数据空间中基于纠删码的数据布局策略

林兵, 姜海鸥, 檀啸, 陈星, 郑裕恒

引用本文

林兵, 姜海鸥, 檀啸, 陈星, 郑裕恒. 数据空间中基于纠删码的数据布局策略[J]. 计算机科学, 2026, 53(2): 196-206.

LIN Bing, JIANG Haiou, TAN Xiao, CHEN Xing , ZHENG Yuheng. Data Placement Strategy Based on Erasure Code in Data Space [J]. Computer Science, 2026, 53(2): 196-206.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241200199

数据空间中基于纠删码的数据布局策略

林　兵[1,3]　姜海鸥[2]　檀　啸[1]　陈　星[3,4]　郑裕恒[3,4]
1 福建师范大学物理与能源学院　福州 350117
2 北京大数据先进技术研究院数据空间技术与系统全国重点实验室　北京 100195
3 福建省网络计算与智能信息处理重点实验室　福州 350116
4 福州大学计算机与大数据学院/软件学院　福州 350108
(wheellx@163.com)

摘　要　针对云边环境下面向多目标优化的科学工作流数据布局问题,考虑数据可靠性、工作流执行时延和数据中心负载均衡等因素,提出了数据空间中基于纠删码的数据布局策略。首先,提出在科学工作流执行时使用低存储开销的纠删码冗余技术以提供容错能力,并通过构建数据空间来管理工作流产生的多样化数据;其次,设计了一种响应式多目标进化算法(Interactive Multi-Objective Evolution Algorithm,IMOEA),同时优化执行时延和数据中心负载均衡,通过与决策者交互,使算法生成的解决方案更符合决策者的期望,提高了优化结果的个性化和可接受性。实验结果表明,针对不同规模和类型的工作流,相比于DIST,MOGA 和 RAND 算法,IMOEA 在空间指标(Space,SP)上分别降低了 2.3%～36.34%,15.71%～44.01%和 22.50%～47.64%,在超体积指标(Hypervolume,HV)上分别优化了 7.84%～38.23%,14.65%～48.4%和 45.01%～109.45%。此外,IMOEA 算法可以很好地对决策者的偏好做出反应,找到令决策者满意的数据布局方案。

关键词:数据空间;云边环境;科学工作流;数据布局;纠删码;多目标优化
中图分类号　TP338

Data Placement Strategy Based on Erasure Code in Data Space

LIN Bing[1,3],JIANG Haiou[2],TAN Xiao[1],CHEN Xing[3,4] and ZHENG Yuheng[3,4]
1 College of Physics and Energy,Fujian Normal University,Fuzhou 350117,China
2 Advanced Institute of Big Data,Beijing,National Key Laboratory of Data Space Technology and System,Beijing 100195,China
3 Fujian Key Laboratory of Network Computing and Intelligent Information Processing,Fuzhou 350116,China
4 College of Computer and Data Science/College of Software,Fuzhou University,Fuzhou 350108,China

Abstract　In response to the multi-objective optimization layout problem of integrated data within scientific workflows in cloud-edge environments,factors such as data reliability,workflow execution latency,and data center load balancing are considered,and a data placement based on erasure coding within the data space is proposed. Firstly,low-storage-overhead erasure code redundancy technology is proposed to provide fault tolerance in scientific workflow execution,and a data space is constructed to manage the diverse data generated by the workflow. Secondly,an Interactive Multi-Objective Evolution Algorithm(IMOEA) is designed to simultaneously optimize execution latency and datacenter load balancing. By interacting with decision-makers,the algorithm generates solutions that better align with the decision-makers' expectations,enhancing the personalization and acceptability of the optimization results. Experimental results show that for workflows of different scales and types,compared to other algorithms such as DIST,MOGA,and RAND,IMOEA reduces spatial metrics(Space,SP) by 2.3%～36.34%,15.71%～44.01%,and 22.50%～47.64%,and improves hypervolume metrics(Hypervolume,HV) by 7.84%～38.23%,14.65%～48.4%,and 45.01%～109.45%,respectively. Additionally,IMOEA algorithm effectively responds to decision-makers' preferences,finding

到稿日期:2024-12-30　返修日期:2025-03-28
基金项目:国家自然科学基金(62072108);福建省高校产学合作项目(2022H6024,2021H6026);数据空间技术与系统全国重点实验室资助项目(QZQC2024015-3);福建省促进海洋与渔业产业高质量发展专项资金(FJHYF-ZH-2023-02);福建省技术创新重点攻关及产业化项目(2024XQ004).
This work was supported by the Natural Science Foundation of China(62072108),University-Industry Cooperation of Fujian Province(2022H6024,2021H6026),Founded Projects of the National Key Laboratory of Data Space Technology and Systems(QZQC2024015-3),Fujian Provincial Special Fund for Promoting the High-Quality Development of Marine and Fishery Industries(FJHYF-ZH-2023-02) and Fujian Province Key Technology Innovation and Industrialization Projects(2024XQ004).
通信作者:姜海鸥(seagullwill@foxmail.com)

satisfactory data placement solutions.

Keywords　Data space，Edge-cloud environments，Scientific workflows，Data placement，Erasure code，Multi-objective optimization

1　引言

工作流结构复杂，通常由许多存在数据依赖或控制关系的计算任务组成[1]。科学工作流系统是在各科学领域有着广泛应用的数据密集型系统。近年来，科学工作流中的数据量逐渐增大，使用云边环境来部署科学工作流成为一种常见举措，其主要优势在于云数据中心的巨额存储空间以及边缘数据中心传输数据低时延的特性。随着科学应用复杂性的增加，云边环境中的数据存储与处理面临着新的挑战。一方面，所存储和处理的数据逐渐由单一的结构化数据向存在非结构化数据的复杂数据转变，如存储处理结构规范的表格数据逐渐演变到复杂的图像和经验性文本类型数据，这需要新的方式来处理。另一方面，科学工作流中存在丢失会导致严重后果的关键数据，如何保证这类数据在云边环境下的可靠性也是当下需解决的重要问题。

对于逐渐复杂的数据，数据空间是一种新型网络架构，它是以计算为中心转变到以数据为中心，旨在将非结构化数据整合为结构化数据，从而实现云边环境中数据的高效存储与处理[2]。对于数据可靠性方面，随着海量数据的不断涌入和存储节点数量的快速增长，云边存储系统持续面临着由软件、硬件、网络和电源故障引起的数据丢失风险，这种原本发生概率较小的事件正逐渐演变为云边环境存储系统的常见现象。因此，关键数据必须在部分硬件故障时仍然能正常读写。通常，提高可靠性的最简单方法是对数据存储多个副本，以应对硬盘损坏率等情况。根据磁盘损坏率的估算，使用 3 个数据副本可以达到较高的可靠性水平。但这种方式会导致较多的额外存储空间，限制了存储系统的发展。为了解决这一问题，存储领域一直致力于寻找一种冗余较少的方式来实现高可靠性。本文研究了使用纠删码来高效存储数据空间中的关键数据。在纠删码方案中，要存储的数据被分为多个数据块和校验块，这些块分布在用户可访问的不同存储节点上。用户可以从任何可访问的数据中心检索数据块和校验块，从而构造原始数据以供使用。这种方式在不同节点上存储数据块和校验块，即使某个节点发生故障，也可以通过其他节点上的数据块和校验块进行数据恢复，提高了系统的可靠性并减少了存储空间的浪费，为存储系统的健康发展提供了更有效的解决方案。

引入纠删码技术后，数据块和校验块的动态管理、节点故障恢复机制的协调等，均增加了系统的管理负担。例如，系统可能需要动态调整数据块的存储位置，以保证在节点发生故障时能够通过其他可用节点恢复数据，这对存储管理提出了更高的要求，给实际应用带来了诸多挑战[3]。利用数据空间不仅能够有效地存储和检索数据，还能捕捉并保存数据之间的复杂关系。不同于传统的数据管理模式，数据空间具备动态维护数据的能力，能够适应数据的变化和更新，实时反映数据之间的相互联系，从而为数据分析和决策提供更为准确的信息支持。

引入数据空间进行数据管理的同时，科学工作流对快速响应与高效数据处理以及各节点负载均衡的要求同样必须得到满足，因此在数据空间中执行科学工作流数据管理时，时延和负载均衡都是关键考量因素。一方面，在需要快速响应和高效数据处理的应用场景中，如大规模数据分析和实时实验监测，执行时延的增加可能导致研究进展缓慢，影响数据分析的及时性，造成严重的后果。另一方面，需要确保各个节点的负载分配合理，避免出现某些节点过载导致的性能下降甚至系统崩溃的情况。通过有效的负载均衡策略，可以最大程度地利用资源，提高系统整体的性能和可靠性。然而，在实际应用中，往往很难同时最大化时间性能和负载均衡，因为这两个目标通常是相互矛盾的。在优化执行时延的过程中，很难保证数据分布的均匀性，这可能导致部分节点负载过重，影响系统整体性能。此外，决策者往往缺乏对数据空间现状的了解，也无法准确判断何种执行时延和负载均衡适合他们的应用。优化数据的存储效率、访问速度、可靠性是数据布局策略需考虑的关键因素。对于数据空间中的数据布局，在考虑关键数据可靠性，兼顾多种优化目标的同时，提出一种合理的数据布局策略至关重要。

