# 流式大数据平台下的弹性数据迁移能效优化策略

蒲勇霖[1]，许小龙[1]，于炯[2]，李梓杨[2]，国冰磊[3]
（1. 南京信息工程大学软件学院，江苏 南京 210044；2. 新疆大学软件学院，新疆 乌鲁木齐 830002；
3. 湖北文理学院计算机工程学院，湖北 襄阳 441053）

**摘 要：**针对流式计算框架在最初设计时缺乏能效方面的考虑，导致其存在高能耗与低效率的问题，提出一种流式大数据平台下的弹性数据迁移节能优化策略。首先，建立负载预测模型与资源判定模型，并进一步设计负载预测算法，通过预测负载变化趋势确定节点资源占用，找到资源过载与过剩节点；其次，建立资源约束模型与最优数据迁移模型，由此提出最优数据迁移算法，以提高节点资源利用率为目的进行数据迁移；最后，建立能耗模型，计算集群进行数据迁移后节约的能耗。实验结果表明，数据迁移节能优化策略能够对集群内节点资源变化做出及时响应，并在提高节点资源利用率的基础上，有效提高集群数据处理的能效。

**关键词：**流式计算；负载预测；资源约束；数据迁移；能效
**中图分类号：**TN311
**文献标志码：**A
**DOI:** 10.11959/j.issn.1000−436x.2024006

# Energy-efficient optimization strategy based on elastic data migration in big data streaming platform

PU Yonglin[1], XU Xiaolong[1], YU Jiong[2], LI Ziyang[2], GUO Binglei[3]
1. School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
2. School of Software, Xinjiang University, Urumqi 830002, China
3. School of Computer Engineering, Hubei University of Arts and Science, Xiangyang 441053, China

**Abstract:** Focused on the problem that the stream computing platform was suffering from the high energy consumption and low efficiency due to the lack of consideration for energy efficiency in designing process, an energy-efficient optimization strategy based on elastic data migration in big data streaming platform (EEDM-BDSP) was proposed. Firstly, models of the load prediction and the resource judgment were set up, and the load prediction algorithm was designed, which predicted the load tendency and determine node resource occupancy, so as to find nodes of resource overload and redundancy. Secondly, models of the resource constraint and the optimal data migration were set up, and the optimal data migration algorithm was proposed, which data migration for the purpose of improving node resource utilization. Finally, model of the energy consumption was set up to calculate the energy consumption saved by the cluster after data migration. The experimental results show that the EEDM-BDSP changes node resources in the cluster can responded on time, the resource utilization and the energy-efficient are improved.

**Keywords:** stream computing, load prediction, resource constraint, data migration, energy-efficient

**收稿日期：**2023−07−18；**修回日期：**2023−11−08
**通信作者：**国冰磊，binglei@hbuas.edu.cn
**基金项目：**国家自然科学基金资助项目（No.62262064）；新疆维吾尔自治区重点研发计划基金资助项目（No.2022295358）；江苏省高等学校自然科学基金资助项目（No.23KJB520019）；新疆维吾尔自治区自然科学基金资助项目（No.2022D01C56）；湖北省自然科学基金资助项目（No.2022CFB805）
**Foundation Items:** The National Natural Science Foundation of China (No.62262064), The Key Research and Development Project of Xinjiang Uygur Autonomous Region (No.2022295358), The Natural Science Foundation of Jiangsu Higher Education Institutions (No.23KJB520019), The Natural Science Foundation of Xinjiang Uygur Autonomous Region (No.2022D01C56), The Natural Science Foundation of Hubei (No.2022CFB805)

## 0　引言

近年来，随着各种高速互联技术的发展，物联网场景下产生的数据呈指数级增长，并与互联网共同成为各大行业数据的主要来源。但是各种数据增长所伴随的能耗问题也呈爆炸性增加。预计到 2035 年，全国数据中心能耗将达到4 505亿~4 855亿千瓦时，届时全国数据中心的碳排放量将超过亿吨量级[1]。此外，全球电子可持续发展倡议组织（GeSI, global e-sustainability initiative）在《节能化 2020 年：在信息时代推动低碳经济》报告中指出，2020 年全球二氧化碳排放量达到 519 亿吨（其中，信息通信技术领域产生 14 亿吨）[2]。从这些数据中可以看出，IT 行业的污染与能耗已经不容忽视[3]。因此，加强 IT 行业的绿色发展，既符合我国的政策需求也是一个重要的机遇与挑战。

希捷（Seagate）公司与 IDC 联合发布的《数据时代 2025》白皮书中预测，2025 年全球数据量将达到 163 ZB。其中，超过 25%的数据将成为实时数据，而物联网实时数据占比将达到实时数据的 95%[4]。针对大数据处理的高性能集群一般分为批量处理框架与流式处理框架两类。其中以 MapReduce 为代表的批量处理框架表现为先存储后计算，需要与磁盘进行频繁交互，无法满足数据的实时性处理需求；流式处理框架由于其强大的实时性，为实时大数据分析提供了良好的框架层解决方案[5]。但是流式处理框架在高速处理实时数据的同时伴随着高能耗的问题，制约了平台的发展[6]，因此如何解决流式处理框架的节能计算是一个不容忽视的问题。

流式计算框架作为高实时性的大数据处理框架，深受知名 IT 企业的青睐，但其发展也面临着一系列的挑战。1) 由于其框架存在无序性与无限性的特点，无法预知数据流的走向，导致节点出现资源过载或过剩的问题，从而影响框架的性能并带来额外的能耗。2) 流式框架需要处理多变性的流式数据，而其调度策略无法根据流式作业的状态信息动态调节集群并行度，对节点资源的利用率及框架的性能造成一定的影响。

针对上述挑战，本文主要的研究工作如下。

1) 通过融合机器学习算法建立了流式框架的负载预测模型，预测集群节点内的数据流速与负载波动，确定节点数据处理的实际情况，并为资源判定模型提供支持。

2) 通过负载预测模型建立了资源判定模型，并根据节点资源的占用情况，制定以节能为目的的弹性资源调度计划，为集群内的数据迁移提供了理论支持。

3) 通过资源判定模型建立了最优数据迁移模型，根据作业内的最短路径对集群的规模进行动态调整，并进一步提出流式大数据平台下的弹性数据迁移节能优化策略（EEDM-BDSP, energy-efficient strategy based on elastic data migration in big data streaming platform）。该策略包含负载预测算法与最优数据迁移算法，在集群节点出现资源过载或过剩时，动态调节集群节点规模与作业的拓扑结构，避免出现资源占用问题影响集群数据处理性能。

4) 通过数据迁移结果建立了能耗模型，根据集群数据的处理时间计算集群的能耗，并与原集群能耗进行对比，验证节能策略的效果。

## 1　相关研究

目前，针对主流实时大数据处理框架的节能计算相对较少，导致其能耗呈指数级增长，带来的影响已制约框架的发展。针对流式大数据处理的节能计算一般分为两类，其一为硬件节能方法[7]，其二为软件节能方法[8]。

硬件节能方法主要包括替换电子元件[9]或动态调压[10]2 种思路。文献[11]通过将集群节点内的部分 CPU 替换成 GPU，以解决对图像数据的处理，提高集群数据处理的效率，降低能耗，实验结果表明该方法在节约 9.69%能耗的前提下减少了 8.63%访问时间。文献[12]通过使用高能效的并行随机存取机（PRAM, parallel random access machine）替换原集群的动态随机存储器（DRAM, dynamic random access memory），进而达到提高集群整体能效的目的，实验结果表明该方法有效降低了集群 36%~42%的能耗。文献[6]通过对流式处理集群的内存电压进行动态调节，在不影响集群性能的前提下分别节约了 28.5%与 35.1%的能耗。文献[10]通过使用动态电压缩放管理（DVFS, dynamic voltage frequency scaling），对集群节点的 CPU 电压进行动态缩放管理，有效降低了集群的整体能耗。文献[13-14]通过动态电压缩放管理或替换高能耗的电子元件，以达到节能的效果。但是由于替换高能耗的电子元件价

