathrm{da}_{j1}, \mathrm{da}_{j2}, \cdots, \mathrm{da}_{jn}\}$; /*节点分配到线程的数据集合*/

$T=t$; /*集群数据处理总时间*/

$T'=t'$; /*数据迁入关键节点非关键线程后，集群数据处理总时间*/

$R=\dfrac{\mathrm{C}}{tE}$; /*原集群能效比*/

$R'=\dfrac{\mathrm{C}}{t'E'}$; /*拓扑执行 DMMCE 算法后集群能效比*/

输出

数据迁入关键节点上的线程

初始化

$E'(C)=\{e'_{ab},e'_{mn},\cdots,e'_{yz}\},\forall e_{ji}\in E(C)$; /* 对于任意的非关键节点非关键线程 e_{ji}，都存在同源线程*/

1) if 满足算法 1 允许数据迁入关键节点 then

2) if $e'_{ab}=e_{ji}$ then

/*判断线程 e_{ji} 是否存在同源的关键节点非关键线程*/

3) while $T=T'$ do

4) $e'_{ab}=\mathrm{da}_{ji_{\mathrm{input}}}$

/*数据迁入关键节点非关键线程*/

5) if $T<T'$ then

6) $e'_{ef}=e_{ji}$;

/*查找下一个同源的关键节点非关键线程*/

7) end if

8) if $n_i=N''(C)$ then

/*关键节点 n_i 为极限节点*/

9) $n_j=n_i$;

/*查找下一个关键节点*/

10) end if

11) end while

12) else

13) while $R\leqslant R'$ do

14) $e'_{mn}=\mathrm{da}_{ji_{\mathrm{input}}}$;

/*数据迁入关键节点关键线程*/

15) if $R'<R$ then

16) $e'_{de}=e_{ji}$

/*查找下一个同源的关键节点关键线程*/

17) end if

18) if $n_j=N''(C)$ then

/*关键节点 n_j 为极限节点*/

19) $n_k=n_j$

/*查找下一个关键节点*/

20) end if

21) end while

22) Zookeeper.update("configuration file");

/*更新 Zookeeper 的配置文件*/

23) end if

算法 2 选择数据迁入的线程。根据最优线程数据重组原则，选择最优迁入数据的线程。步骤 1)判断集群是否允许进行数据迁移；步骤 13)~步骤 21)根据集群性能不变的前提下，完成数据迁移；步骤 18)~步骤 27)根据集群能效比的变化，完成数据迁移；步骤 22)更新 Zookeeper 的配置文件，防止因拓扑路径发生改变而对集群数据传输与处理造成影响。

集群应在限制时间开销的前提下执行数据迁移合并算法。首先，算法 2 判断非关键节点非关键线程是否存在同源的关键节点非关键线程，其时间复杂度为 $O(1)$；其次，若存在同源的关键节点非关键线程，则需要将数据迁移到该线程，若关键路径发生改变，则循环查找下一个同源关键节点非关键线程，其时间复杂度为 $O(nT')$；最后，若不存在同源的关键节点非关键线程，则需要查找同源的关键节点关键线程，并将数据迁移到该线程，在集群能效比低于原集群后，则循环查找下一个同源关键节点关键线程，其时间复杂度为 $O(nR')$。则算法 2 的时间复杂度 $T(B)$ 为

$$T(B)=O(1)+O(nT')+O(nR') \tag{24}$$

其中，T' 表示拓扑执行 DMMNE 算法后集群数据处理及传输总时间，R' 表示拓扑执行 DMMCE 算法后集群的能效比。

4.3 节点降压算法

根据 3.4 节与 3.5 节可知，集群执行节点降压算法需要计算非关键节点电压的最低值，因此需要节点降压原则。

在满足算法 1 与算法 2 的前提下，为计算集群执行节点降压算法后非关键节点电压的最低值，需要针对不同的节点降压制约条件进行分析。其制约条件主要分为以下两点：1)在非关键节点降压过程中，集群拓扑关键路径不发生改变；2)需要针对不同的模型确定每次节点电压的改变量。针对非关键节点的降压原则，提出算法 3。同理，算法 3 根据是否存在关键节点非关键线程分为关键线程中的数据迁移合并节能算法（EDMMCE, energy-efficient algorithm for data migration and merging among critical executor）与非关键线程中的数据迁移合并节能算法（EDMMNE, energy-efficient algorithm for data migration and merging among non-critical executor），

具体的算法过程如下所示。

算法 3　节点降压算法

输入

$N'_n=\{n'_1,n'_2,\cdots,n'_m\}$; /*集群内非关键节点集合*/

$N'_i=\mathrm{ve}_i$; /*非关键节点电压的输出值*/

$t=W^{\mathrm{executor}}_{\mathrm{computation}}W^{\mathrm{edge}}_{\mathrm{node\ cost}}$; /*集群数据处理及传输的总时间*/

$n'_i=t(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}'_{i+1})$; /*集群数据处理及传输的总时间对节点电压的影响*/

$n'_i=R(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}'_{i+1})$; /*集群能效比对节点电压的影响*/

输出

非关键节点电压

1) if 满足 DMMNE 算法 then

2) $\mathrm{vc}=\alpha(t(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}'_{i+1})+\phi\mathrm{ve}_{i+1}-\mathrm{ve}_i)\cdot\sum_{s=0}^{i}(\phi\varepsilon)^{i-s}\frac{\partial}{\partial\mathrm{vc}}\mathrm{ve}_s$;

/*计算非关键节点电压的改变量*/

3) while $T=T'$ do

4) $n'_i=V_{n'_i}(\mathrm{ve}_i)-\mathrm{vc}$;

/*非关键节点电压以 vc 的步长降低*/

5) if $T<T'$ then

6) $n'_i=V_{n'_i}(\mathrm{ve}_i)+\mathrm{vc}$;

/*非关键节点电压以 vc 的步长上升*/

7) end if

8) end while

9) else

10) $\mathrm{vc}'=\alpha(R(\mathrm{ve}_i,\mathrm{da},\mathrm{ve}'_{i+1})+\phi\mathrm{ve}_{i+1}-\mathrm{ve}_i)\sum_{s=0}^{i}(\phi\varepsilon)^{i-s}\frac{\partial}{\partial\mathrm{vc}}\mathrm{ve}_s$;