本文的主要贡献如下：

1）提出一种数据空间中基于纠删码的数据布局模型，以较低的存储开销来保证关键数据的可靠性。

2）提出一种响应式多目标进化算法 IMOEA 来解决所提出的问题。IMOEA 算法不仅可以获得兼顾执行时延与数据中心负载均衡的均匀且多样化的解，同时允许决策者将他们的偏好信息融入优化过程，从而在不探索整个搜索空间的情况下找到最满意的解决方案。即使决策者不时地改变他们的偏好，IMOEA 算法仍然能够快速地重新聚焦于感兴趣的新区域，可以潜在地节省大量的运行时间。

3）通过仿真实验验证，本文提出的基于 IMOEA 算法的数据布局策略要优于其他数据布局算法，且可以适应不同的云边环境。

2　相关工作

针对构建数据空间实现数据管理的问题，Xiao 等[4]提出了一种基于本体的数据访问框架，用于数据集成、存储、语义查询和知识推理；Li 等[5]提出了一种将分散的数据库与知识架构关联起来的数据空间构建方法；Wang 等[6]通过元数据提取完成对数据的统一描述，然后通过数据关系的构建和数据服务映射构建数据空间。上述研究主要聚焦于数据的统一描述、语义查询和知识推理，对数据空间中关键数据的可靠性保障和高效布局缺乏深入探讨。本文不仅致力于解决数据空间的构建问题，还重点关注其在云边环境中的应用场景，填补了现有研究中的重要空白。

目前，许多针对科学工作流的相关研究集中在优化传输

时延。Li 等[7]在云环境中提出了一种基于数据依赖破坏度的矩阵划分模型和面向数据中心的数据布局方法，在进行数据划分时尽可能地最小化破坏数据依赖度，使得同一数据中心上的数据依赖关系较高，以降低数据集跨数据中心传输时延。Cui 等[8]构建了一种基于任务、数据集和数据中心的三方图模型，提出了一种基于遗传算法的数据布局策略，同时将每个数据集在不同数据中心存储若干副本，以减少数据中心之间的数据传输量。Lin 等[9]结合云计算和边缘计算中的数据布局特点，考虑了数据中心之间的带宽、边缘数据中心的数量、边缘数据中心的存储容量等因素，将粒子群算法和遗传算法相结合以优化数据传输时延。Li 等[10]考虑工作流间的共享数据集，将多个工作流视为一个整体，构建了一种工作流级数据布局模型，并提出了一种两阶段的数据布局策略，使用离散粒子群优化算法来优化数据传输时延。Du 等[11]将差分进化算法与粒子群算法结合，在云边环境中考虑工作流间共享数据集和数据中心地理分布情况，结合云计算和边缘计算提出了一种考虑多工作流的数据布局策略以优化工作流执行时延。尽管上述研究在优化传输时延方面取得了显著成效，但大多忽略了数据中心负载均衡的问题。这种忽视可能导致某些数据中心因过载而性能下降甚至崩溃，而其他数据中心却处于闲置状态，资源得不到充分利用。

追求工作流的传输时延固然重要，但数据中心间的负载均衡同样不可忽视。Deng 等[12]在边缘环境中提出了一种多层图分区算法，将同一数据中心内的数据集进行合并，之后将图最小化切割后映射回原始工作流图，旨在最小化数据中心之间的数据传输量、优化数据中心负载均衡的同时满足固定数据约束。Zheng 等[13]对云环境下数据密集型应用的特点进行分析，提出了一种三阶段数据布局策略，同时对数据传输时延和数据中心间的负载均衡进行优化。Shang 等[14]提出了一种基于任务分配的数据布局策略，该方法通过遍历数据集，根据不同的副本建立条件来判断是否需要生成数据副本，以优化数据布局中的费用问题，同时尽可能兼顾负载均衡。Cheng 等[15]运用基于 KneePoints 的多目标优化算法来解决云环境下的数据布局问题，将固定数据集进行初始布局后，利用多目标优化算法对非固定数据集进行进一步布局，最终能取得兼顾数据传输时间和负载均衡的数据布局方案。Wei 等[16]将数据项和边缘数据中心映射到虚拟平面，并根据数据的流行度将数据布局到边缘数据中心，从而优化数据检索延迟和负载均衡。尽管上述研究在优化负载均衡方面取得了一定进展，但在数据可靠性方面存在明显不足。这些工作未充分考虑关键数据的容错需求，也缺少面向数据可靠性的备份机制，可能在数据中心发生故障时导致数据丢失或服务中断。

综上所述，目前关于数据布局的研究主要集中于云环境、边缘环境以及云边协同环境。这些研究大多关注如何缓解海量数据对数据中心的存储压力以及工作流间共享数据集的高效管理。然而，它们在数据可靠性方面的探讨较为薄弱，对于执行时延与数据中心负载均衡之间的权衡也缺乏深入研究。因此，从提升数据可靠性、多目标优化的角度进一步完善数据布局策略，仍具有重要的研究价值。此外，现有研究较少探索结合数据空间进行科学工作流的数据布局问题。针对这一不足，本文提出了一种将数据空间整合数据存储到云边环境的方法，从而实现数据布局优化，为科学工作流提供高效、可靠的支持。

3 系统模型

数据空间整合框架下的云边环境在基于纠删码的工作流数据布局架构如图 1 所示，主要由数据空间、工作流以及数据布局器 3 个部分组成。用户提交的工作流请求被提交到数据布局器，数据布局器根据指定的布局策略将各个工作流任务调度到不同的数据中心执行、将不同任务执行后产生的数据布局到不同的数据中心。为了防止某些关键数据丢失，数据中心将关键数据通过纠删码技术加工成若干编码块以保证数据可靠性，在需要使用数据时将若干编码块解码为原数据。

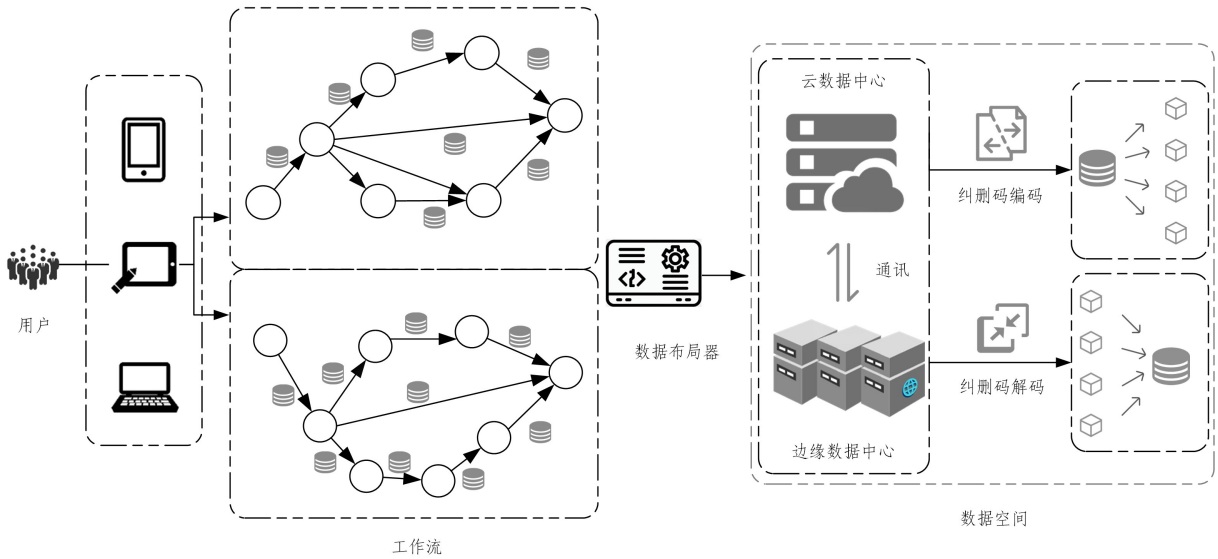


图 1 数据空间中基于纠删码的工作流数据布局架构

Fig. 1 Data plcament architecture of workflow based on erasure code in data space

