面向边缘智能的边缘存储调度算法研究

Research on Edge Storage Scheduling Algorithm for Edge Intelligence

杜　量[1,2],谭　跃[3],邓　玲[1,2],曾楚轩[1,2],曹　畅[3],杨　杰[1,2](1. 联通智能制造科技产业〔广东〕有限公司,广东 深圳 518102;2. 联通粤港澳大湾区创新研究院,广东 深圳 518102;3. 中国联通研究院,北京 100048)

Du Liang[1,2],Tan Yue[3],Deng Ling[1,2],Zeng Chuxuan[1,2],Cao Chang[3],Yang Jie[1,2](1. China Unicom Smart Manufacturing Technology Industry〔Guangdong〕Ltd., Co.,Shenzhen 518102,China;2. China Unicom Guangdong-Hong Kong-Macao Greater Bay Area Innovation Research Institute,Shenzhen 518102,China;3. China Unicom Research Institute,Beijing 100048,China)

摘　要:

随着边缘设备数量激增,边缘存储作为边缘智能的基础设施面临数据规模指数级扩张、长期运营下多边缘数据中心间资源利用率失衡和读写延迟波动较大的挑战。针对该问题,提出了基于最优传输的MF-Sinkhorn算法,该算法构建融合节点资源状态、服务性能和请求距离的多重因子成本矩阵,通过Sinkhorn迭代动态优化存储节点选择,实现全局资源最优匹配。实验表明,相比传统最近邻策略,该方法使存储均衡度提升2.91%,写入时延降低6.01%,有效解决了多边缘数据中心资源均衡性问题,为边缘智能场景提供了高效存储调度方案。

关键词:

边缘存储;调度算法;多重因子成本矩阵

doi:10.12045/j.issn.1007-3043.2026.08.013

文章编号:1007-3043(2026)08-0072-07

中图分类号:TN919

文献标识码:A

开放科学(资源服务)标识码(OSID):

Abstract:

With the proliferation of edge devices, edge storage—as critical infrastructure for edge intelligence, faces challenges including exponential data growth, imbalanced resource utilization among edge data centers, and significant fluctuations in read/write latency during prolonged operation. To address these issues, an MF-Sinkhorn algorithm based on optimal transport theory is proposed. The method constructs a multi-factor cost matrix integrating node resource status, service performance, and request distance, then dynamically optimizes storage node selection through Sinkhorn iteration to achieve global optimal resource matching. Experiments demonstrate that compared with traditional nearest-neighbor strategies, this approach improves storage balance by 2.91% and reduces average write latency by 6.01%, effectively resolving resource imbalance in multi-edge data centers and providing an efficient storage scheduling solution for edge intelligence scenarios.

Keywords:

Edge storage;Scheduling algorithm;Multi-factor cost matrix

引用格式:杜量,谭跃,邓玲,等. 面向边缘智能的边缘存储调度算法研究[J]. 邮电设计技术,2026(8):72-78.

0 引言

近年来,AI大模型的快速发展催生了边缘侧本地化训练与推理的海量需求,使边缘智能成为研究热点[1-2]。边缘智能高度依赖边缘计算的实时处理能力,而边缘存储作为边缘计算的关键基础设施则面临数据规模急剧增长带来的容量、性能和能效三重挑战。现有研究将边缘存储根据功能定位划分为3类架构模式[3]:数据缓冲型、临时数据处理型和本地自治型。目前,针对第3种在边缘节点基于联邦学习等算法进行边缘本地提效的研究相对成熟(如边缘智能的联邦学习算法[4-6]、Wang等人设计的深度强化学习算法[7]),但仍有其局限性。Zhou等人认为边缘智能的范围不应

基金项目:国家重点研发计划(2022YFF0610300)

收稿日期:2026-07-06

该仅局限于边—端上运行AI算法,也应该包括在边—云上运行AI算法[8]。Ebrahimzadeh等人认为随着智能移动设备数量的不断增加及其对资源需求的日益增长,仅依赖边缘层资源来满足智能设备的业务需求将变得越来越具有挑战性[9]。针对边缘节点层算力不足的情况,亟需构建跨域资源协同机制路径,实现边缘节点与多边缘数据中心动态资源调度与协同管理机制。在国家政策与企业生产需求的推动下,已建设了大量的边缘数据中心,这为边缘存储突破传统就近调度模式、构建全局资源优化体系提供了基础设施支撑。对象存储技术[10]作为AI训练和大规模数据分析的核心基础设施,其Bucket创建操作将永久决定数据物理存放位置,其存储效率直接影响边缘智能的存储效能。