/*计算非关键节点电压的改变量*/

11) while $R\leqslant R'$ do

12) $n'_{i'}=V_{n'_{i'}}(\mathrm{ve}_i)-\mathrm{vc}'$;

/*非关键节点电压以 vc′ 的步长降低*/

13) if $R'<R$ then

14) $n'_{i'}=V_{n'_{i'}}(\mathrm{ve}_i)+\mathrm{vc}'$;

/*非关键节点电压以 vc′ 的步长上升*/

15) end if

16) end while

17) end if

算法 3 根据不同的节能算法分别对非关键节点进行降压调节，达到节能的目的。步骤 2)~步骤 8)针对集群拓扑执行 DMMNE 算法后，并根据集群性能不变的前提下，计算非关键节点电压的最低值。步骤 10)~步骤 16)针对集群拓扑执行 DMMCE 算法后，并根据集群能效比，计算非关键节点电压的最低值。

与算法 1 与算法 2 相同，集群执行节点降压算法同样需要注意算法的时间复杂度。首先，算法 3 需要判断集群拓扑是否执行 DMMNE，其算法时间复杂度为 $O(1)$；其次，根据集群拓扑数据处理及传输时间不变的前提下，计算非关键节点电压的最低值，其算法时间复杂度为 $O(1)+O(nT')$；最后，根据集群能效比，计算非关键节点电压的最低值，其算法时间复杂度为 $O(1)+O(nR')$。则算法 3 的时间复杂度 $T(C)$ 为

$$T(C)=3O(1)+O(nT')+O(nR') \tag{25}$$

此外，根据算法评估可以看出，算法 3 包括这 2 个算法，即 EDMMNE 与 EDMMCE。且必须在满足算法 1 与算法 2 的前提下，集群拓扑才能执行这 2 个算法。因此集群拓扑执行 EDMMNE 的时间复杂度为

$$T(n_1)=O(n)+O(nT')+O(1)+O(nT')=O(n)+2O(nT')=O(nT') \tag{26}$$

集群拓扑执行 EDMMCE 的时间复杂度为

$$T(n_1)=O(n)+O(nR')+O(1)+O(nR')=O(n)+2O(nR')=O(nR') \tag{27}$$

4.4　能耗模型

Storm 集群在执行拓扑时的能耗一般分为 2 类，分别为基础能耗与动态能耗，其中基础能耗为物理机的待机能耗，一般来说，同一种类型物理机的基础能耗是一个固定常量。动态能耗表示集群执行拓扑时数据处理及传输的能耗，一般根据任务的不同，产生的能耗也不相同，因此动态能耗是一个变量。定义在一段时间 t 内基础能耗为 E_t^{basic}，动态能耗为 E_t^{dynamic}，则 Storm 集群在执行拓扑时的总能耗 E_t 为

$$E_t=E_t^{\mathrm{basic}}+E_t^{\mathrm{dynamic}} \tag{28}$$

由于 E_t^{basic} 为一个固定常量，因此在这里不作

考虑，而 E_t^{dynamic} 作为一个变量可以用经典的物理公式表示，即能耗 E_t^{dynamic} 是集群功耗与运行时间的乘积（单位为 J）。计算式为

$$E_t^{\text{dynamic}} = \int_t^{t+\Delta t} P^{\text{dynamic}} \mathrm{d}t \tag{29}$$

其中，P^{dynamic} 表示集群在执行拓扑时的功耗，定义集群未执行节能策略时的能耗为 $E_t^{\text{original dynamic}}$，执行节能策略后的能耗为 $E_t^{\text{EMM dynamic}}$，则执行节能策略后节约的能耗 $E_t^{\text{save dynamic}}$ 为

$$E_t^{\text{save dynamic}} = E_t^{\text{original dynamic}} - E_t^{\text{DMM dynamic}} \tag{30}$$

将式(30)代入式(29)，化简得到集群执行 DMM-Storm 节约的能耗为

$$E_t^{\text{DMM dynamic}} = \begin{cases} \int_t^{t+\Delta t} MP_1 \mathrm{d}t, & \text{拓扑执行EDMMNE算法} \\ \int_t^{t+\Delta t} MP_2 \mathrm{d}t, & \text{其他} \end{cases} \tag{31}$$

其中，P_1 表示集群执行 EDMMNE 后集群的功耗，P_2 表示集群执行 EDMMCE 后集群的功耗，M 表示集群数据量。根据式(30)可知，集群拓扑执行 2 种算法后节约的能耗 $E_t^{\text{save dynamic}}$ 为

$$E_t^{\text{save dynamic}} = \begin{cases} E_t^{\text{original dynamic}} - \int_t^{t+\Delta t} MP_1 \mathrm{d}t, & \text{拓扑执行EDMMNE算法} \\ E_t^{\text{original dynamic}} - \int_t^{t+\Delta t} MP_2 \mathrm{d}t, & \text{其他} \end{cases} \tag{32}$$

根据 3.2 节可知，集群数据处理及传输总时间等于集群路径总开销，因此将式(4)代入式(32)化简得

$$E_t^{\text{save dynamic}} = \begin{cases} E_t^{\text{original dynamic}} - \sum_{i=1}^{n} (W_{\text{computation cost}}^{\text{executor}} + W_{\text{node cost}}^{\text{edge}}) MP_1, \\ \text{拓扑执行EDMMNE算法} \\ E_t^{\text{original dynamic}} - \sum_{i=1}^{n} (W_{\text{computation cost}}^{\text{executor}} + W_{\text{node cost}}^{\text{edge}}) MP_2, \\ \text{其他} \end{cases} \tag{33}$$

其中，式(33)中的相关参数与式(4)、式(32)相同，表示集群拓扑执行 2 种算法后节约的总能耗。

5 实验与评价

本文实验的目的为验证数据迁移合并节能策略的有效性。其主要的评估指标包括集群的吞吐量、CPU 资源占用率、内存资源占用率、能耗与能效比等，实验选取 Intel 公司发布在 GitHub 上的基准测试[10]，最后对实验结果进行评估和分析。