格高昂，而动态电压缩放管理存在较大误差且不好调整，都不适合部署在大规模的集群环境中。文献[15]针对 Storm 框架在进行数据处理时存在高能耗的问题，提出数据迁移合并节能策略，该策略使用 DVFS 技术对迁移资源后工作节点的 CPU 进行电压缩放管理，有效降低了集群的能耗与节点间的通信开销，但是存在以下两点问题：1) 对集群拓扑处理数据的实时性造成一定影响；2) 使用 DVFS 技术调节集群节点 CPU 的电压存在较大的偶然性，且技术的实现难度较高，不适合部署于大规模集群中。

软件节能方法包括资源调度[16]或建立能耗感知模型[17]2 种思路。文献[18]建立了一种能耗感知模型，通过在流式处理框架中查找一种数据处理流速与通信开销之间的平衡，确定此时集群的能耗最低。文献[16]建立了一种自适应作业调节的能耗感知模型，通过对集群内的数据处理信息进行实时捕捉，找到数据处理与集群能耗之间的平衡，实验结果表明该方法在降低集群时间开销的前提下有效节约了能耗。文献[19]对流式大数据处理的框架进行了优化，提出了一种基于 Storm 框架的实时资源调度节能策略，该策略通过构建 Storm 的能耗、响应时间以及资源利用率之间的数学关系，获得了满足高能效和低响应时间的条件，并以此建立了实时资源调度模型，该模型在实现最佳能效的前提下，对集群的资源进行调度。文献[20-21]通过资源调度与建立能耗优化模型达到节能的效果。目前，软件节能方法是研究的重点。

文献[22]提出一种可通知兼顾数据处理时延与集群能耗的弹性感知节能（LEEDSP）方法，该方法通过建立能耗感知模型，找到数据处理时延与集群能耗的平衡，达到节约集群能耗的效果。但是该节能策略并不是始终有效的，由于流式处理框架的数据一旦提交将持续运行，持续的执行该方法会导致集群数据处理的不平衡，从而影响集群数据的正常处理。

文献[23]通过研究 Storm 集群的拓扑结构，提出一种基于 Storm 的线程重分配与数据迁移节能（ERDM）策略。该策略通过优化集群拓扑数据处理的路径，在提高数据处理效率、降低拓扑时延的条件下有效降低了集群能耗。但是该策略无法预测集群负载的波动与动态迁移集群内的数据，导致数据处理出现滞后，影响集群的性能。

本文提出的节能策略与现有研究成果的不同之处如下。

1) 文献[9-14]大多采用替换低能效电子元件或动态电压缩放管理，其成本高昂且难以部署在大规模集群环境中，本文通过使用弹性数据迁移节能策略，动态调节集群的规模，可以较好地部署在大规模的集群环境中，避免集群资源溢出与浪费。

2) 文献[15-22]大多从集群整体资源的使用情况出发，并未考虑数据迁移后节点资源对数据处理的影响，本文通过量化集群 CPU、内存与网络带宽，建立资源判定模型，避免了数据迁移对节点的影响。

3) 文献[23]虽然评估集群数据迁移后节点资源的使用情况，但是无法准确定位集群执行数据迁移的时间，可能造成一定的误差，本文通过建立负载预测模型，定位节点资源的使用情况，能够及时响应策略的执行。

## 2 问题建模与分析

为量化集群作业处理数据的能耗，本节首先建立负载预测模型，通过预测集群数据处理的负载波动，及时响应迁移策略的执行；其次，建立资源判定模型，通过节点计算负载的变化趋势，确定节点的资源占用情况，判断节点数据处理是否出现资源过载或过剩，从而执行弹性数据迁移策略；再次，建立最优数据迁移模型，将资源过剩节点内的资源按照最优路径迁出，并删除该节点，将资源过载节点内的数据部分迁出，缓解节点资源紧张的问题；最后，建立能耗模型，根据数据处理时间、节点数与节点功率，评估集群能耗。整体架构如图 1 所示。

### 2.1 负载预测模型

由于集群的拓扑作业在处理数据时，按照默认的数据传输策略发送数据，可能会造成节点资源过剩或过载的问题。为确定节点资源的占用情况且避免出现调度滞后，需要预测节点计算负载的变化趋势，通过节点计算负载的上升与降低，以判断节点的资源占用。

令源节点发送数据的速率为 $V=\{v_1,v_2,\cdots,v_n\}$，样本的负载波动为 $\varepsilon$，计算负载的均值为 $\mu$。引入差分整合移动平均自回归（ARIMA, autoregressive integrated moving average）模型[24]对节点的资源占用与数据流速进行预测，要求集群的负载

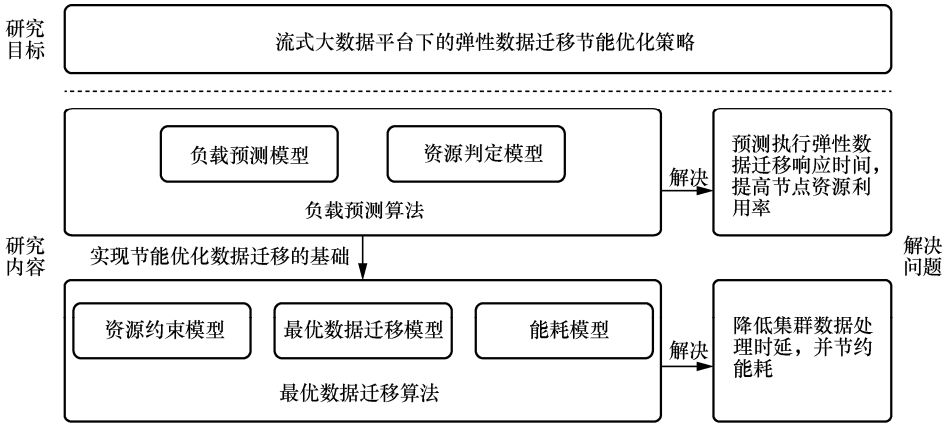



图 1　整体架构

波动与数据流速的均值相对稳定，即负载波动与数据流速之间存在相关关系，集群的数据处理负载趋于平稳。

实际情况下，集群的负载波动与数据流速之间剧烈波动，上一时刻的流速与下一时刻的流速产生较大的变化，则令集群在 $t$ 时刻与 $t$–1 时刻的负载一阶差分取值为 $v_t^1 = v_t - v_{t-1}$。然而，由于一阶差分不足以解决负载的波动问题，因此在一阶差分的基础上计算 $t$ 时刻与 $t$–1 时刻的流速差值，从而得到计算负载的二阶差分。依次类推，获得负载波动 $d$ 阶差分，使数据流速与负载波动之间趋于平稳，则 $d$ 阶差分为

$$v_t^d = v_t^{d-1} - v_{t-1}^{d-1} \tag{1}$$

**定义 1**　负载预测模型。为预测集群内的负载情况，需要确定节点数据流速与负载波动之间的函数关系，由于数据在 $t$ 时刻的数据流速为 $v_t^d$，$t$–1 时刻的数据流速为 $v_{t-1}^{d-1}$，负载波动为 $\varepsilon$，计算负载的均值为 $\mu$，则在 $i$ 时刻获得预测模型为

$$v_t^d = \sum_{i=1}^{p} \lambda_i v_{t-i}^{d-1} + \varepsilon_i + \mu_t \tag{2}$$

其中，$\lambda_i$ 表示预测模型的参数，根据最大似然估计方法计算；$p$ 表示当前时刻的数据流速；$\mu_t$ 表示 $t$ 时刻计算负载的均值；$\varepsilon_i$ 表示在 $i$ 时刻负载的波动。此外，由于 $\varepsilon_i$ 可能影响预测模型的准确性，因此需要尽可能消除负载变化带来的波动问题。通过取负载波动的平均值以消除负载变化带来的影响，则在 $i$ 时刻负载的波动可表示为

$$\varepsilon_i = \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_0 \tag{3}$$