本文从系统架构设计和核心算法创新2个维度展开研究。在架构设计层面,创新性地构建了支持动态资源映射的协同调度框架,有效解决传统边缘存储架构中最近邻固定映射机制导致的资源利用率低下的问题;在算法创新层面,将多边缘节点与多边缘数据中心的Bucket创建映射问题转变为高效求解最优传输问题(Optimal Transport,OT)[11],通过将多维度指标融入Sinkhorn算法[12],提出一种基于多重因子的Sinkhorn算法(Multi-Factor Sinkhorn,MF-Sinkhorn),首次实现了Sinkhorn算法在多边缘数据中心存储调度场景下的适配优化,有效提升边缘存储架构的整体性能,实现边缘数据中心的负载均衡优化。

1 边缘存储架构

1.1 边缘存储业务架构

目前,产业界与学术界对边缘存储架构还未形成统一定义[13]。基于边缘节点层算力不足时需要将部分数据发送到边缘数据中心的前提[3],并基于部分云企的建设实践,提出中心云层、边缘数据中心层、边缘设备节点层三级分层架构(见图1)。其中,最上层为EB级中心云,部署于核心城市群,兼具海量数据存储与全域(包含中心云节点、边缘数据中心节点、边缘设备节点)监控功能和控制功能。中间层为区域级TB级边缘数据中心(EDC),可依据业务需求建设在蜂窝基站周边或城市近郊区域,建设间距在几百米到百公里不等。底层是终端边缘设备节点层,包含智能终端、IoT设备等海量节点。

1.2 边缘存储调度架构

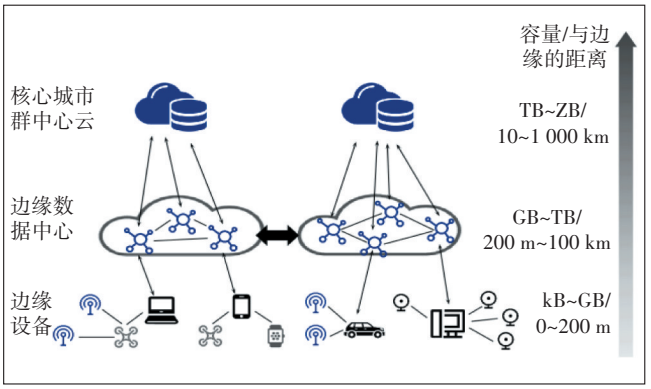

图1　边缘存储业务架构

在当前产业实践中,边缘设备节点(ED)入网时普遍采用静态映射机制与边缘数据中心(EDC)建立固定关联(见图1)。长期运行数据表明,这种静态分配模式会导致显著的资源不均衡现象[14]。

a) 存储容量失衡:部分EDC因空间较小,持续写入导致磁盘接近写满,随时触发写入保护机制。

b) 负载使用不均:部分EDC承接数据读写任务少,长期处于低负载状态。

c) 资源配置错配:高性能EDC节点未能有效承接近邻区域的突发流量,性能资源闲置。

边缘存储调度架构如图2所示,该架构是在一个核心城市群内工作的三级动态边缘存储调度架构,其创新性体现在分层协同机制。

a) 中心云管控制层。可选用工业级消息队列Kafka实现EDC资源状态的收集与全网广播,并可通过Paxos算法选举主EDC。选举过程由中心云触发,各EDC节点根据实时资源评分参与投票。