5.1 实验环境

为验证 DMM-Storm 的有效性，实验需要将 Storm 集群部署在 19 台普通 PC 机上，且每台 PC 机的网卡统一为 100 Mbit/s LAN，内存统一为 8 GB。根据不同节点的运行状况，具体硬件配置如表 1 所示。

表 1 Storm 集群配置

节点	CPU	内存	网络带宽
Nimbus ZooKeeper_1 (Leader)	Intel core i7 4790 3.6 GHz Quad Core	8 GB DDR3 1 066 MHz	100 Mbit/s LAN
Supervisor_1 ~Supervisor_{16}	Intel core i7 4790 3.6 GHz Quad Core	8 GB DDR3 1 066 MHz	100 Mbit/s LAN
Zookeeper_2、 Zookeeper_3 (Follower)	Intel core i7 4790 3.6 GHz Quad Core	8 GB DDR3 1 066 MHz	100 Mbit/s LAN

其中，控制台节点 UI、关联节点 Zookeeper_1（Leader）与主控节点 Nimbus 运行在同一台物理机上。从节点 Supervisor_1~Supervisor_{16} 与关联节点 Zookeeper_2、Zookeeper_3（Follower）分别部署在 18 台不同 PC 机上。此外，随机从 16 台包含工作节点的 PC 机上选定 3 台进行 nmon 测试监控，记录 CPU 占用率、网络带宽占用率及内存占用率等信息。各节点软件配置如表 2 所示。

表 2 Storm 集群软件配置

参数	数值
OS	CentOS 6.4
Storm	1.2.2
JDK	1.7 Open JDK
Zookeeper	3.4.11
Python	2.6
MySQL	5.1.73
VMware	12.1.1

为全面测试 DMM-Storm 在各类不同资源开销下的有效性，实验选取 4 组基准测试，分别是 CPU

敏感型（CPU-Sensitive）的 WordCount、网络带宽敏感型（Network-Sensitive）的 Sol、内存敏感型（Memory-Sensitive）的 RollingSort 和 Storm 在真实场景下的应用 RollingCount。各基准测试运行时工作进程数量与当前所需的工作节点数量保持一致（即一个工作节点对应一个工作进程），具体的参数配置如表 3 所示。

表 3　　基准测试参数配置

基准测试	参数	数值
WordCount	component.spout_num	60
	component.split_bolt_num	120
	component.count_bolt_num	120
	topology.works	16
	topology.acker.executors	16
	topology.max.spout.pending	200
RollingSort	component.spout_num	60
	component.sort_bolt_num	120
	emit.frequency	10
	chunk.size	2 000 000
	message.size	100 000
Sol	topology.level	3
	message.size	2 000
	component.spout_num	60
	component.bolt_num	120
RollingCount	component.spout_num	60
	component.split_bolt_num	120
	component.rolling_count_bolt_num	120
	window.length	150
	emit.frequency	30

表 3 中，component.xxx_num 表示该基准测试组件并行度，SOL 中的 topology.level 表示拓扑的层次，需要设置为大于或等于 2 的整数，这里设置为 3，结合 component.xxx_num 的配置参数来看，表示一个 spout 组件运行着 60 个实例，2 个 bolt 组件运行着 120 个实例。此外，4 组基准测试统一设置 topology.works 为 16，表示各基准测试运行时一个工作节点内仅分配一个工作进程；统一设置 topology.acker.executors 为 16，表示保证线程间数据流的可靠传输；为防止元组传输因超时而重传，通过多次实验结果统一设置 topology.max.spout.pending 为 200；最后，统一设置每个 message.size 等于一个 tuple 的大小。

为验证 DMM-Storm 的效果，本文还与 WNDVR-Storm[42]进行了对比。该策略的核心思想为通过动态调节工作节点内存电压实现节能的目的。该节能策略为流式处理节能策略的代表，且包括关键路径节能（DVRCP, DRAM voltage regulation on critical path）与非关键路径节能（DVRNP, DRAM voltage regulation on non-critical path）2 个算法。表 4 列举了 WNDVR-Storm 在基准测试 RollingCount 上的配置参数。表 4 中的 β 与 γ 分别表示工作节点 CPU 使用率与数据传输量的阈值，表 4 内的结果都是通过对 RollingCount 进行采样获得。component. spout_num 与 component. bolt_num 分别为 60 与 120，topology.works 与 topology.acker.executors 都是 16，是为了与本文的策略保持一致，保证在同等条件下验证策略的有效性。

表 4　　WNDVR-Storm 参数配置

参数	数值
component.spout_num	60
component.split_bolt_num	120
component.rolling_count_bolt_num	120
topology.max.spout.pending	200
topology.workers	16
topology.acker.executors	16
T	30 s
β	65%
γ	50 000 tuple/s
执行 DVRCP 内存电压的取值范围	1.25~1.50 V
执行 DVRNP 内存电压的取值范围	1.36~1.50 V

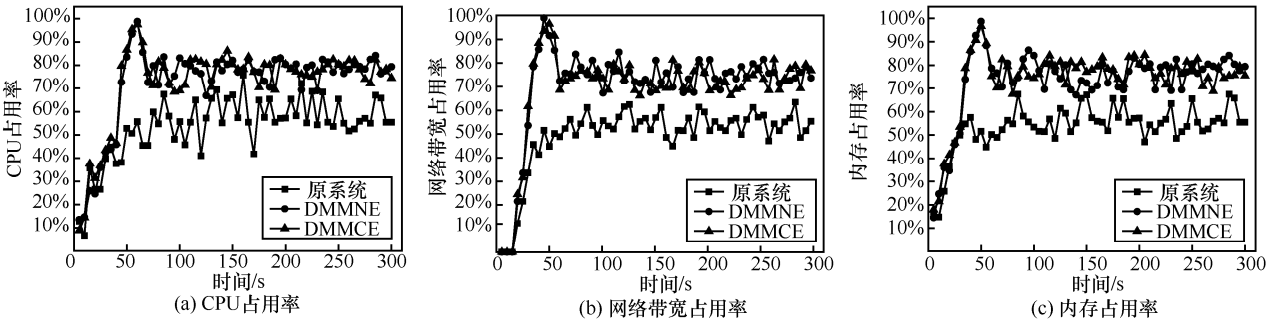
5.2　执行节能策略的电压选择

为便于观测，以下测试均设置 metrics.poll 为 5 000 ms，metrics.time 为 300 000 ms，即每组实验每 5 s 进行一次采样，时长为 5 min。