3.1　云边环境模型

云边协同环境 $\boldsymbol{S}=\{\boldsymbol{S}_{\text{cld}},\boldsymbol{S}_{\text{edg}}\}$ 包含了远端的云平台和近端的边缘侧。其中云平台 $\boldsymbol{S}_{\text{cld}}=\{s_1,s_2,s_3,\cdots,s_j\}$ 包含了若干云数据中心，它们拥有近乎无限的存储空间；而边缘侧包含了若干边缘数据中心 $\boldsymbol{S}_{\text{edg}}=\{s_{j+1},s_{j+2},s_{j+3},\cdots,s_{j+k}\}$，它们存储空间较为紧张但更加靠近用户位置，可以缩短数据的传输时延。每个数据中心可以被表示成 $\langle c_i^{\text{total}},c_i^{\text{used}},\gamma_i,a_i^{\text{enc}},a_i^{\text{dec}}\rangle$。其中 c_i^{total} 表示数据中心的总存储容量，特别地，对于云数据中心不设置存储容量限制；c_i^{used} 表示数据中心在数据布局过程中已使用的存储容量；$\gamma_i\in\{0,1\}$ 表示数据中心类型，$\gamma_i=0$ 表示云数据中心，$\gamma_i=1$ 表示边缘数据中心。使用纠删码来保证数据可靠性，会涉及到对数据的编码和解码。纠删码编码指在原始数据块中添加冗余，将原始数据编码为若干编码块。编码块中包含若干数据块与校验块，使用 a_i^{enc} 表示数据中心对单位大小数据的纠删码编码速度。在需要使用原始数据时，将若干数据块和校验块放入解码器进行纠删码解码以进行数据恢复，使用 a_i^{dec} 表示数据中心对单位大小数据的纠删码解码速度。此外，数据中心间的网络带宽由式(1)定义：

$$\boldsymbol{B}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1|S|} \\ b_{21} & b_{22} & \cdots & b_{2|S|} \\ \vdots & \vdots & \cdots & \vdots \\ b_{|S|1} & b_{|S|2} & \cdots & b_{|S||S|} \end{bmatrix} \tag{1}$$

其中，b_{ij} 代表了数据中心 s_i 与数据中心 s_j 的带宽，本文假设网络带宽为常数。

3.2　工作流模型

工作流模型用 $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E},\boldsymbol{D})$ 有向无环图来表示，包含任务集 $\boldsymbol{V}$、任务依赖关系 $\boldsymbol{E}$ 与数据集 $\boldsymbol{D}$。工作流模型中的任务集用 $\boldsymbol{V}=\{v_1,v_2,\cdots,v_j\}$ 表示，使用 $\boldsymbol{D}_i^{\text{input}}$ 表示任务 v_i 的输入数据集，使用 $\boldsymbol{D}_i^{\text{output}}$ 表示任务 v_i 的输出数据集。工作流模型中任务依赖关系集合用 $\boldsymbol{E}\subseteq\boldsymbol{V}\times\boldsymbol{V}$ 表示，若存在 $e_{ij}\in\boldsymbol{E}$，则表示任务 v_i 与任务 v_j 存在依赖关系，任务 v_j 是任务 v_i 的后继任务，要在任务 v_i 完成后才能执行。使用式(2)表示工作流执行过程中的任务调度方案，任务调度方案 $\boldsymbol{M}$ 含有每个任务执行时所在的数据中心编号，使用 $\widetilde{M}(v_i)$ 表示 v_i 执行时所在的数据中心。

$$\boldsymbol{M}=\bigcup_{i=\{1,\cdots,|V|\}}\{(v_i,s_j)\} \tag{2}$$

$\boldsymbol{D}=\{\boldsymbol{d}_1,\boldsymbol{d}_2,\cdots,\boldsymbol{d}_k\}$ 代表数据集，数据集 $\boldsymbol{D}$ 是一系列数据编码块的集合。具体来说，数据 $\boldsymbol{d}_i$ 存放着第 i 号原始数据的信息。若第 i 号原始数据为关键数据，则 $\boldsymbol{d}_i$ 包含了该原始数据通过纠删码编码后的数据块和校验块；若第 i 号原始数据为普通数据，则 $\boldsymbol{d}_i$ 只存放原始数据本身。每个数据 $\boldsymbol{d}_i$ 含有属性 $\langle z_i,y_i,m_i,n_i\rangle$，其中 z_i 表示第 i 号原始数据的数据大小。使用 y_i 来标记数据的类型，$y_i=1$ 表示第 i 号原始数据的数据为关键数据，需要使用纠删码技术来保证该数据的可靠性；$y_i=0$ 则表示第 i 号原始数据的数据为普通数据，无须使用纠删码技术产生冗余数据。m_i 表示关键数据进行纠删码编码后的数据块数量，而 n_i 则表示校验块数量。需要注意的是，关键数据的每个编码块需要存储在不同的数据中心，以保证数据可靠性。若第 i 号原始数据为普通数据，则 m_i 与 n_i 均为 0，表示不进行纠删码编码。此外，若第 i 号原始数据为关键数据，由于纠删码会将数据编码为等大的 m_i 个数据块与 n_i 个同样大小的冗余校验块，则 $\boldsymbol{d}_i$ 内每个编码块的大小都为 z_i/m_i。式(3)表示工作流执行过程中的数据布局，数据布局方案 $\boldsymbol{P}$ 是数据与其存放数据中心组成二元组的集合，含有每一个数据或编码块的布局位置，使用 $\widetilde{\boldsymbol{P}}(d_{ij},s_k)$ 表示 d_{ij} 布局的数据中心。

$$\boldsymbol{P}=\bigcup_{i=\{1,\cdots,|\boldsymbol{D}|\},j=\{1,\cdots,|\boldsymbol{d}_i|\}}\{(d_{ij},s_k)\} \tag{3}$$

在工作流的执行过程中，需要保证关键数据的可靠性，防止其丢失。然而传统的数据副本方案会造成较高的冗余，使用纠删码技术能够以极低的存储成本提供高效和高可用的存储服务。图 2 展示了使用纠删码的工作流实例，在该实例中，每一个关键数据被拆分成 2 个数据块与 1 个校验块，只使用了 50％的额外开销，便实现了和两个副本冗余这种 100％额外开销相同的容错能力。

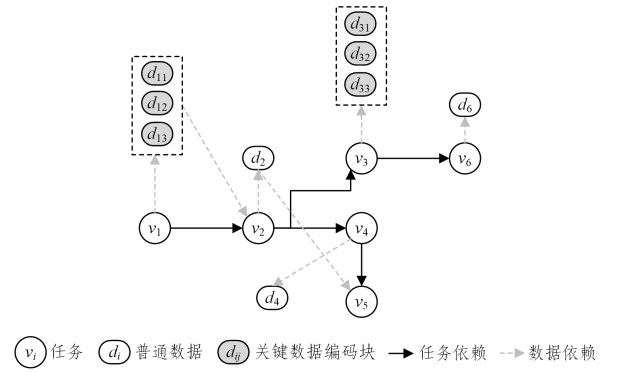


图 2　使用纠删码的工作流实例

Fig. 2　Example of workflow using erasure code

3.3　优化目标

工作流数据布局涉及用户以及服务提供商这两个不同的利益主体。用户希望在尽可能短的时间内完成工作流的执行，而服务提供商则希望在不同的数据中心间实现负载均衡，以提高系统的可用性。单一目标的优化往往不能很好地解决所有利益主体的需求，因此本节将工作流执行时延和数据中心负载均衡同时纳入优化目标。

3.3.1　工作流执行时延

由于科学工作流中的数据量巨大，任务调度时间远小于数据传输时间，因此忽略任务调度时间，主要考虑与数据相关的时间开销。工作流总执行时延 T_{total} 的定义如式(4)所示，工作流执行时延包含了总编码时延 T_{enc}、总解码时延 T_{dec} 和总传输时延 T_{tran}，其中 T_{enc} 与 T_{dec} 是纠删码的引入带来的额外时间开销。

$$T_{\text{total}}=T_{\text{enc}}+T_{\text{dec}}+T_{\text{tran}} \tag{4}$$

若数据 $\boldsymbol{d}_i$ 在数据中心 s_j 上进行纠删码编码，则它的编码时延如式(5)所示：

$$t_{\text{enc}}(\boldsymbol{d}_i,s_j)=z_i/a_j^{\text{enc}} \tag{5}$$

总编码时延如式(6)所示。统计工作流中的所有任务，遍历每个任务的输出数据集，其中每一个关键数据都会在它们

生成的时候进行纠删码编码，也就是在任务执行所在的数据中心进行编码，因此总编码时间就是它们的和。

$$T_{\text{enc}}=\sum_{i=1}^{|V|}\sum_{d_j\in \boldsymbol{D}_i^{\text{output}}} t_{\text{enc}}(\boldsymbol{d}_j,\widetilde{M}(v_i))\cdot y_j \tag{6}$$