其中，$\theta_i$ 表示该模型的参数，根据最大似然估计方法计算；$\varepsilon_0$ 表示初始时刻负载的波动；$q$ 表示当前时刻负载的波动。将式(3)代入式(2)，可得

$$v_t^d = \sum_{i=1}^{p} \lambda_i v_{t-i}^{d-1} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_0 + \mu_t \tag{4}$$

此外，根据总数的波动预测 $d$ 阶差分具体取值，并通过贝叶斯信息准则（BIC, Bayesian information criterion）确定 $p$ 与 $q$ 的取值，可得

$$\mathrm{BIC}=d\ln(n) - 2\ln(v) \tag{5}$$

其中，$d$=$p$+$q$ 表示待求参数值，$n$ 表示样本总数，$v$ 表示预测模型。根据评估，确定 BIC 的最小取值为最优参数值，获得三元组($d$, $p$, $q$)。

### 2.2　资源判定模型

根据负载预测模型预测节点计算负载变化趋势，确定节点的资源占用情况。若节点计算负载稳定在一个较高值，则节点的资源占用不断提高；若节点的计算负载稳定在一个较低值，则节点的资源占用不断降低。因此，通过建立资源判定模型可以找到资源过剩与过载节点。

**定义 2**　资源判定模型。令流式计算的节点集合为 $\{n_1, n_2, \cdots, n_M\}$，集群节点资源的极限为 $R_n$，定义 1 预测的原节点资源占用为 $O_n$，则节点 CPU、内存、网络带宽、磁盘空间四类资源的极限为 $R_n = \{R_n^C, R_n^M, R_n^B, R_n^D\}$，预测节点的四类资源为 $O_n = \{O_n^C, O_n^M, O_n^B, O_n^D\}$，获得

$$\left\{O_{n_i}^C << R_{n_i}^C \text{ and } O_{n_i}^M << R_{n_i}^M \text{ and } O_{n_i}^B << R_{n_i}^B ; \right.$$
$$\left. \text{and } O_{n_i}^D << R_{n_i}^D \right\} = R_s \tag{6}$$

其中，$n_i$ 表示集群内的第 $i$ 个节点，$R_s$ 表示资源过剩节点。根据式(6)可知，如果原节点的四类资源均远远小于资源极限，则定义该节点为资源过剩节点（经多次实验测试，四类资源占用率均小于 30%为资源过剩节点）。若预测节点的资源接近资源极限，获得

$$\left\{O_{n_i}^C \leqslant R_{n_i}^C \text{ or } O_{n_i}^M \leqslant R_{n_i}^M \text{ or } O_{n_i}^B \leqslant R_{n_i}^B \text{ or } O_{n_i}^D \leqslant R_{n_i}^D\right\}=R_f \tag{7}$$

其中，$R_f$ 表示资源过载节点。根据式(7)可知，如果原节点四类资源有一类接近极限，则定义该节点为资源过载节点（经多次实验测试，四类资源占用率大于 90%为资源过载节点）。资源判定模型如图 2 所示。

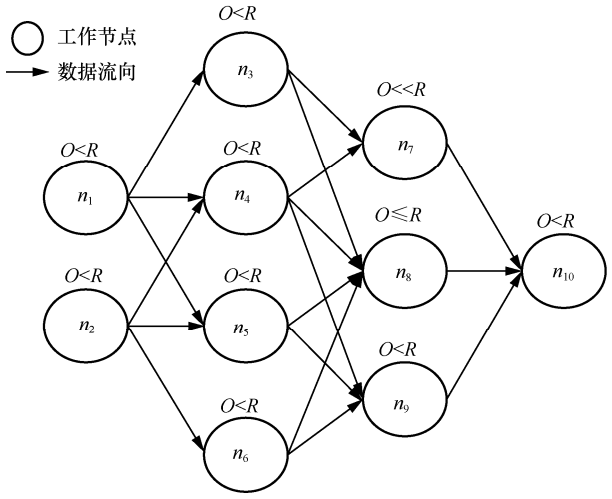



图 2 资源判定模型

由图 2 可知，$n_7$ 为资源过剩节点， $n_8$ 为资源过载节点。根据资源判定模型确定节点资源的占用情况，划分集群内的资源过剩与过载节点，为建立最优数据迁移模型奠定了基础。

### 2.3 最优数据迁移模型

根据定义 1 与定义 2，找到数据迁移的时机以及数据迁移的节点，将资源过剩节点内的数据全部迁出，资源过载节点内的数据部分迁出，具体的迁移方案通过最优数据迁移模型确定。此外，为避免迁入数据的节点成为新的资源过载节点，首先需要建立资源约束模型。

**定义 3** 资源约束模型。令节点迁入数据后，增加的节点资源为 $G_n$，为满足数据迁入后该节点不会成为新的资源过载节点，则有

$$\left\{O_{n_i}^C+G_{n_i}^C < R_{n_i}^C \text{ or } O_{n_i}^M+G_{n_i}^M < R_{n_i}^M \text{ or } O_{n_i}^B+G_{n_i}^B < R_{n_i}^B \text{ or } O_{n_i}^D+G_{n_i}^D < R_{n_i}^D\right\}=\text{RC} \tag{8}$$

其中，RC 表示资源约束模型，$G_{n_i}$ 表示迁入数据增加的四类节点资源。式(8)表示该节点现有资源占用率，不会导致其成为资源过载节点。为确定节点数据的迁移过程，建立最优数据迁移模型。

**定义 4** 最优数据迁移模型。由于资源过剩与过载节点内的数据都需要迁出，因此最优数据迁移模型需要满足以下条件：被迁入数据的节点需要尽可能满足资源约束模型，但集群节点内数据负载波动，导致被迁入数据后的节点可能不满足资源判定模型，则根据资源判定模型，该节点成为新的资源过载节点，需要重新对其节点内的数据进行迁移。此外，为避免节点数据出现不相容的问题，数据迁出节点通过其上游节点，向其他下游节点迁入数据。最优数据迁移模型如图 3 所示。

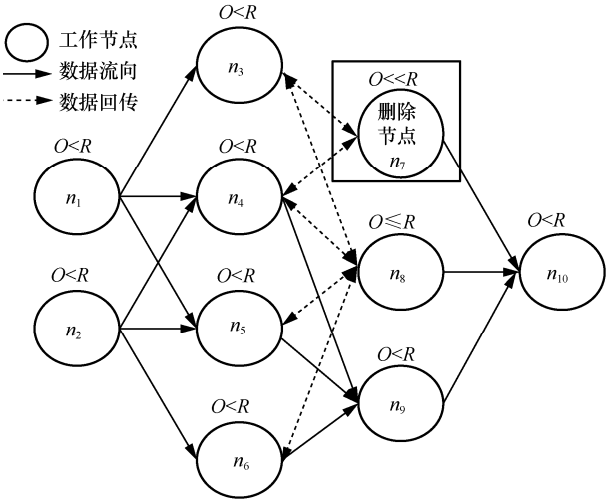



图 3 最优数据迁移模型

由图 3 可知，$n_7$ 的数据通过 $n_3$ 与 $n_4$ 向 $n_8$ 与 $n_9$ 迁入，且 $n_7$ 的数据全部迁出后删除该节点；$n_8$ 的数据通过 $n_4$、$n_5$ 与 $n_6$ 向 $n_9$ 迁入，以此降低 $n_8$ 的资源占用。根据式(8)与图 3，获得数据迁移模型为

$$n_i \xrightarrow[\text{UN}]{\text{RC}} n_j \tag{9}$$

其中，RC 表示资源约束模型，UN 表示 $n_i$ 的上游节点。式(9)表示过剩或过载节点 $n_i$ 在满足资源约束模型的基础上，将 $n_i$ 的数据通过其上游节点迁入 $n_j$。此外，若节点 $n_j$ 在迁入数据后不满足资源约束模型，