5.2.1　数据迁移测试

为了集群内数据迁移能够正常执行，现对集群关键节点各类资源的占用率进行测试，并统计集群执行数据迁移合并算法后关键节点各类资源的占用率，具体的结果如图 4 所示。

图 4 为 RollingCount 下各类资源的占用率对比，同理可得，WordCount、Sol 与 RollingSort 这 3 个基准测各类资源的占用率对比。为观察集群执行数据迁移合并算法的具体效果，表 5 统计了各类资源占用情况的平均值。



(a) CPU占用率 (b) 网络带宽占用率 (c) 内存占用率

图 4 集群执行 RollingCount 后，各类资源的占用率对比

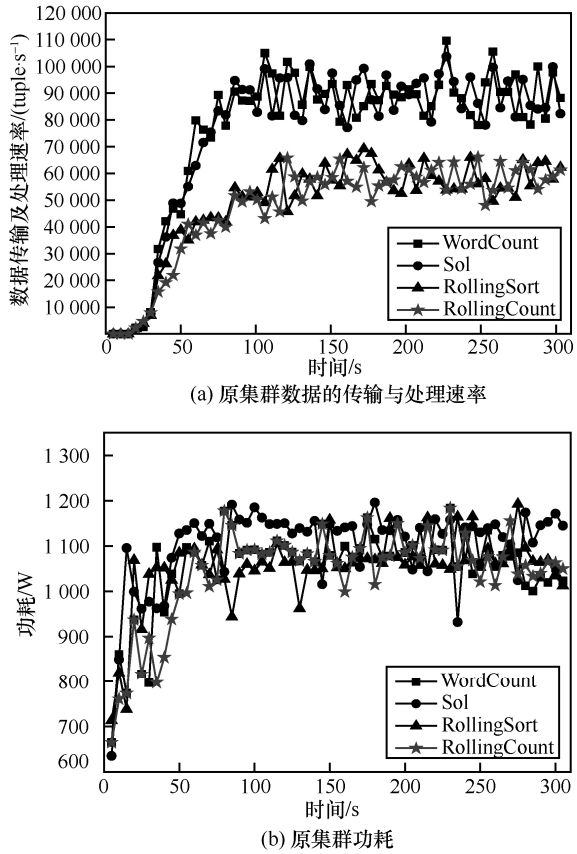
表 5 集群执行不同基准测试各类资源的占用率

基准测试	算法	CPU 占用率	网络带宽 占用率	内存 占用率
RollingCount	原系统	53.8%	49.4%	53.5%
	DMMNE	72.5%	70.5%	73.5%
	DMMCE	73.1%	69.2%	74.3%
WordCount	原系统	72.6%	71.3%	61.4%
	DMMNE	86.3%	88.3%	80.7%
	DMMCE	85.2%	89.2%	79.8%
Sol	原系统	24.5%	69.6%	51.3%
	DMMNE	60.6%	88.7%	72.8%
	DMMCE	63.4%	87.3%	71.9%
RollingSort	原系统	58.7%	51.2%	49.8%
	DMMNE	72.4%	75.4%	74.8%
	DMMCE	73.7%	76.4%	75.6%

根据图 4 与表 5 可知，集群执行数据迁移合并算法后关键节点各类资源的占用率急剧上升，数据迁移合并算法效果明显，且执行 DMMNE 与 DMMCE 的资源占用基本相同。此外，由图 4 可知，策略对集群 3 类资源的占用率的影响时长在第 45~65 s，共耗时约 20 s，资源占用率急速上升，其原因为集群拓扑执行数据迁移合并算法，数据迁移过程中，集群资源消耗巨大。65 s 后集群资源占用率趋于稳定，由此，非关键节点线程上的数据顺利迁出，并可通过节点降压原则降低非关键节点电压，从而实现节能的效果。

5.2.2 节点降压测试

根据 3.4 节可知，节点降压原则存在 2 个制约条件，即非关键节点电压的限制条件与非关键节点电压的改变量，其中执行 EDMMNE 算法，非关键节点电压的限制条件为集群性能；执行 EDMMCE 算法，非关键节点电压的限制条件为集群能效比。集群性能可通过单位时间内数据的传输及处理速率表示，集群能效比可通过单位时间内的集群功耗表示。非关键节点电压的改变量可通过式(15)与式(22)确定。则原集群 5 min 内集群吞吐量以及功耗可通过图 5 表示。



(a) 原集群数据的传输与处理速率

(b) 原集群功耗

图 5 原集群吞吐量与功耗

由图 5 可知，通过使用 DVFS[38]技术，集群执行 EDMMNE，规定单位时间内数据的传输与处理速率不发生改变，则可每次以 vc 为步长降低非关键节点电压；集群执行 EDMMCE，规定集群能效比不低于原集群，则可每次以 vc′ 为步长降低非关键节点电压。图 6 为 RollingCount 下，集群执行 2 种算法非关键节点电压的改变情况。

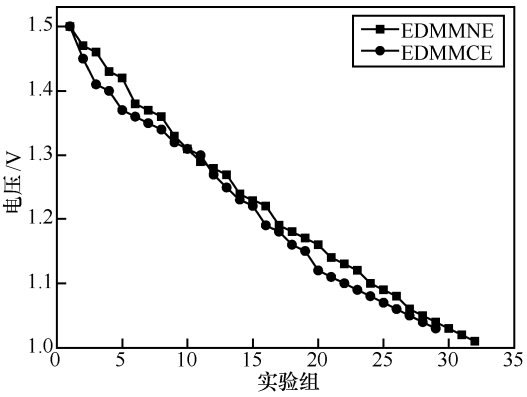


图 6　集群执行 RollingCount 后，2 种算法非关键节点电压的改变情况

由图 6 可知，集群执行 EDMMNE 非关键节点电压的最低值为 1.01 V，集群执行 EDMMCE 非关键节点电压的最低值为 1.03 V。同理可得，WordCount 后，集群执行 EDMMNE 非关键节点电压的最低值为 1.06 V；集群执行 EDMMCE 非关键节点电压的最低值为 1.05 V。Sol 后，集群执行 EDMMNE 非关键节点电压的最低值为 1.04 V；集群执行 EDMMCE 非关键节点电压的最低值为 1.05 V。RollingSort 后，集群执行 EDMMNE 非关键节点电压的最低值为 1.03 V；集群执行 EDMMCE 非关键节点电压的最低值为 1.02 V。