类似地，若数据 $\boldsymbol{d}_i$ 在数据中心 s_j 上进行纠删码解码，则它的解码时延如式(7)所示：

$$t_{\text{enc}}(\boldsymbol{d}_i,s_j)=z_i/a_j^{\text{dec}} \tag{7}$$

总解码时延如式(8)所示，与总编码时延类似。不同的是，总解码时延需要遍历每个任务的输入数据集，在任务执行时若需要获取关键数据，需要将该关键数据的编码块从不同的数据中心传输到任务执行所在的数据中心以还原数据。

$$T_{\text{dec}}=\sum_{i=1}^{|V|}\sum_{d_j\in \boldsymbol{D}_i^{\text{input}}} t_{\text{dec}}(\boldsymbol{d}_j,M(v_i))\cdot y_j \tag{8}$$

数据 d_{ij} 从数据中心 s_k 传输到数据中心 s_l 的传输开销 t_{tran} 的定义如式(9)所示：

$$t_{\text{tran}}(d_{ij},s_k,s_l)=\begin{cases}\dfrac{z_i}{b_{kl}}, & y_i=0\\ i, & y_i=1\end{cases} \tag{9}$$

其中，若数据 d_{ij} 为普通数据，则使用原始数据大小来计算传输时延；若数据 d_{ij} 为关键数据，其表示一个编码块，则使用编码块大小来计算传输时延。

总传输时延如式(10)所示：

$$T_{\text{tran}}=\sum_{i=1}^{|V|}\sum_{j\neq i}^{|V|}\sum_{k=1}^{|D|}\sum_{l=1}^{|d_k|} t_{\text{tran}}(d_{kl},s_i,s_j)\cdot h(i,j,k,l) \tag{10}$$

其中，$h(i,j,k,l)\in\{0,1\}$，$h(i,j,k,l)=1$ 表示数据 k 的第 l 个编码块 d_{kl} 存在从数据中心 s_i 到数据中心 s_j 的传输，否则 $h(i,j,k,l)=0$。

3.3.2 数据中心负载均衡

服务提供商希望均衡数据中心间的负载，将流量分发到不同的数据中心上，从而减轻单个数据中心的压力，提高整体系统的性能和响应速度。因此将数据中心间的负载均衡作为本文的目标函数之一。文献[17]采用数据中心使用率的标准差来描述数据中心间的负载均衡，但数据中心的容量较大，数据集大小对于数据中心的容量来说比例很小，故计算结果并不准确。本文使用数据中心已使用容量的标准差 C 来评价数据中心之间的负载均衡，如式(11)所示：

$$C=\sqrt{\frac{1}{|\boldsymbol{S}|}\left(\sum_{i=1}^{|S|}(c_i-\bar{c})^2\right)} \tag{11}$$

其中，$\bar{c}$ 表示数据中心的平均已使用容量，如式(12)所示：

$$\bar{c}=\frac{1}{|\boldsymbol{S}|}\left(\sum_{i=1}^{|S|}c_i\right) \tag{12}$$

3.4 问题公式化

针对数据空间下基于纠删码的工作流多目标数据布局问题，本文旨在满足关键数据可靠性、数据中心存储容量限制的前提下，最小化工作流执行的总时延以及优化数据中心的负载均衡。基于以上定义，得到数据布局的总目标，如式(13)所示：

$$\begin{aligned}&\min\ T_{\text{total}},C\\ &\text{s.t.}\ \ \forall i\in|\boldsymbol{S}|,c_i^{\text{used}}\leqslant c_i^{\text{total}}\\ &\qquad\forall y_i=1,|\boldsymbol{d}_i|=m_i+n_i\end{aligned} \tag{13}$$

4 基于 IMOEA 的数据布局策略

本文提出的数据布局问题考虑了两个利益主体的不同目标，涉及工作流执行时延和数据中心负载均衡这两个互相冲突的目标。本章中提出了一种响应式多目标进化算法 IMOEA 来求解所提出的问题。该算法允许程序运行时将偏好信息整合到优化过程中，从而在不探索整个搜索空间的情况下找到决策者最喜欢的解决方案。此外，在运行时即使不断改变偏好，算法仍然能够快速地重新关注他们感兴趣的新区域，潜在地节省了大量的运行时间[18]。

4.1 问题编码

本文使用二维数组构建候选解粒子，粒子 i 的数据布局方案 $\boldsymbol{X}_i$ 如式(14)所示：

$$\boldsymbol{X}_i=(\boldsymbol{x}_{i1},\boldsymbol{x}_{i2},\cdots,\boldsymbol{x}_{i|\boldsymbol{D}|}) \tag{14}$$

每一位 $\boldsymbol{x}_{ij}$ 代表第 i 个粒子中数据 j 的存放位置，如式(15)所示：

$$\boldsymbol{x}_{ij}=(q_{ij1},q_{ij2},\cdots,q_{ij|S|}) \tag{15}$$

其中，$q_{ijk}\in\{0,1\}$，$q_{ijk}=1$ 表示在第 i 个粒子内，数据 j 在第 k 个数据中心存放着某个编码块或是原始数据，反之代表没有。若数据 j 为关键数据，则 $\boldsymbol{x}_{ij}$ 中 $q_{ijk}=1$ 的数量为编码块的数量，不同的编码块需要放在不同的数据中心上；若数据 j 为普通数据，则 $\boldsymbol{x}_{ij}$ 中 $q_{ijk}=1$ 只会出现一次，普通数据不进行编码，只会将原始数据保存在某个数据中心。图 3 给出了使用纠删码的二维编码粒子对应的数据布局方案，使用纠删码将关键数据编码为 2 个数据块与 1 个校验块，故每个关键数据需要将 3 个编码块布局在不同的数据中心上。

图 3 使用纠删码的二维编码粒子对应的数据布局

Fig. 3 Two-dimensional encoding particles corresponding to the data placement using erasure coding

4.2 适应度函数

本文的研究目的是优化工作流执行时延以及数据中心负载均衡，并充分考虑决策者偏好信息，得到最符合决策者需求的数据布局方案。但本文的编码不具备健全性，会产生不可行解粒子。导致不可行解的原因包括关键数据可靠性不达标和不满足数据中心容量约束。关键数据可靠性不达标指的是没有生成指定数量的数据块与校验块，未达到容错等级要求；

不满足容量约束表示至少有一个边缘数据中心存储的数据超出容量限制，使用非法数据集 $\boldsymbol{D}_{\text{inf}}$ 来描述导致粒子变为不可行解的数据集合。

对可行解和不可行解这两种类型粒子的适应度值 F 的比较，分为以下 3 种情况。

1）若比较的两个粒子都是不可行解，则认为 $\boldsymbol{D}_{\text{inf}}$ 长度更短的粒子适应度更好，代表更多的数据布局在可行的位置，这样的粒子更容易在后续的迭代中变为可行解粒子。

2）若将可行解粒子和不可行解粒子进行比较，则选择可行解。

3）若比较的两个粒子都是可行解，则使用单位向量 $\boldsymbol{\lambda}_i=(\lambda_i^1,\lambda_i^2)$ 来表示决策者的偏好信息。例如，$\boldsymbol{\lambda}_1=(\sqrt{2}/2,\sqrt{2}/2)$ 表示同等关注执行时延与负载均衡，$\boldsymbol{\lambda}_2=(1,0)$ 则表示只关心执行时延，完全不关心负载均衡。

当比较的两个粒子都是可行解时，使用式(16)来定义适应度函数。

$$F(\boldsymbol{X}_i,\lambda_i)=\max\left\{\frac{\lambda_i^1(T_{\text{total}}(\boldsymbol{X}_i)-g_1^1)}{g_1^2-g_1^1},\frac{\lambda_i^2(C(\boldsymbol{X}_i)-g_2^1)}{g_2^2-g_2^1}\right\} \tag{16}$$

其中，g_i^1 表示第 i 个优化方向的下界值，g_i^2 则表示第 i 个优化方向的上界值。该式可以在偏好信息的指导下考虑粒子在每个优化目标上的表现。

4.3　数据布局策略框架

传统的多目标优化旨在全面探索并获得整个 Pareto 前沿面，以提供多种权衡解供决策者选择[19]。而本文所研究的问题，其最终目标并非获取完整的 Pareto 前沿，而是聚焦于寻找最符合决策者偏好的最优解。在此情境下，通常要求决策者在优化执行前明确感兴趣的解空间区域。这里的困难在于，决策者不一定知道解决问题的局限性，可能抱有过于乐观的希望。如果决策者愿意参与优化过程并引导搜索迭代到他们最喜欢的解决方案，这个问题就可以完美解决。因此，可以首先为决策者提供一个大致的 Pareto 前沿面近似值，以便他们了解目标之间的权衡，并在此基础上进一步指定其偏好信息。这些信息将在下一阶段的优化中使用，以避免访问不希望访问的区域，而只探索更优选解决方案所在的区域。这个交互过程会一直持续下去，直到决策者找到最令他们满意的解决方案。鉴于上述考虑，开发了一个响应式多目标进化算法，称为 IMOEA。算法 1 展示了 IMOEA 的工作原理。该算法的流程图如图 4 所示。