则 $n_j$ 根据资源判定模型成为新的资源过载节点，获得模型为

$$n_j \xrightarrow{\text{RD}} n_i \tag{10}$$

其中，RD 表示资源判定模型。

### 2.4　能耗模型

流式处理框架的能耗一般与节点的功率、节点数据处理的时间以及节点的个数有关，根据定义 3，确定数据迁移后集群内的数据实际处理情况，获得能耗模型。

令集群节点的平均功率为 Power，节点数据的处理时间为 Time，集群内节点的个数为 $n$，则集群产生的总能耗 Energy 为

$$\text{Energy}=n(\text{Power}\times\text{Time}) \tag{11}$$

令原集群节点的平均功率为 $\text{Power}_o$，节点数据的处理时间为 $\text{Time}_o$，由于集群内节点的个数为 $n$，则原集群的总能耗 $\text{Energy}_o$ 为

$$\text{Energy}_o=n(\text{Power}_o\times\text{Time}_o) \tag{12}$$

令执行节能策略后，集群内节点的个数为 $m$，其中，数据发生改变的节点个数为 $s$，节点的平均功率为 $\text{Power}_y$，节点数据处理的时间为 $\text{Time}_y$，则现集群的总能耗 $\text{Energy}_y$ 为

$$\text{Energy}_y=(m-s)(\text{Power}_o\times\text{Time}_o)+s(\text{Power}_y\times\text{Time}_y) \tag{13}$$

其中，$m-s$ 表示数据未发生改变的节点个数，则节点的平均功率与数据处理时间同样未发生改变。为计算集群内节约的能耗，获得能耗节约模型 $\text{Energy}_s$ 为

$$\text{Energy}_s=\text{Energy}_o-\text{Energy}_y \tag{14}$$

将式(12)与式(13)代入式(14)，可得执行弹性数据迁移节能策略节约的能耗为

$$\begin{aligned}&\text{Energy}_s=n(\text{Power}_o\times\text{Time}_o)-\\&((m-s)(\text{Power}_o\times\text{Time}_o)+s(\text{Power}_y\times\text{Time}_y))=\\&(n-m)(\text{Power}_o\times\text{Time}_o)-\\&s(\text{Power}_y\times\text{Time}_y-\text{Power}_o\times\text{Time}_o)\end{aligned} \tag{15}$$

## 3　算法的设计及分析

在第 2 节理论模型的基础上，本节提出流式大数据平台下的弹性数据迁移节能优化策略，该策略通过预测集群内负载波动与数据流速的变化趋势，确定数据迁移的约束条件与时机，使之高效节能地执行数据迁移节能策略。

### 3.1　负载预测算法

为判断集群节点的负载波动、数据流速以及数据迁移算法的响应时机，明确彼此之间的函数关系，并在此基础上确定资源过载与过剩节点，本文设计了负载预测算法，根据 BIC 确定不同参数的值。具体的算法描述如算法 1 所示。

**算法 1**　负载预测算法

**输入**　数据预测样本 $F_n$

**输出**　资源过剩与过载节点 $n_i$

**初始化**　$v_i$，$\varepsilon$，$\mu$

1) $F_n=\text{prediction}(F,v_i,\varepsilon)$；

/*预测样本节点内的数据流速与负载波动*/

2) $v_t=\text{different}(v,\mu,d)$；

/*根据计算负载均值训练差分的取值*/

3) $\text{BIC}=d\ln(n)-2\ln(v)$；

/*根据 BIC 计算 $p$ 与 $q$ 的取值*/

4) model=ARIMA（$d,p,q$）；

/*建立负载预测模型*/

5) for $n_i=n_1$ to $n_x$ do

6)　节点的预测资源与资源的极限比较；

7)　if $O_{n_i} \ll R_{n_i}$ then

8)　　节点 $n_i$ 满足 $R_s$；

9)　else

10)　　if $O_{n_i} \leqslant R_{n_i}$ then

11)　　节点 $n_i$ 满足 $R_f$；

12)　　else

13)　　节点 $n_i$ 为正常工作节点；

14)　　end if

15)　end if

16) end for

17) return $n_i$；

算法 1 的输入参数为预测样本 $F_n$，输出参数为资源过剩与过载节点 $n_i$，初始化参数为集群节点数据流速 $v_i$、负载波动 $\varepsilon$ 与计算负载均值 $\mu$。算法第 1)～2)行表示对集群的数据流速与负载波动进行预测，找到负载最平稳时 $d$ 阶的取值。算法第 3)～4)行表示根据 $d$ 阶差分取值与贝叶斯信息准则计算当前时刻负载波动与流速的取值，并根据 $d$、$p$、$q$

的取值建立负载预测模型。算法第 5)～17)行表示遍历集群节点的资源占用情况，与节点资源极限进行比较，根据式(6)，如果节点资源均远小于资源极限，则节点为资源过剩节点；根据式(7)，如果节点资源有一类接近极限，则节点为资源过载节点；如果不满足上述 2 种情况，则节点为正常工作节点。

由算法 1 可知，首先，预测集群节点的数据流速与负载波动，并根据负载均值训练差分的取值，其算法复杂度为 $O(1)$；其次，根据贝叶斯信息准则计算 $p$ 与 $q$ 的取值，进一步建立负载预测模型，其算法复杂度为 $O(1)$；再次，通过遍历集群节点的资源占用情况并与资源极限进行对比，根据对比信息确定工作节点是否为资源过载、过剩或正常工作节点，其算法复杂度为 $O(n\times1\times1)$。则负载预测算法的时间复杂度 $T(A)$为

$$T(A)=O(1)+O(1)+O(n\times1\times1)=O(n) \tag{16}$$

### 3.2 最优数据迁移算法

根据算法 1 的预测结果，获得资源过载与过剩节点，以此设计合适的数据迁移算法。该算法存在 3 个制约条件：1) 数据的迁移需要尽可能满足资源约束模型；2) 若被迁入数据的节点由于负载波动而不满足资源约束模型，根据资源判定模型，该节点成为新的资源过载节点；3) 为防止节点内数据出现不匹配问题，数据迁移算法需要通过资源过载与过剩节点的上游节点进行迁移。具体的算法描述如算法 2 所示。

**算法 2** 最优数据迁移算法

**输入** 原集群拓扑结构 Topology

**输出** 新的集群拓扑结构 Topology′

**初始化** 资源过剩与过载节点 $n_i$

1) 从 Topology 中根据算法 1 找到 $n_i$；
2) for $n_i$=$n_1$ to $n_x$ do
3) $n_i$ 通过上游节点找到对应的下游节点 $n_j$；
4) if $O_{n_j}+G_{n_j}\geqslant R_{n_j}$ then
5) $n_j$ 不满足资源约束模型；
6) else
7) if $n_i$ 为资源过剩节点 then；
8) $n_i$ 的数据全部迁入 $n_j$；
9) 删除资源过剩节点 $n_i$；
10) else
11) $n_i$ 的数据部分迁入 $n_j$；
12) end if
13) 根据式(10) $n_j$ 成为资源过剩节点；
14) $n_i'$ 的数据部分迁入 $n_j'$；
15) end if
16) end for
17) Zookeeper.update("configuration file")；
18) return Topology′；

算法 2 的输入参数为原集群拓扑结构 Topology；输出参数为新的集群拓扑结构 Topology′；初始化参数为资源过剩与过载节点 $n_i$。算法第 1)～6)行通过遍历下游节点的资源占用，判断该节点是否满足资源约束模型，若节点原资源与迁入资源之和大于资源极限，则选择其他下游节点。算法第 7)～12)行表示判断节点 $n_i$ 为资源过剩节点或资源过载节点，若 $n_i$ 为资源过剩节点，则迁出全部数据后删除该节点；若 $n_i$ 为资源过载节点，则迁出部分导致节点资源过载的数据。算法第 13)～16)行表示节点内数据负载波动，导致 $n_j$ 成为新的资源过载节点，需要重新迁移该节点内的数据；算法第 17)～18)行表示更新 Zookeeper 的配置文件，生成新的集群拓扑结构。