此外，该实验通过降低电压带来的集群拓扑数据处理及传输延迟的问题，但由于集群的关键路径不发生改变，因此在这里不做过多的叙述说明。

5.3　数据迁移合并节能策略实验结果分析

本节首先讨论在 Storm 默认的调度策略与 DMM-Storm 下，分别对 WordCount、Sol 与 RollingSort 进行功耗测试，具体实验结果如图 7 所示。

由图 7 可知，45 s 前 Storm 默认的调度策略与 DMM-Storm 的功耗基本相同，而 45~65 s，共耗时约 20 s，集群执行 DMM-Storm 的功耗急剧上升，其原因为集群在此期间执行数据迁移合并算法，集群资源占用率消耗巨大，因此集群功耗急剧上升。之后 70~90 s，共耗时约 25 s，集群执行默认的调度策略与 DMM-Storm 的功耗又逐渐接近，其原因为执行数据迁移合并算法并不改变集群的功耗。95~115 s，共耗时约 20 s，集群执行 DMM-Storm 的功耗缓慢降低，其原因为集群在此期间执行节点降压算法。120 s 之后集群功耗趋于稳定，集群执行 DMM-Storm 的功耗明显低于原系统。集群 120 s 后的功耗结果如表 6 所示。

表 6　集群稳定后的功耗统计

实验组	最小功耗/W	最大功耗/W	平均功耗/W
test11	1 000.787 46	1 183.516 94	1 083.995 012
test12	840.833 95	985.700 53	905.268 979 5
test13	820.316 05	985.104 32	911.453 409 2
test21	1 015.426 56	1 195.926 06	1 125.715 989
test22	867.373 92	936.333 72	915.091 409 7
test23	852.070 68	947.363 31	911.314 227 2
test31	962.508 52	1192.577 08	1 076.023 367
test32	856.864 18	965.197 93	926.058 879 5
test33	832.394 69	959.708 03	905.334 361 6

根据表 6 可知，集群执行 DMM-Storm 的功耗远低于原集群功耗。此外由于不同的基准测试集群的拓扑并不相同，因此为检测策略在真实环境下的节能效果，需要对 RollingCount 进行测试。

RollingCount 作为 Storm 平台下的一个大数据经典应用程序，广泛应用到各类大数据需求的实时应用场景。本实验采用 RollingCount 对集群真实环境中的功耗进行统计，并计算其实际的能耗。此外，引入 WNDVR-Storm 与 Storm 默认的调度策略以及 DMM-Storm 作对比，其中 WNDVR-Storm 的配置参数与表 4 相同。图 8 为 RollingCount 在不同节能