算法 1　基于 IMOEA 算法的数据布局策略

输入：$\mathbf{G}=(\mathbf{V},\mathbf{E},\mathbf{D})$，$\mathbf{S}$

输出：$\mathbf{X}_{\text{final}}$

1. 对工作流 $\mathbf{G}$ 与数据中心 $\mathbf{S}$ 进行初始化
2. 根据纠删码方案，对关键数据进行编码
3. 随机生成 L 个粒子作为初始种群
   $\mathbf{layouts}=\{\mathbf{X}_1,\mathbf{X}_2,\cdots,\mathbf{X}_L\}$
4. 生成 L 个均匀的单位方向向量
   $\mathbf{vectors}=\{\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2,\cdots,\boldsymbol{\lambda}_L\}$
5. 计算种群的适应度 DataPlacement(**vectors**)
6. 初始化 L 个粒子的个体历史最优
7. while 决策者没找到满意的解决方案 $\mathbf{X}_{\text{final}}$ do
8. 　对每个 λ_i，将其在 vectors 中的 Nb 个近邻记为 **Nbs**
9. 　while 决策者没暂停算法 do
10. 　　for i=1 to i=L do
11. 　　从 $\boldsymbol{\lambda}_i$ 中随机选择一个索引交叉算子
12. 　　$\mathbf{X}_i$ 与个体历史最优执行交叉算子
13. 　　$\mathbf{X}_i$ 执行变异算子 Mutation($\mathbf{X}_i$)
14. 　　end for
15. 　　计算种群适应度 DataPlacement(**vectors**)
16. 　end while
17. 　计算近似的 Pareto 前沿面，决策者从中选择一个较为满意的点，其对应粒子为 X_*
18. 　取 λ_* 的左右边界，在其中生成均匀的 L 个单位方向向量组成新的 vectors
19. end while

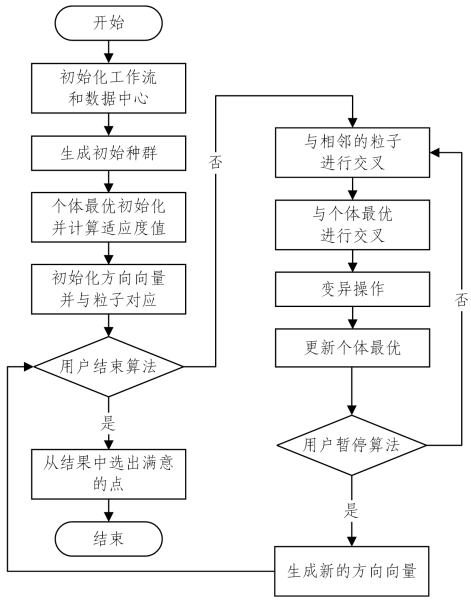


图 4　基于 IMOEA 的数据布局策略流程图

Fig. 4　Flowchart of data placement strategy based on IMOEA

在算法 1 中，首先进行初始化，包含解析工作流，并对其中的任务进行拓扑排序，将数据分类为普通数据和关键数据。初始化数据中心的容量，对最大存储容量进行设置(第 1 行)。根据纠删码方案，将关键数据编码为若干数据块与校验块，计算它们的数据大小(第 2 行)。生成等量的粒子与均匀单位方向向量，并将它们进行绑定，方向向量代表粒子专注于搜索的方向(第 3－4 行)。根据 *DataPlacement*( )计算种群的适应度，并将初始种群中的每个粒子设置为它们的个体历史最优(第 5－6 行)。

进行一次大循环，算法将在决策者找到满意的解时结束(第 7－19 行)。每次大循环开始时，对于每一个方向向量 $\boldsymbol{\lambda}_i$，都将与它们距离最近的 *Nbs* 个方向向量加入它们的邻居集合 ***Nbs*** 中(第 8 行)。进行一次小循环，在决策者暂停之前小循环会一直执行粒子的更新迭代(第 9－16 行)。对于种群中的每个粒子，从其单位向量的邻居集合 ***Nbs*** 中选一个对应的粒子进行交叉。与其方向相近的粒子进行交叉，更容易朝这个方向交叉出更好的粒子。吸收粒子群算法的思想，与个体历史最优粒子进行交叉，最后根据算法 2 执行变异算子 *Mutation*(第 10－14 行)。之后重新计算种群适应度，对种群进

行更新，若新的个体比原个体好，则更新个体历史最优(第 15 行)。

在决策者暂停算法结束小循环后，根据目前的粒子计算近似 Pareto 前沿面，决策者根据需求从前沿面中选择最满意的点作为参考点，在这个点所对应的方向向量附近生成新的方向向量 ***vectors***，下一次循环中种群就会朝着决策者选择的方向迭代，最终得到令决策者满意的数据布局方案。

图 5 展示了交互优化过程的示例。首先，IMOEA 在没有偏好信息的情况下运行几代，得到近似的 Pareto 前沿面(见图 5 中黄色圆圈)，决策者可以在其中选择自己喜欢的点，如点(0.25，0.55)。有了这个点，IMOEA 继续运行更多代，并产生围绕这个点的新的近似 Pareto 前沿面(如图 5 中的蓝色菱形)。此外，决策者还可以选择一个新的偏好点，如点(0.28，0.25)，IMOEA 将在此基础上执行更多代，以更加集中搜索。这个过程一直持续到决策者找到一个满意的 Pareto 前沿面(见图 5 中红色方块)。决策者从这个区域做出最终决定，即点(0.18，0.05)，并在实践中采用相应的数据布局方案。

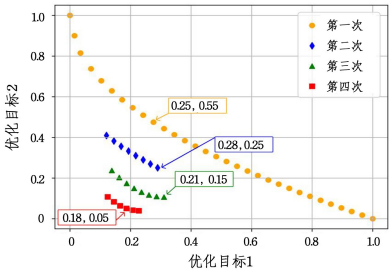


图 5 IMOEA 算法的交互式优化过程(电子版为彩图)

Fig. 5 Interactive optimization process of the IMOEA algorithm

4.4 粒子变异操作

变异时只改变某一个数据的布局位置，变异过程如算法 2 所示。图 6 展示了一个变异过程。

算法 2 变异函数 Mutation( )

输入：$\mathbf{X}_i$

输出：$\mathbf{X}_i$

1. 计算各个数据中心的已使用存储容量
2. if 所有数据中心均没有超出存储容量上限 then
3. 随机选择一个数据 muIndex 进行变异
4. else
5. 随机选择一个存放在超出容量限制数据中心上的数据 muIndex 进行变异
6. if muIndex 对应的数据为普通数据 then
7. 将 muIndex 位数据布局在其他数据中心上
8. else
9. 将 muIndex 位的编码块布局在其他数据中心上

在算法 2 中，首先计算各个数据中心在 $\boldsymbol{X}_i$ 对应的数据布局方案下的已使用存储容量情况，若该粒子为不可解粒子，数据中心超出存储容量上限，则优先选择导致存储容量超出上限的数据进行变异，这样更可能变异出可行解粒子(第 1－5 行)。若变异数据为普通数据，则只在某一个数据中心布局原始数据，变异时只需改变原始数据的位置；若数据为关键数据，则会在多个数据中心布局编码块，变异时需要将所有编码块同时变异，并注意变异前后不改变编码块数量。

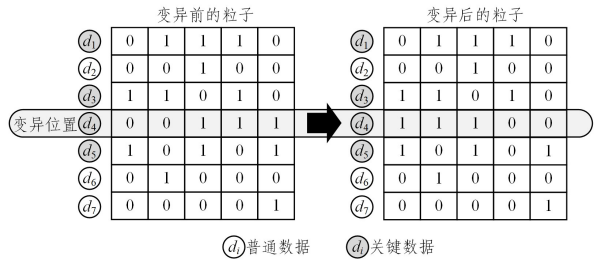


图 6 二维编码粒子的变异实例

Fig. 6 Example of mutation in two-dimensional encoding particles

4.5 数据布局过程

算法 3 给出了纠删码方案中数据的布局过程，从而计算粒子的适应度。对于可行解粒子，计算其总时延与数据中心负载均衡情况；对于不可解粒子，记录超出容量限制的数据中心。*DataPlacement*( )会对种群中所有粒子进行数据布局模拟，首先进行算法初始化，初始化总执行时延和数据中心的已存储容量，然后对初始数据进行布局，并更新数据中心 $\boldsymbol{C}_{\text{used}}$(第 2－4 行)。