由算法 2 可知，首先，选择合适的下游节点用于数据的迁移，其时间复杂度为 $O(n\times1)$；其次，判断 $n_i$ 为资源过剩节点或资源过载节点，其时间复杂度为 $O(1\times1)$；再次，迁移新资源过载节点内的数据，其时间复杂度为 $O(1)$；最后，更新集群的配置文件，确保数据可以正常处理，其时间复杂度为 $O(1)$。则最优数据迁移算法的时间复杂度 $T(B)$为

$$T(B)=O(n\times1)+O(1\times1)+O(1)+O(1)=O(n) \tag{17}$$

此外，弹性数据迁移节能优化策略由算法 1 与算法 2 组成，则其时间复杂度为

$$T(C)=T(A)+T(B)=O(n) \tag{18}$$

### 3.3 算法的实现与部署

实验选择搭建 Flink 框架作为验证策略的集群，在 Flink 中一般通过 Kafka 从数据源读取数据，并通过不同工作节点（TaskManager）协同完成的，之后将技术结果写入 Hadoop 分布式文件系统（HDFS）中。为实现弹性数据迁移节能优化策略，首先需要监控数据处理产生的负载，以此预测计算

负载的变化趋势，制定合适的弹性数据迁移策略，并保存到 Zookeeper 的配置文件中，最后根据迁移结果更新配置文件。为部署弹性数据迁移节能优化策略，本文在原有 Flink 框架的基础上又新增了 5 个模块，具体的架构如图 4 所示。

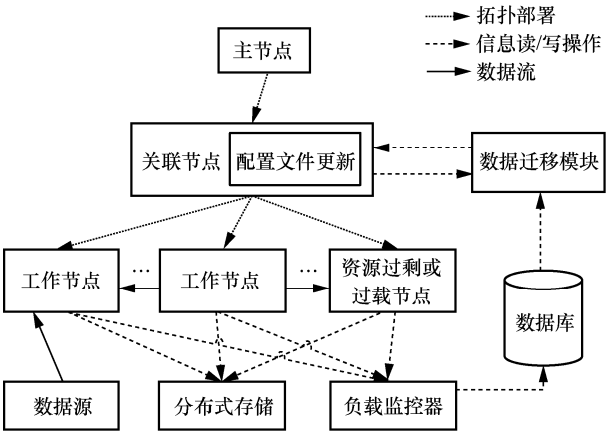



图 4　弹性数据迁移节能优化策略具体架构

1) 负载监控器。部署算法 1，监控节点内的负载波动与数据流速，用于选择资源过剩或过载节点，并把信息发送到数据库。

2) 数据库。负责存储负载监控器传输的信息，将其反馈给数据迁移模块，并实时进行更新。

3) 数据迁移模块。部署算法 2，根据负载反馈信息制定数据迁移计划，将资源过剩或过载节点内的数据迁出。该模块为节能优化策略的核心。

4) 配置文件更新。针对算法 2 改变的集群拓扑，实时更新 Zookeeper 的配置文件，防止因 Zookeeper 的记忆功能而出现数据传输错误的问题。

5) 分布式存储。存储集群内数据的状态信息，并在执行算法 2 的过程中，以异步的方式从 HDFS 中提取对应的状态信息，保证数据的一致性。

## 4　实验结果与分析

为评估本文策略负载波动、计算资源以及能耗的变化，将其分别与原集群、LEEDSP[22] 以及 ERDM[23]进行对比。实验选择 Streaming-Benchmark 作为基准测试，验证负载预测与参数选择的合理性，并对节点资源占用与节能优化结果进行分析与讨论。

### 4.1　实验参数环境

本节实验同构搭建 25 个节点的 PC，包括一个运行主控程序的 JobManager 节点、18 个运行作业的 TaskManager 节点、3 个发送数据的 Kafka 节点，以及 3 个关联作业的 Zookeeper 节点。此外，共有 21 台测电器部署于 21 个节点，其中 JobManager、Kafka、Zookeeper 节点分别部署一台测电器，TaskManager 节点上部署 18 台测电器，用于监测其功耗，原因为不同 TaskManager 节点的作业量不同，其功耗差异较大。测电器型号均为北电电力监测仪（USB 智能版），标准为 GB/T17215-2003。

最后，每个 PC 的网卡统一为 100 Mbit/s LAN，CPU 为 i7 处理器，内存为 8 GB，硬盘为 1 TB。除硬件配置，各节点的软件配置相同，具体的硬件参数和软件参数分别如表 1 和表 2 所示。

**表 1　硬件参数**

| 硬件配置项 | 参数 |
|---|---|
| CPU | Intel core i7 4790 3.6 GHz Quad Core |
| 内存 | 8 GB DDR3 1 066 MHz |
| 硬盘 | 1 TB, 7 200 r/min |
| 网络 | 100 Mbit/s LAN |
| 测电器 | GB/T17215-2003 |

**表 2　软件参数**

| 软件配置项 | 参数 |
|---|---|
| OS | CentOS 6.8 |
| Flink | 1.6.0 |
| Hadoop | 2.7.4 |
| JDK | jdk1.8.0-181-Linux-X64 |
| Zookeeper | 3.4.10 |
| Kafka | 2.10-0.8.2.0 |
| MySQL | 5.7.27 |
| VMware | 12.1.1 |
| MATLAB | R2014a (8.3.0.532) 64 bit |

根据上述配置环境，25 个节点组成的网络拓扑结构如图 5 所示。

此外，实验选择 Streaming-Benchmark 作为基准测试，该测试为 Yahoo 公司开发用于广告数据的实时分析作业，其中存在大规模复杂的状态数据，需要消耗大量的 CPU 与内存资源。根据基准测试，Flink 的重要参数配置如表 3 所示。