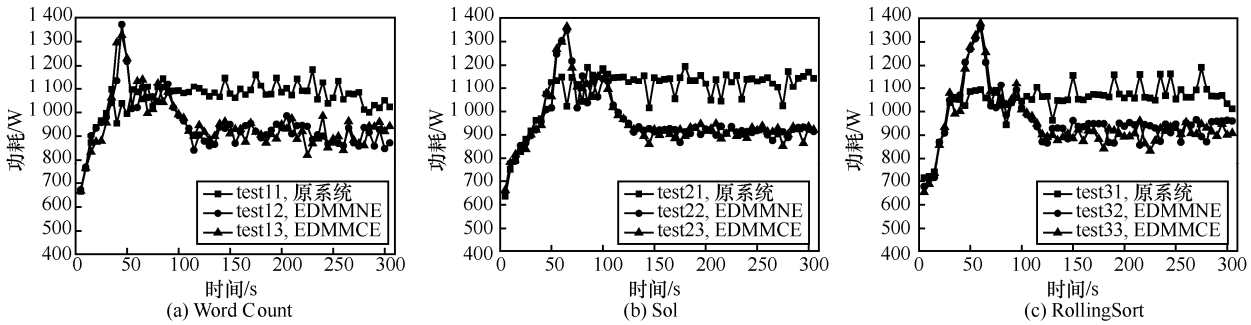


图 7　3 种基准测试下集群的功耗情况

策略下的功耗及能耗对比。

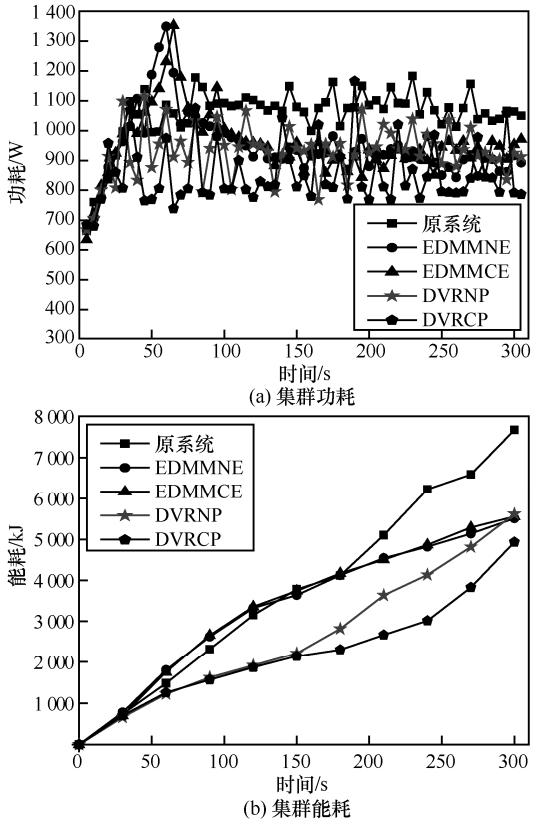


图 8 RollingCount 下集群的功耗与能耗

根据图 8(a)可知，集群功耗随着时间的上升不断增加，其中 Storm 默认的调度策略与 WNDVR-Storm 在 50 s 之后趋于稳定，而 DMM-Storm 则在 115 s 后趋于稳定，其原因为在 40~115 s，共耗时 75 s，集群执行数据迁移合并算法与节点降压算法。集群功耗趋于稳定后，DMM-Storm 与 WNDVR-Storm 的功耗都远低于 Storm 默认的调度策略，执行 EDMMNE 的平均功耗为 915.76 W，执行 EDMMCE 的平均功耗为 922.81 W，执行 DVRNP 的平均功耗为 928.13 W，执行 DVRCP 的平均功耗为 851.43 W，Storm 默认的调度策略的平均功耗为 1 054.58 W。此外虽然执行 WNDVR-Storm 的平均功耗低于 DMM-Storm，但是执行 WNDVR-Storm 的功耗波动较大，非常不稳定，且策略实现较为困难，不适合在大规模集群范围内使用。根据图 8(a)计算集群能耗获得图 8(b)，由图 8(b)可知，30 s 之前集群的能耗相同，其原因为 2 个节能策略中的算法并未触发。集群执行 DMM-Storm 在 190 s 之前低于 Storm 默认的调度策略，其原因为 DMM-Storm 触发了数据迁移合并算法与节点降压算法致使集群能耗增加。此外根据图 8(b)可知，120 s 之后 DMM-Storm 的功耗趋于稳定，而 WNDVR-Storm 上升幅度不断变化且逐渐变高，其原因为随着集群数据量的不断增大，数据量始终超过额定阈值，导致 WNDVR-Storm 基本失效。集群执行 EDMMNE 与 EDMMCE 分别与原集群能耗作对比，执行 EDMMNE 将原系统的能耗降低了 28.1%，执行 EDMMCE 将原系统的能耗降低了 27.6%。综上所述，DMM-Storm 取得了较为理想的节能效果。

5.4 实验规模局限性与有效性评估

由于缺少更多的物理节点，实验选择通过虚拟机建立更多的虚拟节点对集群规模的局限性进行评估，实验分别建立 32、48、64、80 个工作节点与原集群 16 个工作节点进行对比，检验大规模集群下节能策略的有效性，具体结果如图 9 所示。

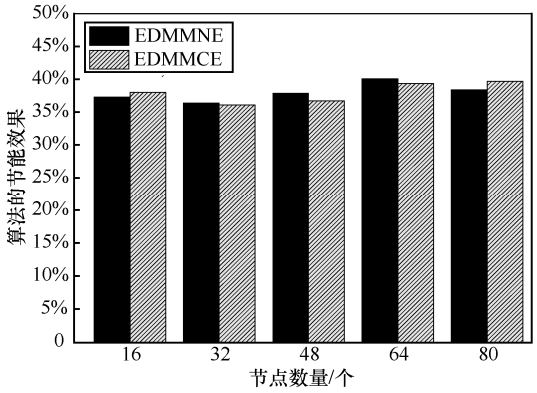


图 9 扩大节点规模后 2 种算法的效果

由图 9 可知，随着节点数的增加，2 种算法节能效果并未受到影响。由于随着节点数总量的增加，非关键节点的数量也在不断增加，2 种算法都是降低非关键节点电压而达到节能的效果，因此并不受集群规模的影响。但策略并未考虑集群规模对性能的影响，现根据节点数的增加计算集群性能，具体的实验结果如图 10 所示。

由图 10 可知，随着集群规模的扩大，执行 EDMMNE 的性能基本不受影响，而执行 EDMMCE 集群性能随节点数的增加不断下降，且集群性能的下降符合线性关系，因此，根据图 10 可得出 EDMMCE 的理论函数为

$$v_t^{\mathrm{M}} = \mathrm{a}N(C) + \mathrm{b} \tag{34}$$

其中，v_t^{M} 表示集群数据处理及传输速率，$N(C)$表示

集群节点数量，a 与 b 为常数，将图 10 的数据代入式(34)，化简可得

$$v_t^{\mathrm{M}} = -256.350\,3N(C) + 101\,751.866\,7 \qquad (35)$$

由式(35)可知，执行 EDMMCE 因节点数增加而对集群性能造成一定的影响，因此 DMM-Storm 存在一定的局限性。

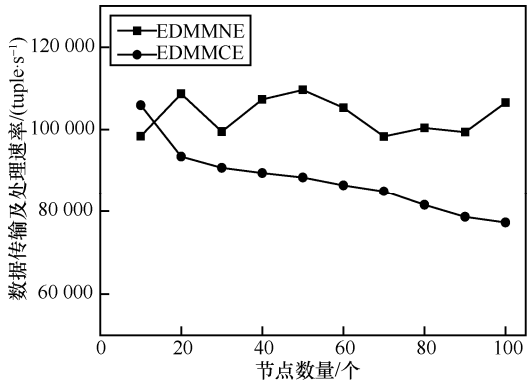


图 10　扩大节点规模后 2 种算法对集群性能的影响

流式大数据处理的节能策略首先应该关注其对集群性能的影响，根据图 10 可知，执行 EDMMNE 对集群的性能基本不造成影响，在此不作考虑，而执行 EDMMCE 对集群的性能造成一定的影响，需要对该模型进行评估，在此引入能效比的概念，具体结果如图 11 所示。

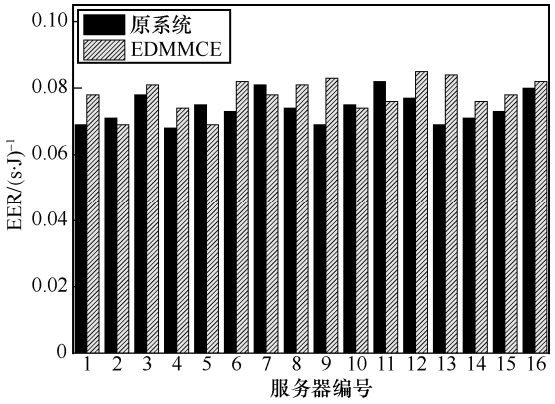


图 11　能效比

由图 11 可知，执行 EDMMNE 后的能效比与原集群的能效比基本相等，则执行 EDMMNE 对集群的性能并不构成影响，但由于式(17)内的常数 C 存在误差，且不同节点间的能耗不相等，因此测量结果并不相同。进过反复实验测量，误差常数 C 的取值范围为[0.81, 0.96]，则原集群平均能效比为 $0.0741\,(\mathrm{s}\cdot\mathrm{J})^{-1}$，执行 EDMMNE 后的集群能效比为 $0.0781\,(\mathrm{s}\cdot\mathrm{J})^{-1}$。综上所述，EDMMNE 是完全可行的。