b) 边缘数据中心层。主EDC节点接收本城市群内多ED的实时Bucket创建请求,结合缓存的各EDC资源状态信息执行调度算法。经过调度算法计算,ED

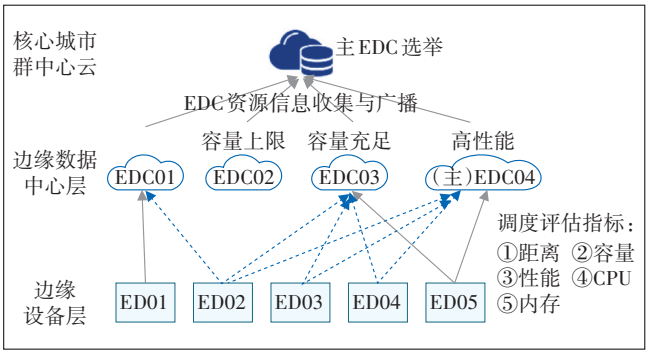

图2　边缘存储调度架构

与EDC之间不再是传统架构中的多对一映射,而是变成多对多映射,一个ED上的数据可根据调度算法的计算按需发往多个EDC进行存储。

2 调度算法

2.1 ED与EDC间调度问题建模

为实现ED与EDC间的资源调度优化,在对象存储系统的Bucket创建阶段实施智能调控策略。将多个ED节点的Bucket创建请求位置分布α建模为$\{x_1,\cdots,x_n\}$,将多个EDC位置分布β建模为$\{y_1,\cdots,y_m\}$,每一个ED节点请求x到每一个EDC请求y之间都有映射成本t,则问题转化为求解最优的映射$T:\{x_1,\cdots,x_n\}\rightarrow\{y_1,\cdots,y_m\}$,使得整体调度成本最小。其中,映射成本由多维指标共同决定,包括但不限于网络传输时延(距离指标)、存储性能、存储空间利用率、内存占用率、网络带宽利用率以及CPU利用率等[15]。这些指标综合反映了单次请求的实际开销,成本越低则调度效益越显著。因此,问题转变为构建一个决策算法,以实时处理多源并发请求,将各ED节点的Bucket创建操作智能分配到最优的EDC节点,最终实现全局映射成本最小与系统综合效益最大化。

上述的调度问题可以形式化为Kantorovich[16]最优传输问题的离散变体,简化描述为"多x→多y"匹配问题,设离散概率分布向量$\alpha\in R^n$和$\beta\in R^m$分别表示ED请求节点和EDC节点资源分布,给定成本矩阵$C\in R^{n\times m}$表征各节点间的传输开销,则最优调度问题转化为求解从α分布到β分布的传输矩阵$P\in R^{n\times m}$,使得在满足边缘约束条件的前提下,全局传输成本最小[17],用公式表示如下:

$$\min_{P\in U(\alpha,\beta)}\langle C,P\rangle\overset{\text{def}}{=}\sum_{i,j}C_{i,j}P_{i,j} \tag{1}$$

$$U(\alpha,\beta)=\left\{P\in R_+^{n\times m}\middle|P1_m=\alpha,P^T1_n=\beta\right\} \tag{2}$$

2.2 MF-Sinkhorn算法

熵正则化Sinkhorn算法[18]通过引入正则项,将经典最优传输(OT)问题转化为在平滑可行域上的求解过程[19],极大地简化了计算流程,在保证解的质量的前提下实现计算效率的指数级提升。在此基础上,MF-Sinkhorn算法在3个方面进行了创新。首先,在构建成本矩阵时,除了距离因子,还引入了多EDC资源状态参数。其次,设计加权融合机制将异构指标统一量化为传输成本。最后,通过正则化系数调整实现存储调度场景的精准适配。这种多维因子整合方法突破了传统OT问题仅考虑距离因子的局限,在保持时间复杂度的同时,能进一步提升存储调度效能。

下文将详细阐述MF-Sinkhorn算法。以边缘数据中心A为例,对其若干资源属性集合做出如下定义:

$$A=[a_1,a_2,a_3,a_4,a_5] \tag{3}$$

其中,$a_i(i=1,2,3,4,5)$分别为该EDC的性能排序、磁盘利用率、内存利用率、带宽利用率和CPU利用率指标。

为方便后续计算,采用离散标准化依次对每个EDC向量进行整理,得到:

$$\hat{a}_i=\frac{a_i-a_{\min}}{a_{\max}-a_{\min}} \tag{4}$$

采用加权平均对多指标进行整合计算,第i个ED节点到第j个EDC节点的多重因子成本矩阵$C_{i,j}$可表示为:

$$C_{i,j}=\hat{A}_j\hat{w}+d_{i,j}w^d \tag{5}$$

其中,$\hat{A}_j$为编号为j的EDC参数向量,$\hat{w}$为EDC一系列参数的权重向量,$d_{i,j}$为编号为i的ED节点到编号为j的EDC节点距离经过离散标准化处理后的结果,w^d为距离权重数值。且EDC各参数权重和距离权重满足:

$$\sum_w w_i+w^d=1 \tag{6}$$

经过上述优化过程,已构建出多边缘数据中心存储调度场景下的多重因子成本矩阵。下一步需先将存储调度中的Kantorovich问题模型转化为熵正则化的Sinkhorn算法求解模型,再将多重因子成本矩阵代入Sinkhorn迭代算法进行进一步求解。