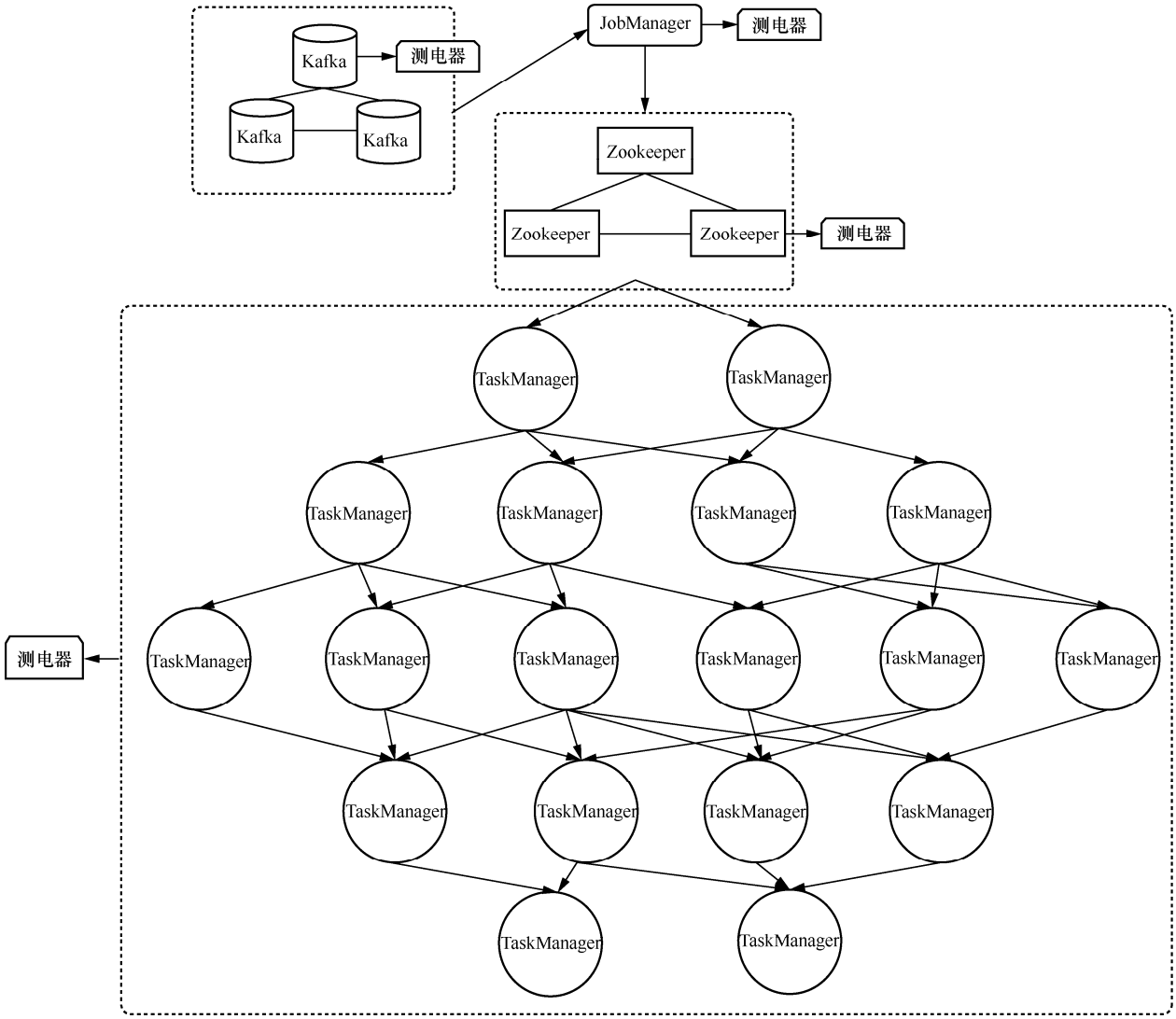



图 5　25 个节点组成的网络拓扑结构

**表 3　　Flink 的重要参数配置**

| 配置项 | 参数值 | 说明 |
|---|---|---|
| JobManager.heap.size/MB | 2 048 | 主节点内存 |
| TaskManager.heap.size/MB | 2 048 | 工作节点内存 |
| TaskManager.numberOfTaskSlots | 2 | 节点线程数目 |
| high-availability | Zookeeper | 开启 HA 模式 |
| state.backend | rocksdb | 状态数据存储 |
| state.backend.incremental | true | 增量式快照 |
| TaskManager.memory.segment-size | 32 758 | 内存分块大小 |

为了验证策略的有效性，实验选择原集群、LEEDSP 以及 ERDM 进行对比。其中，LEEDSP 的核心思想是通过使用 DVFS[10]技术，动态调节流式作业迁移后节点的电压，达到节能的效果，且适用于大多数流式计算框架（包括 Storm、Flink 以及 Spark Streaming 等）。ERDM 的核心思想为根据最优通信开销，动态迁移节点内的作业，以此达到降低传输时延、节约能耗的目的，该策略同样适用于 Flink 框架。此外，LEEDSP 与 ERDM 都是流式处理节能计算经典的研究成果，具有非常高的代表性与前沿性。

为保障同等条件下验证本文策略的效果，LEEDSP 与 ERDM 的实验参数都与表 1~表 3 保持一致。此外，每组实验测试 10 次取其平均值，以保证数据的准确性。

### 4.2　负载预测与资源占用

为使用 ARIMA 模型预测流式计算的负载波动，首先需要确定 $d$ 阶差分的取值，使计算负载与负载波动趋于平稳，则根据式(1)训练 $d$ 的取值，实验统计 300 s 内结果如图 6 所示。

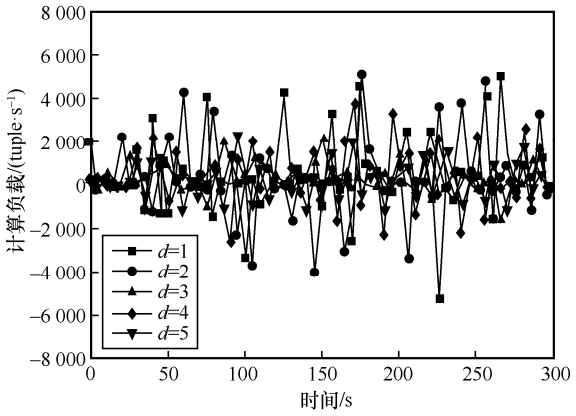



图 6　负载差分取值

根据图 6 可知，当 $d$=1 时，计算负载的波动较大，完全无法满足预测模型对样本平稳性的需求；当 $d$=3 时，120 s 后的计算负载波动趋于平稳；当 $d$=5 时，虽然计算负载的波动趋于收敛，但是与 $d$=3 时的负载波动区别不大，而 $d$=5 时的计算成本较高，且会产生一定的能耗，不利于节能策略的实施，因此，确定 $d$=3 为预测模型的第一个参数。

根据式(3)、式(5)以及 $d$ 的取值，计算 $q$ 与 $p$ 的取值。当 $d$=3、样本总数 $n$=20 时，120 s 内数据的负载波动趋于平稳，获得当前负载的波动 $q$=4，负载的流速 $p$=5，则确定三元组 $(d, p, q)$为(3, 5, 4)，建立负载预测模型。通过负载预测模型，统计 300 s 内集群内各节点在 Streaming-Benchmark 下负载吞吐量的平均值，具体结果如图 7 所示。

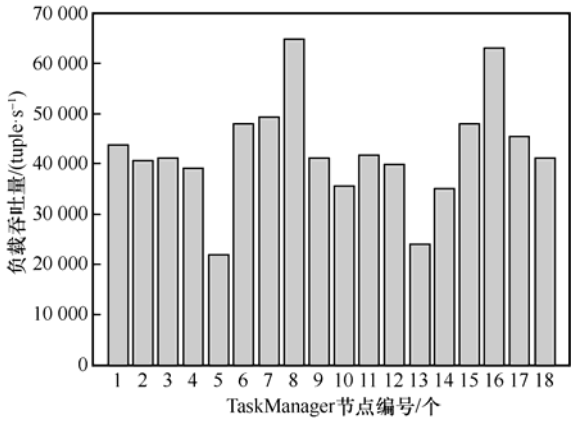



图 7　负载预测平均负载吞吐量

图 7 反映了 300 s 内 18 个 TaskManager 节点的负载吞吐量平均值，其中第 8 个节点与第 16 个节点的负载吞吐量远高于其他节点，而第 5 个节点与第 13 个节点的负载吞吐量远低于其他节点。根据资源判定模型，统计 18 个 TaskManager 的资源占用情况，确定资源过剩与过载节点，具体的结果如图 8 所示。

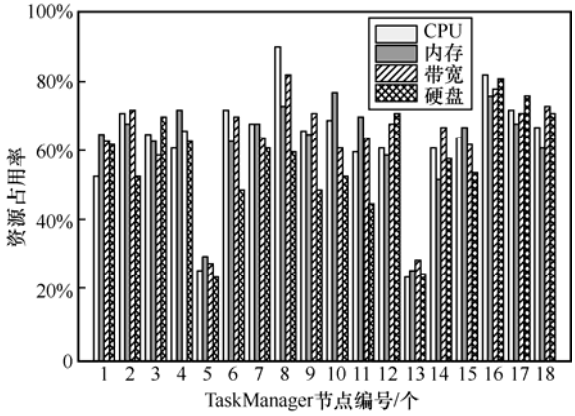



图 8　节点资源占用

根据图 8 可知，第 8 个节点为资源过载节点，第 5 个节点与第 13 个节点为资源过剩节点，则通过资源约束模型与最优数据迁移模型，将第 5 个节点、第 8 个节点以及第 13 个节点内数据迁入合适的节点，其中第 8 个节点内的数据部分迁出，而第 5 个节点与第 13 个节点内的数据全部迁出后删除该节点。预测集群内 16 个 TaskManager 节点的负载吞吐量平均值，发现第 15 个节点的负载吞吐量远高于其他节点，根据资源判定模型，确定其为资源过载节点，具体结果如图 9 所示。