6　结束语

目前，大数据技术飞速发展，各种分布式平台应运而生，其中流式大数据平台因其强大的实时性深受学术界与产业界的关注，而 Storm 平台作为最重要的流式大数据平台之一，在业界具有强大的影响力。但是由于 Storm 平台在最初设计时并未考虑能耗的问题，导致其在执行任务时产生的高能耗问题亟待解决。针对这一问题，本文通过分析 Storm 平台的基本框架，建立了数据分配、路径开销与资源约束 3 个基本模型，并在此基础上提出了资源约束算法、数据迁移合并算法与节点降压算法。此外，根据节点降压算法，确定了非关键节点电压的最小值，以此降低了非关键节点电压，达到了节能的效果。最后通过 4 个基准测试验证了策略的有效性。

下一步的研究工作主要包括以下三方面：1) 提高 Storm 平台的计算能力，增强其任务执行的效率，做到提高性能的同时节约能耗，如利用图形处理器（GPU, graphics processing unit）提高集群的计算能力；2)减少集群节点间的通信，增加集群线程间与进程间的通信，通过减少其部分通信开销，缩短任务的执行时间，以达到节能的效果；3) 减少部分电子元件的内部时延，提高电子元件的高效性，进而提升集群的整体性能，以至从侧面降低集群能耗。

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): 146-169.
MENG X F, CI X. Big data management: concepts, techniques and challenges[J]. Journal of Computer Research and Development, 2013, 50(1): 146-169.

[2] RANJAN R. Streaming big data processing in datacenter clouds[J]. IEEE Cloud Computing, 2014, 1(1): 78-83.

[3] CHEN C L P, ZHANG C Y. Data-intensive applications, challenges, techniques and technologies: a survey on big data[J]. Information Sciences, 2014, 275(11): 314-347.

[4] 孙大为. 大数据流式计算: 应用特征和技术挑战[J]. 大数据, 2015, 1(3): 99-105.
SUN D W. Big data stream computing: features and challenges[J]. Big Data Research, 2015, 1(3): 99-105.

[5] KAMBATLA K, KOLLIAS G, KUMAR V, et al. Trends in big data analytics[J]. Journal of Parallel and Distributed Computing, 2014, 74(7): 2561-2573.

[6] 杨挺, 王萌, 张亚健, 等. 云计算数据中心 HDFS 差异性存储节能优化算法[J]. 计算机学报, 2019(4): 721-735.
YANG T, WANG M, ZHANG Y J, et al. HDFS differential storage energy-saving optimal algorithm in cloud data center[J]. Chinese Journal of Computers, 2019(4): 721-735.

[7] 余晓晖. 数据中心能效测评指南[R]. “云计算发展与政策论坛”技术报告, (2012-03-16)[2019-07-04].
YU X H. Data center energy efficiency assessment guide[R]. Cloud Computing Development and Policy Forum Technical Report, (2012-03-16) [2019-07-04].

[8] 陈小燕, 干丽萍, 郭文平. 大数据可视化工具比较及应用[J]. 计算机教育, 2018, 282(6): 100-105.
CHEN X Y, GAN L P, GUO W P. Comparison and application of big data visualization tools[J]. Computer Education, 2018, 282(6): 100-105.

[9] SUN D, ZHANG G, YANG S, et al. Re-Stream: real-time and energy-efficient resource scheduling in big data stream computing environments[J]. Information Sciences, 2015, 319: 92-112.

[10] 鲁亮, 于炯, 卞琛, 等. 大数据流式计算框架 Storm 的任务迁移策略[J]. 计算机研究与发展, 2018, 55(1): 71-92.
LU L, YU J, BIAN C, et al. A task migration strategy in big data stream computing with Storm[J]. Journal of Computer Research and Development, 2018, 55(1): 71-92.

[11] BORTHAKUR D, GRAY J, SARMA J S, et al. Apache Hadoop goes realtime at Facebook[C]//The 2011 ACM SIGMOD International Conference on Management of Data. ACM, 2011: 1071-1080.

[12] NEUMEYER L, ROBBINS B, NAIR A, KESARI A. S4: distributed stream computing platform[C]//The 10th IEEE International Conference on Data Mining Workshops (ICDMW 2010). IEEE, 2010: 170-177.

[13] 李梓杨, 于炯, 卞琛, 等. 基于流网络的 Flink 平台弹性资源调度策略[J]. 通信学报, 2019, 40(8): 85-101.
LI Z Y, YU J, BIAN C, et al. Flow-network based auto rescale strategy for Flink[J]. Journal on Communications, 2019, 40(8): 85-101.

[14] 卞琛, 于炯, 修位蓉, 等. 基于分配适应度的 Spark 渐进填充分区映射算法[J]. 通信学报, 2017, 38(9): 133-147.
BIAN C, YU J, XIU W R, et al. Progressive filling partitioning and mapping algorithm for Spark based on allocation fitness degree[J]. Journal on Communications, 2017, 38(9): 133-147.

[15] KULKARNI S, BHAGAT N, FU M, et al. Twitter heron: Stream processing at scale[C]//The 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015: 239-250.

[16] ANDERSON Q. Storm real-time processing cookbook[M]. Birmingham: Packt Publishing, 2013: 4-8.

[17] TA V D, LIU C M, NKABINDE G W. Big data stream computing in healthcare real-time analytics[C]//The 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA). IEEE, 2016: 37-42.

[18] MISHNE G, DALTON J, LI Z, et al. Fast data in the era of big data: Twitter’s real-time related query suggestion architecture[C]//The 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013: 1147-1158.

[19] DING W L, HAN Y B, ZHAO Z F, et al. Stream-oriented availability services for endpoint-to-endpoint data transmission[C]//The 2012 International Conference on Cloud and Service Computing. IEEE, 2012: 212-218.