计算工作流执行时的数据布局情况，根据任务的执行顺序遍历任务集 $\boldsymbol{V}$(第 5－20 行)，任务的调度位置会影响数据传输时延、数据编码时延和数据解码时延。尝试将任务放在不同数据中心执行，依次计算任务执行时延(第 6－18 行)。$T_{\text{taskPlace}}$ 记录了任务 task 在数据中心 taskPlace 执行时的时延，对于任务的每一个输入数据，都要计算其传输时延。若数据为普通数据，则将该数据传输到 taskPlace 上；若数据为关键数据，则分为若干编码块存储在不同的数据中心上，并按照需要的编码块数量，依次从带宽更高的数据中心传输即可(第 9 行)。关键数据分为若干编码块传输到数据中心 taskPlace 时，需要将它们解码为原始数据，存在解码时延(第 10 行)。对于任务的每一个输出数据，其处理逻辑与输入数据类似，不同的是，关键数据在生成时需要在数据中心进行编码，存在编码时延(第 13－17 行)。遍历任务可能调度的所有数据中心后，根据 $T_{\text{taskPlace}}$ 决定任务最终的调度位置(选择 $T_{\text{taskPlace}}$ 最小的数据中心，意味着任务在该数据中心执行时可以有最小的执行时延)，并更新 T_{total} 和 $\boldsymbol{C}_{\text{used}}$(第 19 行)。

在所有任务都执行完后，判断该粒子是否为可行解粒子。如果为不可解粒子，则记录导致不可解的数据中心(第 21－22 行)。若该粒子为可行解粒子，则根据数据中心已使用存储容量计算负载均衡指标。最后根据两个目标函数与方向向量计算适应度(第 24－25 行)。

算法 3 数据布局过程 DataPlacement(**vectors**)

输入：**vectors**

输出：**resultList**

1. for i=1 to i=L do
2. $T_{\text{total}} \leftarrow 0$
3. 初始化数据中心容量 $\mathbf{C}_{\text{used}}$
4. 将初始数据进行布局，更新 $\mathbf{C}_{\text{used}}$
5. for each task $\in \mathbf{V}$ do
6. for each taskPlace $\in \mathbf{S}$ do
7. $T_{\text{taskPlace}} \leftarrow 0$
8. for each **inputData** $\in \mathbf{D}_{\text{task}}^{\text{input}}$ do

```
9.            计算传输时延汇总至 T_taskPlace
10.           if inputData 是关键数据 then
11.             T_taskPlace += t_enc(outputData, taskPlace)
12.         end for
13.         for each outputData ∈ D_task^output do
14.           计算传输时延汇总至 T_taskPlace
15.           if outputData 是关键数据 then
16.             T_taskPlace += t_enc(outputData, taskPlace)
17.       end for
18.     end for
19.     根据任务 task 调度的位置，更新 T_total 和 C_used
20.   end for
21.   if C_used 超出存储容量上限 then
22.     记录不可解的数据中心
23.   else
24.     计算负载均衡指标 C
25.     计算适应度值
26. end for
```

5　实验与结果分析

5.1　实验设置

实验选用了 4 个代表性的不同领域的科学工作流数据集：天文学领域的 Montage、地震学领域的 CyberShake、生物科学领域的 Epigenomics，以及引力物理领域的 Inspiral[20]。

表 1 展示了本次实验的相关参数。本次的默认实验中，采取两个数据块与一个校验块的纠删码方案，最多能允许一个编码块丢失，达到与两个副本冗余方案近似的容错能力。决策者会在程序运行期间输入 3 次偏好信息，以指导迭代的方向，同时默认决策者在程序每进行 300 次迭代时输入偏好信息。变异和交叉概率参考了文献[18]，具体来说，基于当前粒子和个体历史最优粒子的差异度来调整惯性权重；与相邻粒子交叉的概率 α_1、与个体历史最优粒子交叉的概率 α_2 都使用线性变化策略。

表 1　实验相关参数

Table 1　Experimental related parameters

参数	值
初始种群大小	100
纠删码的数据块数量	2
纠删码的校验块数量	1
决策者偏好输入次数	3
决策者等待时迭代次数	300
关键数据比例	25%
纠删码编码速度	500 M/s
纠删码解码速度	250 M/s
Nb	15
$w^{\max}$	0.9
$w^{\min}$	0.4
αstart 1	0.9
αend 1	0.2
αstart2	0.4
αend 2	0.9

默认实验环境如下：云边环境下包含 4 个存储容量有限的边缘数据中心与 1 个存储容量近乎无限的云数据中心，数据中心的基准容量 c_{stand} 如式(17)所示，设置边缘数据中心的容量上限为 c_{stand} 的 1.4 倍，不同数据中心的带宽如式(18)所示，单位为 M/s。

$$c_{\text{stand}}=\frac{\sum_{i=1}^{|D|}z_i}{|\boldsymbol{S}|-1} \tag{17}$$

$$\boldsymbol{B}=\begin{bmatrix}0&10&20&30&40\\10&0&150&150&150\\20&150&0&150&150\\30&150&150&0&150\\40&150&150&150&0\end{bmatrix} \tag{18}$$

5.2　对比算法

5.2.1　评价指标

采用间距评价指标 SP[21] 和超体积 HV[22] 对算法性能进行评估，其中间距评价指标 SP 通过比较临近解之间的距离来衡量求解结果的分布性，SP 定义如式(19)所示：

$$SP=\sqrt{\frac{\sum_{i=1}^{|S|}(d_i-\bar{d})^2}{|\boldsymbol{S}|-1}} \tag{19}$$

其中，$|\boldsymbol{S}|$ 是算法求解的非支配解的数量，d_i 为最近解之间的曼哈顿距离。当非支配解均匀分布在整个 Pareto 前沿面上，SP 会变得较小，代表解集的分布性好。

超体积 HV 通过计算非支配解与参考点之间的超体积来衡量个体所支配的空间，HV 的定义如式(20)所示：

$$HV=volume\left(\bigcup_{i=1}^{|\boldsymbol{S}|}c^i\right) \tag{20}$$

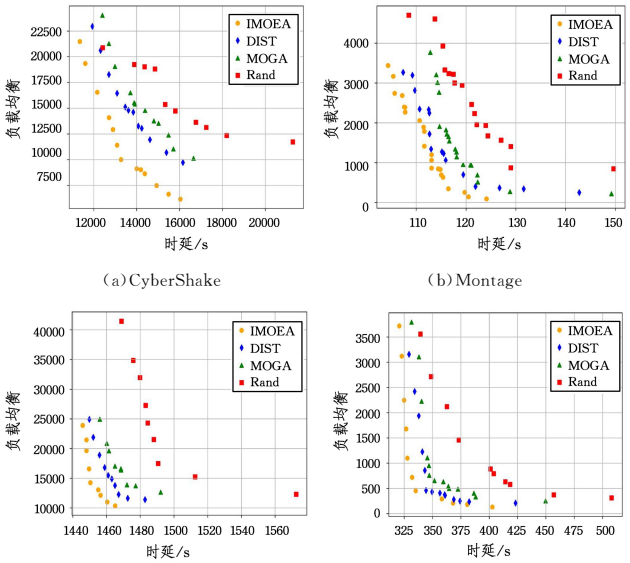
其中，c_i 是由非支配解与参考点作为对角线所围成的超立方体，当非支配解覆盖了较多目标空间时，HV 会变得较大，代表解集的多样性很好。对于本文所研究的二维问题，HV 就是非支配解与参考点所围成的体积。对两个优化目标进行归一化后计算 HV，同时将参考点设置为(1,1)。

5.2.2　实验结果与分析

本节将严格比较 IMOEA 算法和同样基于响应式的动态交互式安全与时间算法(DIST)[18]，以及一种基于多目标遗传算法(MOGA)[23] 和一种简单但高效的随机策略(Rand)，并调整了它们的部分实现，以适应云边环境下基于纠删码的数据布局策略。分别对 4 种不同类型的中型规模科学工作流进行实验，重复 20 组独立实验最终统计平均值，并将执行时延指标缩小 1000 倍作为最终结果展示，单位为秒。图 7 展示了实验结果。

从图 7 可以得出，IMOEA 算法的表现明显优于 DIST 算法、MOGA 算法和 Rand 算法。与另外 3 种算法相比，IMOEA 算法能够生成时延更低、负载均衡更优的数据布局方案。DIST 算法由于没有与种群最优个体进行交叉操作，并且变异概率是恒定的，因此其缺乏灵活性，无法根据搜索过程中个体的状态动态调整搜索策略，导致其在负载均衡优化方面的表现不尽如人意，尤其是在 CyberShake 工作流中的效果较差。而 MOGA 算法作为传统的多目标遗传算法，虽然能够处理多目标优化问题，但它在处理较为复杂的工作流时，难以生成时延较低的解，这表明其在全局搜索能力和局部搜索精度之间的平衡处理上存在不足。Rand 算法的随机性较强，没有针对性的优化策略，因此在所有工作流中的表现都不理想，无法适应不同类型的环境需求。相比之下，IMOEA 算法通过引入动态适应的变异概率和与历史最优个体交叉的“精英指导”策略，能够更高效地探索解空间，并加速算法向优