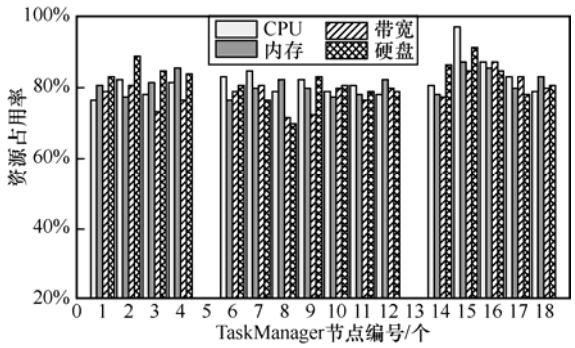



图 9　第一次迁移数据后节点资源占用

根据图 9 可知，确定第 15 个节点为新的资源过载节点，则通过资源约束模型与数据迁移模型，将该节点内的数据部分迁出，预测集群 TaskManager 节点的平均负载吞吐量，具体结果如图 10 所示。

根据图 10 可知，此时集群内各 TaskManager 节点的负载波动稳定在一个合理的区间。因此，解决了集群内节点资源过载与过剩的问题。

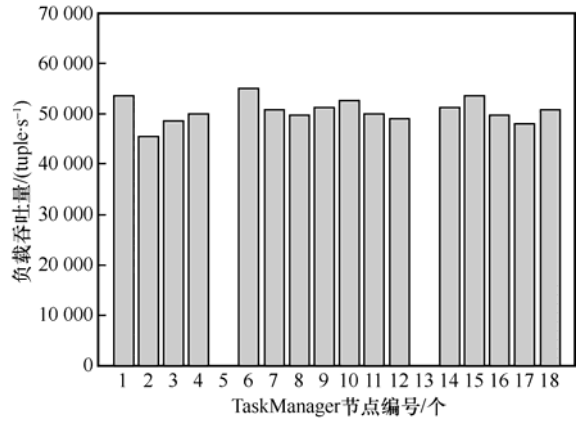



图 10　最终预测平均负载吞吐量

### 4.3　节能效果及分析

为计算 EEDM-BDSP 的节能效果，首先需要统计数据处理的时延与节点的功率。

1) 集群计算时延

计算时延是评价集群性能的重要指标，本文将所提 EEDM-BDSP、ERDM、LEEDSP，以及原集群进行对比，实验统计 900 s 内在 Streaming-Benchmark 下，4 种方法的计算时延，具体结果如图 11 所示。

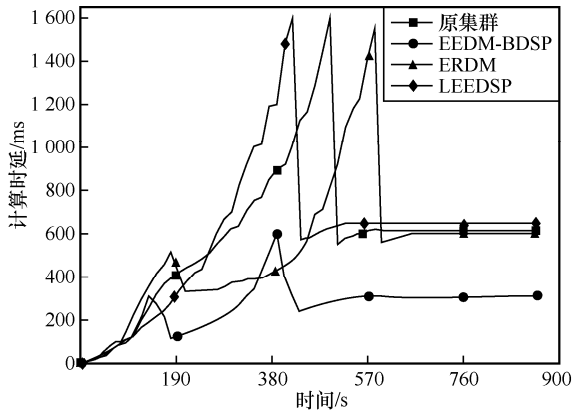



图 11　不同策略下的计算时延对比

根据图 11 可知，480 s 后 EEDM-BDSP 的计算时延趋于平稳，且平均计算时延相对较低，而原集群的 120 s 后的计算时延逐渐呈指数增长，其原因为 120 s 后集群节点的计算资源接近极限，数据的处理出现拥塞，并且随时间的推移不断加大。但原集群在 480 s 后计算时延突然下降，其原因为资源负荷节点崩溃，流式集群不再向其发送数据。

ERDM 在 90 s 后出现计算资源快速增长的问题，在 180 s 后，通过执行迁移策略，降低集群节点的计算时延，但在 360 s 后又出现计算时延骤然增加的问题，并于 600 s 后同样出现负荷节点崩溃。而 EEDM-BDSP 在 120 s 后计算时延增加的同时快速执行数据迁移策略，避免了 90 s 的响应时间，在 360 s 后再次出现计算时延激增的同时，重新执行数据迁移策略，避免节点再次出现资源过载的问题，其原因如下。在 120 s 与 360 s 时，EEDM-BDSP 通过负载预测算法快速找到了资源过载节点，并对其进行数据迁移，避免节点因资源不足而导致计算时延指数增长的问题。

LEEDSP 在 120 s 后同样出现计算时延增加的问题，并在 240 s 后出现计算时延骤然增加的问题，其原因是 LEEDSP 在 240 s 后，通过使用 DVFS 技术降低节点电压，使其节点数据的处理效率降低，进而导致数据的拥塞更加严重，增加了集群的计算时延。

综上所述，EEDM-BDSP 降低集群计算时延的效果最好，其平均计算时延为 315.43 ms，而原集群节点由于出现资源满负荷，其平均计算时延为 602.14 ms。相对于原集群，EEDM-BDSP 的计算时延平均下降 47.26%。因此，执行 EEDM-BDSP 有效提高了集群的性能。

2) 集群能耗

集群能耗是反映节能策略优劣的重要指标，本文通过节点功率与节点数据处理时间的乘积以计算集群的能耗，如式(11)所示。此外，通过式(15)计算 EEDM-BDSP 节约的能耗。为计算集群的能耗，首先需要统计单位时间内节点的功率，并引入 ERDM、LEEDSP 以及原集群与 EEDM-BDSP 进行对比，具体结果如图 12 所示。

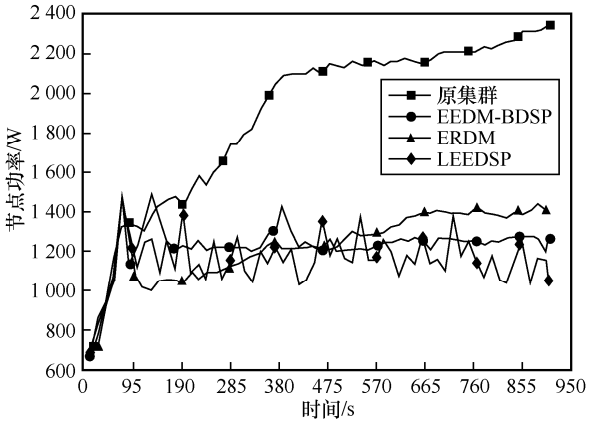



图 12　不同策略下的 TaskManager 节点功率对比

如图 12 所示，原集群、EEDM-BDSP、ERDM 以及 LEEDSP 中单个 TaskManager 节点的平均功率分别为 1 969.4 W、1 233.1 W、1 251.9W 以及

1 163.4W。与原集群相比，EEDM-BDSP 的平均功率降低了 37.43%，其原因为原集群出现节点资源瓶颈，数据传输出现拥塞，导致节点的功率增大。但 EEDM-BDSP 在 120 s 与 360 s 时，节点的功率出现激增，而 ERDM 在 90 s 时，节点的功率同样出现激增，其原因为在这些时刻，2 种节能策略都进行数据迁移，节点计算资源的占用较大，但由于时间较短，对节点平均功率的影响较小，可忽略不计。ERDM 在 360 s 后同样出现节点功率增加，其原因为 ERDM 失效。LEEDSP 功率在 180 s 时同样出现激增，之后迅速降低，其原因为通过 DVFS 动态降低了节点的功率，但由于 DVFS 的实现具有偶然性，且响应较为困难，不适合部署在大规模集群中。此外，由于执行 EEDM-BDSP 仅有 16 个 TaskManager 节点，EEDM-BDSP 总平均功率为 20 124.1 W，原集群总平均功率为 36 135.2 W，因此，总平均功率降低了 44.31%。通过节点的功率，计算相同数据量下集群的能耗，具体结果如图 13 所示。