熵正则化Sinkhorn算法求解方法为在式(1)给出的目标函数中,引入一个熵正则化项$D(P)$来简化最优传输问题。正则化的效果是使原本非常复杂和计算密集的问题变得更易于求解,同时提高数值稳定性。Kantorovich问题转变为:

$$\min_{P\in U(\alpha,\beta)}\langle C,P\rangle+\varepsilon D(P) \tag{7}$$

其中,$\varepsilon>0$表示收敛阈值,是一个较小的常数,用来控制正则化项的权重;$D(P)$为熵正则项,其表达式为:

$$D(P)=\sum_{i,j}P_{i,j}\log P_{i,j}-P_{i,j} \tag{8}$$

使用拉格朗日乘子法将原始Kantorovich问题转化为对偶形式进行求解(具体推导参见文献[20]),传输矩阵可表示为:

$$P_{i,j}=u_iK_{i,j}v_j \tag{9}$$

$$K_{i,j} = e^{-C_{i,j}/\varepsilon},\ u_i = e^{-f_i/\varepsilon},\ v_i = e^{-g_j/\varepsilon} \tag{10}$$

其中，$f \in R^n$和$g \in R^m$为引入对偶变量。最终传输矩阵为：

$$P = \mathrm{diag}(u)K\mathrm{diag}(v) \tag{11}$$

将式(5)中求解出的多重因子成本矩阵$C_{i,j}$带入式(10)，并迭代更新u和v，当迭代达到一定次数或者解的变化量小于阈值ε时，求解得到的P即为目标传输矩阵。

图3所示为上述求解过程的算法伪代码。在初始阶段，算法首先对所有EDC节点的资源向量进行离散化标准化处理，通过Min-Max归一化方式使各维度特征值映射到[0,1]区间，消除不同量纲对计算结果的影响。基于标准化后的数据，算法综合计算边缘请求节点与边缘数据中心之间的距离指标与资源指标构建多重因子成本矩阵，矩阵中每个元素值反映特定ED节点到EDC节点的综合调度成本。完成多重因子成本矩阵的构建后，算法通过熵正则化对成本矩阵进行平滑处理。熵正则化旨在增强计算过程的稳定性，并加速收敛，尤其在处理大规模数据时能够显著提高计算效率。随后，算法进入迭代阶段，这一阶段是Sinkhorn算法的核心。每轮迭代中，算法会分别对ED节点请求分布和EDC节点资源分布向量进行归一化，确保请求的总量与需求匹配，且EDC节点资源分配符合其实际能力。归一化操作分2步进行：首先，对ED节点请求分布进行归一化，确保所有请求的总和等于预设的资源总量；然后，对EDC资源分布进行列归一化，使每个EDC的资源需求得到合理满足。通过多轮迭代调整权重，直到ED请求分布和EDC资源分布趋于稳定。最终，当满足收敛条件时，算法输出一个最优调度矩阵，该矩阵表示了每个ED节点请求应分配到的每个EDC节点数量的占比。

Algorithm　MF-Sinkhorn algorithm $L_C^{\varepsilon}(a,\ b)$

Input：A(EDC综合评价参数向量)，α(ED节点请求向量，行之和)，β(目标分布向量，列之和)，f，g(正则化参数)，ε(收敛阈值)
Output：P(运输矩阵)

1　计算评价参数离散标准化 $\hat{a}_i = \dfrac{a_i - a_{\min}}{a_{\max} - a_{\min}}$
2　计算多重因子成本矩阵 $C_{i,j} = \hat{A}_j w + d_{i,j} w^d$
3　计算多重因子指数成本矩阵 $K_{i,j} = e^{-C_{i,j}/\varepsilon}$
4　初始化：$u = 1_{n_a},\ v = 1_{n_B}$
5　while Sinkhorn not converged do
6　$u \leftarrow \dfrac{\alpha}{Kv}$
7　$v \leftarrow \dfrac{\beta}{K^T v}$
8　end while
9　return P $\leftarrow P = \mathrm{diag}(u)K\mathrm{diag}(v)$