秀区域的收敛，从而在不同工作流下都能展现出较好的适应性和优化效果。同时从图 7 中也可以直观地看出，IMOEA 算法在每个方向上均能生成高质量的解，而 Rand 算法则在某些方向上显得过于稀疏。这种差异归因于 IMOEA 算法中均匀分布的单位方向向量，它们能够有效地引导粒子迭代，使每个粒子专注于不同方向的探索，保证了解集的均匀性。因此，IMOEA 算法展现出优异的解集分布性。从 HV 值来看，IMOEA 算法在 4 种类型工作流的 3 种不同规模下均取得了最高的 HV 值。



(a) CyberShake (b) Montage

(c) Epigenomics (d) Inspiral

图 7 4 种不同工作流下不同算法的实验结果

Fig. 7 Experimental results of different algorithms for four kinds of workflows

由图 7 可以直观地发现，IMOEA 算法生成的近似 Pareto 前沿更接近坐标轴。这说明 IMOEA 算法生成的解集不仅覆盖了更广泛的目标空间，还与真实 Pareto 前沿更加接近。这进一步表明 IMOEA 算法具有较强的全局搜索能力，能够在探索新解空间与开发已知优秀区域之间实现良好的平衡。这样的策略不仅能够避免陷入局部最优，其通过利用已知优秀解有效地指导搜索过程，还显著提升了算法的收敛速度和效率，能够高效地探索并发现优质的 Pareto 最优解。

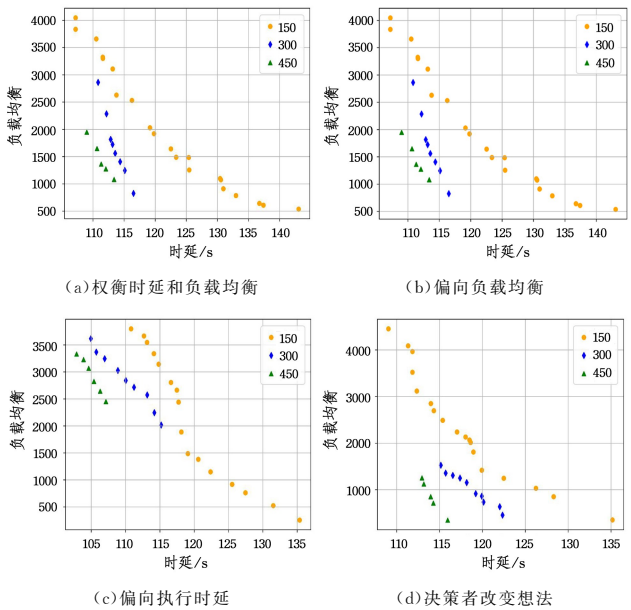
5.3 决策者偏好变化实验

为了验证 IMOEA 算法可以有效地关注决策者的偏好区域而不是整个搜索空间，使用中型规模的 Montage 工作流进行了 4 组实验，每组实验中决策者会展示不同的偏好。首先，IMOEA 算法迭代 150 代，由于决策者可能对问题没有先验知识，因此设置初始单位向量均匀地遍布整个第一象限。值得注意的是，在迭代结束时一个近似的 Pareto 前沿面将会被展示，决策者可以通过它了解目标之间的权衡。在实验中，决策者分别在 150 代、300 代和 450 代时逐步指定他们的偏好。

图 8(a)显示了决策者更喜欢时延和负载均衡同时兼顾的解决方案。在这种情况下，决策者总是选择得到的近似 Pareto 前沿面的中间解。IMOEA 算法在 150 代之后得到一个近似的 Pareto 前沿面(见黄色圆圈)。然后决策者选择接近中间的参考点。基于这一点，IMOEA 将其搜索空间缩小到一个新的近似 Pareto 前沿面(见蓝色菱形)，该近似值集中在选定的参考点周围。基于这个新生成的 PF 近似值，决策者继续指定自己的偏好点，并引导 IMOEA 进一步缩小搜索空间，从而更准确地生成决策者最满意解所在区域的近似值。

图 8(b)和图 8(c)分别展示了决策者选择更短执行时延、更优负载均衡的方案的情况。在图 8(b)中，近似 Pareto 前沿面逐渐向下方移动，这里的执行时延较短，负载均衡相对较差。相反，在图 8(c)中，近似 Pareto 前沿面逐渐向左边移动，因为决策者更喜欢负载均衡更优的解决方案。Pareto 前沿面之所以有这样的变化趋势，是因为一旦决策者确定了一个参考点，IMOEA 算法就可以有效地生成尽可能集中在符合决策者偏好的区域的 Pareto 前沿面。这种方式避免了花费同样的精力来生成整个 Pareto 前沿面，这无疑节省了大量的运行时间。

图 8(d)显示了决策者在执行时延和负载均衡之间摇摆，向相反方向改变偏好的情况。一开始，决策者选择了右下角的点，这意味着他们更喜欢具有负载均衡的解决方案。有了这个新的点，IMOEA 算法产生了更多这样的解，负载均衡更好(见图 8(d)中蓝色菱形)。从这里开始，决策者改变了想法，偏向于低执行时延的解决方案，而不是更优负载均衡的解决方案，因此他们选择了左上角的点，以进一步引导 IMOEA 算法向新的兴趣区域进化。可以看出，即使决策者指定不同的参考点指向相反的方向，IMOEA 算法也能够有效地使优化过程适应变化。



(a) 权衡时延和负载均衡 (b) 偏向负载均衡

(c) 偏向执行时延 (d) 决策者改变想法

图 8 IMOEA 算法对决策者偏好的适应实验(电子版为彩图)

Fig. 8 Experiment on the adaptation of user preferences by the IMOEA algorithm

5.4 数据中心容量变化实验

为了探究边缘数据中心存储容量对算法的影响，选取 Montage 的中型规模工作流进行实验，调整了边缘数据中心的容量，将它们设置为基准容量的倍数{1，1.2，1.4，2，2.6，5}。实验结果表明，容量设为基准容量 1.4 倍倍数的结果与设置 2 倍数以上基本重合，这表明当容量设置为基准容量的 1.4 倍时，边缘数据中心的存储容量就已经满足了数据布局

的需求，此时继续加大存储容量也不会导致结果变化。因此只绘制了存储容量为基准容量倍数的{1，1.2，1.4}倍的实验结果，如图 9 所示。

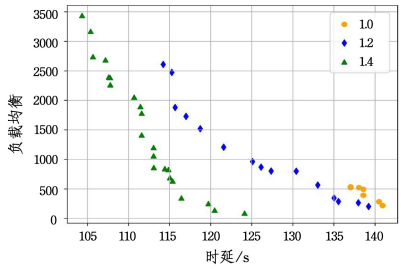


图 9　不同边缘数据中心存储容量下的实验结果

Fig. 9　Experimental results with varying edge datacenter capacities

当基准容量较小时，数据更多地被放在云数据中心上，导致传输时延大大增加，同时，由于数据中心存储容量的减少，将数据中心已使用存储容量的标准差作为负载均衡的指标，其自然会随着存储容量的减少而下降。当存储容量过低时，如数据中心的存储容量设为基准容量的 1 倍时，由于可行的数据布局方案过于有限，因此其近似 Pareto 前沿面上的点较少。

5.5　数据中心数量变化实验

为了观察数据中心容量变化对算法的影响，选取 Montage 的中型规模工作流进行实验，在其他条件不变时，仅改变边缘数据中心的数量，将它们分别设置为{4，6，8}，新增边缘数据中心之间的带宽设置为 150 M/s，云数据中心的带宽设置为 15 M/s。数据中心数量的变化会导致编码复杂性变化，数据中心更多则编码粒子会更加复杂、算法更难收敛，因此边缘数据中心数量为 4 时迭代共计 1000 次，数据中心数量为 6 时迭代共计 1500 次，数据中心数量为 8 时迭代共计 2000 次，在迭代时决策者不改变偏好信息。实验结果如图 10 所示。

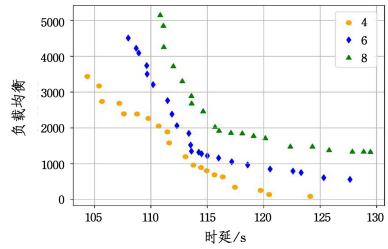


图 10　不同边缘数据中心数量下的实验结果

Fig. 10　Experimental results with varying numbers of edge datacenters

实验结果表明：1）随着数据中心增多，执行时延逐渐增加，在数据总量不变的情况下，增加数据中心数量会减少边缘数据中心的存储上限，导致数据要分配到更多不同的数据中心上，增加了数据的传输时延；2）随着数据中心增多，数据中心的负载均衡会增加，数据中心的数量增多会不可避免地导致某些数据中心闲置，有限的数据更难均分到更多的数据中心上。同时，IMOEA 算法在不同数据中心数量下都有稳定的表现，有较为均匀的 Pareto 前沿面。