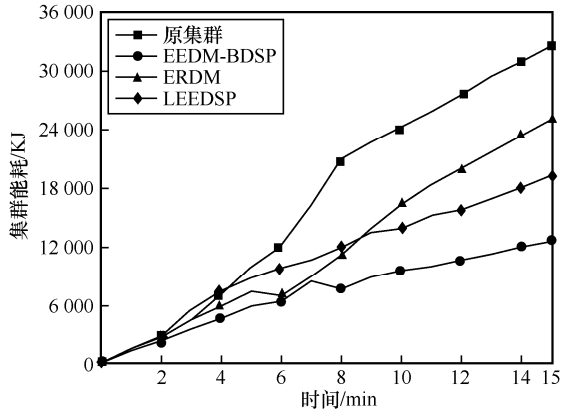



图 13　相同数据量下的集群能耗对比

根据图 13 可知，15 min 内 EEDM-BDSP 的能耗为 12 534.7KJ，原集群的能耗为 32 678.5 KJ，ERDM 的能耗为 25 138.2 KJ，LEEDSP 的能耗为 19 352.1 KJ。与原集群相比，EEDM-BDSP 的能耗降低了 61.64%，其原因为原集群随着数据量的增加，节点的资源达到过载，集群数据处理出现拥塞，并伴随高额的能耗。相比于 ERDM，由于 EEDM-BDSP 的节点资源占用率一直没有达到极限，而 ERDM 在 6 min 后，集群数据处理出现拥塞，节点资源同样接近极限，使之能耗迅速升高。相比于 LEEDSP，由于 LEEDSP 只考虑 CPU 的能耗，而流式计算框架的实时性导致内存与网络带宽等部件能耗也不容忽视，并伴随着数据量的增加，能耗所占比重不断增大。

综上所述，EEDM-BDSP 有效节约了集群的能耗。

## 5　结束语

高能耗是制约绿色流式计算框架发展的重要瓶颈之一。针对这一问题，本文通过研究流式计算框架的负载预测模型与资源判定模型，建立最优数据迁移模型以及能耗模型，并进一步提出流式大数据平台下的弹性数据迁移节能优化策略。该策略包含负载预测算法与最优数据迁移算法，使流式计算框架在降低节点资源瓶颈以及提高节点资源利用率的同时，有效节约集群的能耗。最后，实验通过与不同策略在计算时延与能耗两方面进行对比，验证策略的实际效果。下一步的研究工作主要包含以几个下方面。

1) 通过研究弹性资源分配节能策略，将其部署在边缘计算等新型场景，进一步推广节能的思想。

2) 通过数据压缩等方式，提高单位时间数据的处理效率，以此达到提高能效的目的。

3) 通过数据冗余删除等方法，节约节点存储资源，并提高其数据处理的能效。

## 参考文献：

[1] LV X, YE R Q, LIU M. Green cloud 2022[R]. 2023.

[2] WU H Q. Green ICT impact on energy conservation and emission reduction[J]. China Communications, 2008, 5(3): 79-84.

[3] KIM H, CHOI H, KANG H, et al. A systematic review of the smart energy conservation system: from smart homes to sustainable smart cities[J]. Renewable and Sustainable Energy Reviews, 2021: doi.org/10.1016/J.RSER.2021.110755.

[4] SEAGATE. Data age 2025[R]. 2023.

[5] 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): 97-117.
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): 97-117.

[6] 蒲勇霖, 于炯, 鲁亮, 等. 基于 Storm 平台的数据恢复节能策略[J]. 计算机研究与发展, 2021, 58(3): 479-496.
PU Y L, YU J, LU L, et al. Energy-efficient strategy based on data recovery in storm[J]. Journal of Computer Research and Development, 2021, 58(3): 479-496.

[7] SHIN D, JANG H, OH K, et al. An energy-efficient DRAM cache architecture for mobile platforms with PCM-based main memory[J]. ACM Transactions on Embedded Computing Systems, 2022, 21(1): 1-22.

[8] DAI X X, YANG P, ZHANG X Y, et al. RESPIRE: reducing spatial-temporal redundancy for efficient edge-based industrial video ana-

lytics[J]. IEEE Transactions on Industrial Informatics, 2022, 18(12): 9324-9334.

[9] ZENG Q S, DU Y Q, HUANG K B, et al. Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing[J]. IEEE Transactions on Wireless Communications, 2021, 20(12): 7947-7962.

[10] SAMRIYA J K, TIWARI R, Cheng X C, et al. Network intrusion detection using ACO-DNN model with DVFS based energy optimization in cloud framework[J]. Sustainable Computing: Informatics and Systems, 2022, 35: 100746.

[11] PIETRI I, ZHUANG S C, CASAS M, et al. Evaluating scientific workflow execution on an asymmetric multicore processor[C]// Proceedings of European Conference on Parallel Processing. Berlin: Springer, 2018: 439-451.

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

[13] FEI X W, LI K L, YANG W D, et al. Analysis of energy efficiency of a parallel AES algorithm for CPU-GPU heterogeneous platforms[C]//Proceedings of 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Piscataway: IEEE Press, 2019: 499-508.

[14] JAVADPOUR A, SANGAIAH A K, PINTO P, et al. An energy-optimized embedded load balancing using DVFS computing in cloud data centers[J]. Computer Communications, 2023, 197: 255-266.

[15] 蒲勇霖, 于炯, 鲁亮, 等. 基于 Storm 平台的数据迁移合并节能策略[J]. 通信学报, 2019, 40(12): 68-85.
PU Y L, YU J, LU L, et al. Energy-efficient strategy for data migration and merging in Storm[J]. Journal on Communications, 2019, 40(12): 68-85.

[16] CHENG D Z, CHEN Y, ZHOU X B, et al. Adaptive scheduling of parallel jobs in spark streaming[C]//Proceedings of IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2017: 1-9.

[17] XU X L, XUE Y, QI L Y, et al. Load-aware edge server placement for mobile edge computing in 5G networks[C]//Proceedings of Service-Oriented Computing. Cham: Springer International Publishing, 2019: 494-507.

[18] DAYARATHNA M, LI Y L, WEN Y G, et al. Energy consumption analysis of data stream processing: a benchmarking approach[J]. Software: Practice and Experience, 2017, 47(10): 1443-1462.

[19] 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.

[20] YANG S, LI F, SHEN M, et al. Cloudlet placement and task allocation in mobile edge computing[J]. IEEE Internet of Things Journal, 2019, 6(3): 5853-5863.

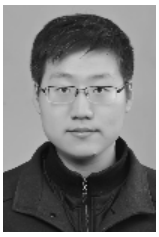
[21] HUANG X F, ZHOU S. Dynamic compression ratio selection for edge inference systems with hard deadlines[J]. IEEE Internet of Things Journal, 2020, 7(9): 8800-8810.

[22] MATTEIS D T, MENCAGLI G. Keep calm and react with foresight[J]. ACM SIGPLAN Notices, 2016, 51(8): 1-12.

[23] 蒲勇霖, 于炯, 鲁亮, 等. Storm 平台下的线程重分配与数据迁移节能策略[J]. 软件学报, 2021, 32(8): 2557-2579.
PU Y L, YU J, LU L, et al. Energy-efficient strategy based on executor reallocation and data migration in storm[J]. Journal of Software, 2021, 32(8): 2557-2579.

[24] YANG H, LI X, QIANG W, et al. A network traffic forecasting method based on SA optimized ARIMA-BP neural network[J]. Computer Networks, 2021, 193: 108102.

[作者简介]



**蒲勇霖**（1991− ），男，山东淄博人，博士，南京信息工程大学讲师、硕士生导师，主要研究方向为边缘计算、协同计算、绿色计算等。

**许小龙**（1988− ），男，江苏南通人，博士，南京信息工程大学教授、博士生导师，主要研究方向为边缘计算、服务计算等。

**于炯**（1964− ），男，北京人，博士，新疆大学教授、博士生导师，主要研究方向为网格计算、并行计算、分布式系统。

**李梓杨**（1993− ），男，新疆乌鲁木齐人，博士，新疆大学副教授、硕士生导师，主要研究方向为大数据分析、机器学习。

**国冰磊**（1991− ），女，湖北襄阳人，博士，湖北文理学院讲师，主要研究方向为分布式数据库系统、绿色计算。