[20] SHIN D J, PARK S K, KIM S M, et al. Adaptive page grouping for energy efficiency in hybrid PRAM-DRAM main memory[C]//ACM Research in Applied Computation Symposium. ACM, 2012: 395-402.

[21] BONAMY R, BILAVARN S, MULLER F. An energy-aware scheduler for dynamically reconfigurable multi-core systems[C]//International Symposium on Reconfigurable Communication-Centric Systems-On-Chip. IEEE, 2015: 1-6.

[22] KIM H S, SHIN D I, YU Y J, et al. Towards energy proportional cloud for data processing frameworks[M]. San Jose: USENIX Association, 2010: 1-8.

[23] FAISAL S M, TZIANTZIOULIS G, GOK A M, et al. Edge importance identification for energy efficient graph processing[C]//IEEE International Conference on Big Data. IEEE, 2015: 347-354.

[24] SONG J, MA Z, THOMAS R, et al. Energy efficiency optimization in big data processing platform by improving resources utilization[J]. Sustainable Computing: Informatics and Systems, 2019, 21: 80-89.

[25] MU J, PEI Y, LI W, et al. Research on energy saving optimization strategy of substation operation based on big data technology[C]//2018 Chinese Control And Decision Conference (CCDC). IEEE, 2018: 3567-3571.

[26] DE MATTEIS T, MENCAGLI G. Keep calm and react with foresight: strategies for low-latency and energy-efficient elastic data stream processing[J]. Journal of Systems and Software, 2016, 51(8): 1-12.

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

[28] LANG W, PATEL J M. Energy management for MapReduce clusters[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2): 129-139.

[29] 宋杰, 李甜甜, 朱志良, 等. 云数据管理系统能耗基准测试与分析[J]. 计算机学报, 2013, 36(7): 1485-1499.
SONG J, LI T T, ZHU Z L, et al. Benchmarking and analyzing the energy consumption of cloud data management system[J]. Chinese Journal of Computers, 2013, 36(7): 1485-1499.

[30] 廖彬, 张陶, 于炯, 等. MapReduce 能耗建模及优化分析[J]. 计算机研究与发展, 2016, 53(9): 2107-2131.
LIAO B, ZHANG T, YU J, et al. Energy consumption modeling and optimization analysis for MapReduce[J]. Journal of Computer Research and Development, 2016, 53(9): 2107-2131.

[31] LIAO B, YU J, ZHANG T, et al. Energy-efficient algorithms for distributed storage system based on block storage structure reconfiguration[J]. Journal of Computer Research & Development, 2015, 48(2): 71-86.

[32] SHIN D J, PARK S K, KIM S M, et al. Adaptive page grouping for energy efficiency in hybrid PRAM-DRAM main memory[C]//ACM Research in Applied Computation Symposium. ACM, 2012: 395-402.

[33] ZHOU S, CHELMIS C, PRASANNA V K. High-Throughput and Energy-Efficient Graph Processing on FPGA[C]//International Symposium on Field-Programmable Custom Computing Machines. IEEE, 2016: 103-110.

[34] 廖彬, 张陶, 于炯, 等. 温度感知的 MapReduce 节能任务调度策略[J]. 通信学报, 2016, 37(1): 61-75.
LIAO B, ZHANG T, YU J. Temperature aware energy-efficient task

scheduling strategies for MapReduce[J]. Journal on Communications, 2016, 37(1): 61-75.

[35] VASUDEVAN V, FRANKLIN J, ANDERSEN D. FAWN damentally power-efficient clusters[C]//The 12th Workshop on Hot Topics in Operating Systems (HotOS 09). Usenix Association, 2009: 1-5.

[36] 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.
LIAO B, YU J, SUN H, et al. Energy-efficient algorithms for distributed storage system based on data storage structure reconfiguration[J]. Journal of Computer Research and Development, 2013, 50(1): 3-18.

[37] GUO B, YU J, LIAO B, et al. A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing[J]. Journal of Network and Computer Applications, 2017, 84: 118-130.

[38] WANG Z, WANG H, ZHAO W, et al. Energy optimization of parallel programs in a heterogeneous system by combining processor core-shutdown and dynamic voltage scaling[J]. Future Generation Computer Systems, 2019, 92: 198-209.

[39] CORDESCHI N, SHOJAFAR M, AMENDOLA D, et al. Energy-efficient adaptive networked datacenters for the QoS support of real-time applications[J]. The Journal of Supercomputing, 2014, 71(2): 448-478.

[40] PANDA A, CHATHA K S. An embedded architecture for energy-efficient stream computing[J]. IEEE Embedded Systems Letters, 2014, 6(3): 57-60.

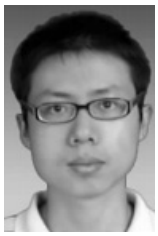
[41] ZONG Z, MANZANARES A, RUAN X, et al. EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters[J]. IEEE Transactions on Computers, 2010, 60(3): 360-374.

[42] 蒲勇霖, 于炯, 鲁亮, 等. Storm 平台下工作节点的内存电压调控节能策略[J]. 通信学报, 2018, 39(10): 101-121.
PU Y L, YU J, LU L, et al. Energy-efficient strategy for work node by DRAM voltage regulation in Storm[J]. Journal on Communications, 2018, 39(10): 101-121.

[作者简介]

蒲勇霖（1991− ），男，山东淄博人，新疆大学博士生，主要研究方向为流式计算、绿色计算、内存计算等。

于炯（1964− ），男，新疆乌鲁木齐人，博士，新疆大学教授、博士生导师，主要研究方向为并行计算、分布式系统、绿色计算等。



鲁亮（1990− ），男，天津人，博士，中国民航大学讲师，主要研究方向为分布式系统、内存计算、绿色计算。

李梓杨（1993− ），男，新疆乌鲁木齐人，新疆大学博士生，主要研究方向为流式计算、内存计算等。

卞琛（1981− ），男，江苏南京人，博士，广东金融学院副教授，主要研究方向为分布式系统、内存计算、绿色计算等。

廖彬（1986− ），男，新疆乌鲁木齐人，博士，新疆财经大学副教授、硕士生导师，主要研究方向为分布式系统、数据库理论与技术、绿色计算等。