5.6　关键数据比例变化实验

图 11 展示了 Montage 的中型规模工作流在不同关键数据比例下的实验结果。随着关键数据比例的增加，为更多数据提供可靠性保护的需求加大，纠删码的应用使数据被划分为多个数据块和校验块，这显著增加了编码和解码的计算开销，进而对工作流的执行时延造成一定影响。此外，虽然关键数据比例的增加提高了系统的冗余水平，给存储资源带来更大的压力，但由于纠删码在存储开销上的增长幅度相对较小，且数据在编码后被分解为更小的编码块，从而更有利于实现数据中心的负载均衡，因此系统在不同关键数据比例下仍能保持较好的负载均衡能力。

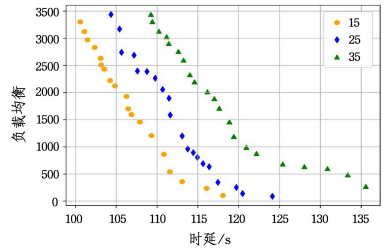


图 11　不同关键数据比例下的实验结果

Fig. 11　Experimental results with varying proportions of critical data

5.7　数据中心带宽变化实验

探究数据空间下带宽的变化对基于纠删码数据布局的影响，针对 Montage 的中型规模工作流，在默认实验设置的基础上，修改数据中心之间的带宽为默认带宽的倍数{0.5，1，1.5，3}。实验结果如图 12 所示，由于数据的传输时延受带宽的影响呈线性变化，而数据的编码解码时延虽不受带宽影响，但执行时延里传输时延占比较大。因此随着带宽的变化，总时延近似看作随着带宽增大而线性减小，同时带宽的变化不会影响数据中心负载均衡。

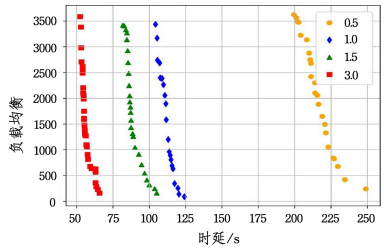


图 12　不同网络带宽下的实验结果

Fig. 12　Experimental results with varying network bandwidths

结束语　本文提出了一种数据空间中基于纠删码的数据布局模型，用于解决云边环境下考虑数据可靠性的科学工作流数据布局问题。同时，利用该模型，还提出了一种响应式多目标优化算法 IMOEA，以获得云边环境下的符合用户偏好的数据布局方案。该算法不仅在时延和负载均衡上有良好的均衡性和多样性，同时结合了来自决策者的偏好信息，潜在地节省了大量的运行时间。最后进行 3 部分实验：1）多目标优化算法的常见性能指标 SP 和 HV 表明，本文所提的策略与其他几种经典数据布局算法相比，表现出了更优秀的性能；2）决策者在程序运行时按照不同的偏好选择参考点，IMOEA 算法可以很好地根据决策者偏好选择进化方向，得到令决策者满意的数据布局方案；3）在默认实验设置的基础上改变实验配置，如边缘数据中心数量、边缘数据中心存储容量上限、网络带宽和关键数据的比例，实验结果表明，IMOEA 算法对不同的环境的都能表现出很好的适应性。

未来工作中，将考虑动态的网络环境，因为真实世界的网络带宽应是波动的，甚至可能会出现某些数据中心产生故障后下线的情况，一段时间内被视为可行的数据布局方案可能在发生变化后变得不可行，反之亦然。直观地说，对云边环境的变化做出反应的最简单方法是将每个更改视为必须从头开始解决的新问题。如果有足够的时间，这绝对是一个可行的选择。然而，在实际情况中，可用于重新优化的时间通常相当有限。因此，未来可以设计一种高效的动态算法，以适应云边环境的动态变化。

参 考 文 献

[1] LI J, LIN B, CHEN X. Reliability Constraint-oriented Workflow Scheduling Strategy in Cloud Environment[J]. Computer Science, 2023, 50(10): 291-298.

[2] FRANKLIN M, HALEVY A, MAIER D. From databases to dataspaces: a new abstraction for information management[J]. ACM Sigmod Record, 2005, 34(4): 27-33.

[3] LI J, LI B. Erasure coding for cloud storage systems: A survey [J]. Tsinghua Science and Technology, 2013, 18(3): 259-272.

[4] XIAO G, CALVANESE D, KONTCHAKOV R, et al. Ontology-based data access: A survey[C]// International Joint Conferences on Artificial Intelligence. 2018: 5511-5519.

[5] LI P, CHENG K, JIANG P, et al. Investigation on industrial dataspace for advanced machining workshops: enabling machining operations control with domain knowledge and application case studies[J]. Journal of Intelligent Manufacturing, 2022, 33: 103-119.

[6] WANG Y, CHENG Y, ZHU Y, et al. Exploration on industrial system-aware dataspace towards smart manufacturing[C]// 2022 IEEE 18th International Conference on Automation Science and Engineering(CASE). IEEE, 2022: 1883-1889.

[7] LI X J, WU Y, LIU X, et al. Datacenter-Oriented Data Placement Strategy of Workflows in Hybrid Cloud [J]. Journal of Software, 2015, 27(7): 1861-1875.

[8] CUI L, ZHANG J, YUE L, et al. A genetic algorithm based data replica placement strategy for scientific applications in clouds [J]. IEEE Transactions on Services Computing, 2015, 11(4): 727-739.

[9] LIN B, ZHU F, ZHANG J, et al. A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing[J]. IEEE Transactions on Industrial Informatics, 2019, 15(7): 4254-4265.

[10] LI X, ZHANG L, WU Y, et al. A novel workflow-level data placement strategy for data-sharing scientific cloud workflows [J]. IEEE Transactions on Services Computing, 2016, 12(3): 370-383.

[11] DU X, TANG S, LU Z, et al. A novel data placement strategy for data-sharing scientific workflows in heterogeneous edge-cloud computing environments[C]// 2020 IEEE International Conference on Web Services. IEEE, 2020: 498-507.

[12] DENG K, REN K, ZHU M, et al. A data and task co-scheduling algorithm for scientific cloud workflows[J]. IEEE Transactions on Cloud Computing, 2015, 8(2): 349-362.

[13] ZHENG P, CUI L Z, WANG H Y, et al. A Data Placement Strategy for Data-Intensive Applications in Cloud[J]. Chinese Journal of Computers, 2010, 33(8): 1472-1480.

[14] SHANG L, LIU X. Scientific Workflow Dataset Layout Based on Task Assignment and Dataset Replicas [J]. Computer Engineering, 2020, 46(5): 122-130.

[15] CHENG H, LI X, WU Y, et al. A multi-objective optimization-based data placement strategy for scientific workflows in cloud environment[J]. Computer Applications and Software, 2017, 34(3): 1-6.

[16] WEI X, WANG Y. Popularity-based data placement with load balancing in edge computing[J]. IEEE Transactions on Cloud Computing, 2021, 11(1): 397-411.

[17] DENG K, REN K, SONG J, et al. A Clustering based Coscheduling Strategy for Efficient Scientific Workflow Execution in Cloud Computing[J]. Concurrency and Computation: Practice and Experience, 2013, 25(18): 2523-2539.

[18] WANG X, VEERAVALLI B, SONG J, et al. On the Design and Evaluation of an Optimal Security-and-Time Cognizant Data Placement for Dynamic Fog Environments[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 34(2): 489-500.

[19] HUANG Z Q, LIN B, LU Y, et al. Site Selection and Capacity Determination Method for Charging Stations Oriented to Multi-objective Optimization [J]. Journal of Fujian Normal University (Natural Science Edition), 2024, 40(2): 23-35.

[20] BHARATHI S, CHERVENAK A, DEELMAN E, et al. Characterization of scientific workflows[C]// 2008 Third Workshop on Workflows in Support of Large-scale Science. IEEE, 2008: 1-10.

[21] SCHOTT J R. Fault tolerant design using single and multicriteria genetic algorithm optimization[D]. Massachusetts: Massachusetts Institute of Technology, 1995.

[22] ZITZLER E, THIELE L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach[J]. IEEE transactions on Evolutionary Computation, 1999, 3(4): 257-271.

[23] ZHANG M, REN H, XIA C. A dynamic placement policy of virtual machine based on MOGA in cloud environment[C]// 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications. IEEE, 2017: 885-891.

LIN Bing, born in 1986, Ph.D, associate professor, postgraduate supervisor, is a member of CCF (No. 83773M). His main research interests include cloud computing technology and computational intelligence.

JIANG Haiou, born in 1987, Ph.D, associate researcher. Her main research interests include cloud computing technology, big data and service computing.

（责任编辑：何杨）