图3　算法伪代码

3 仿真与结果分析

3.1 评估指标与仿真环境

评估多边缘数据中心存储调度算法时，核心维度是容量均衡性和性能指标。理想的调度策略需同时优化存储空间利用率并最小化处理延迟。为此，设计了2项评估指标，分别为多EDC空间利用率均衡指标σ和多EDC写入性能指标L。下文将详细说明其计算方法。

指标1：多EDC空间利用率均衡性指标σ。U_n为每个EDC持续创建一定数量的Bucket后，随文件对象(Object)写入一段时间后的空间利用率，其计算如下：

$$U_n = \frac{CU_o + W}{C} \tag{12}$$

其中，C为EDC总容量，U_o为调度前EDC空间利用率数值，W为这一段时间内的数据写入量。基于上述定义，用于衡量多EDC空间利用率均衡性的标准差σ的计算公式为：

$$\sigma(U) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(U_i - \bar{U}\right)^2} \tag{13}$$

其中，U_i为每个EDC的空间利用率，$\bar{U}$为多EDC平均空间利用率。

指标2：多EDC综合写入性能指标L。该指标可用多EDC在一段时间内文件写入的平均时延情况来反映，计算如下：

$$L(l) = \frac{1}{n}\sum_{i=1}^{n}\bar{l}_i \tag{14}$$

其中，$\bar{l}_i$为第i个EDC一段时间内写入请求的平均时延，$L(l)$为多EDC综合写入平均时延。

在进行仿真实验前，需要设定一系列初始条件，包括ED节点与EDC节点分布位置、EDC资源状态情况、比例尺定义、时延估算方法等。接下来将对这些关键预设条件进行简单说明。

实验在一个300×450单位的长方形坐标系内(比例尺约为1∶1 km)，随机选取100个点代表不同地理位置的ED节点分布，另外选取10个点代表不同地理位置的EDC分布。每个ED节点发起10次Bucket创建请求，每个Bucket内写入10 GB的数据(三副本方式落盘)，构成总体请求量。

实验构建了模拟EDC异构特性的多维资源数据集,分别为性能、磁盘、容量、内存、带宽以及CPU,基本资源情况如表1所示。构建成本矩阵时,各资源的权重预设如下:距离为40%,性能与容量各为20%,磁盘、带宽、CPU各为6%,内存为2%。同时,为评估调度算法对写入时延的影响,建立了多EDC时延计算模型,该模型包含2个核心因素:一是地理距离时延,可按照1 ms/km模拟光缆传输基础时延;二是EDC性能差异时延影响,可参考表1(文件大小为1 MB)根据EDC资源状态预设不同时延。多EDC环境下的综合写入平均时延则可综合以上2个时延数值计算得到。

表1　各边缘数据中心资源与时延

编号	性能	磁盘	容量	内存	带宽	CPU	时延
1	2	0.2	0.1	0.2	0.2	0.2	50
2	2	0.3	0.2	0.2	0.2	0.2	50
3	1	0.3	0.7	0.1	0.1	0.1	15
4	4	0.6	0.3	0.3	0.4	0.3	100
5	4	0.6	0.5	0.3	0.4	0.3	100
6	4	0.6	0.5	0.4	0.4	0.4	100
7	4	0.7	0.5	0.4	0.4	0.4	150
8	8	0.8	0.8	0.5	0.6	0.5	100
9	9	0.8	0.8	0.5	0.8	0.5	200
10	9	0.9	0.9	0.5	0.9	0.5	200

3.2 仿真结果分析

为验证MF-Sinkhorn算法的效果,从100个ED节点随机选取5个进行可视化分析。如图4所示,在300×450单位内模拟某一地理范围,蓝色标记为ED节点,红色为EDC节点,黑色连线表示请求分配关系,线条粗细反映了请求量级。调度结果显示,部分请求仍被分配至最近邻EDC节点;同时算法基于资源利用率、性能等指标,智能将部分请求分配至次近邻或更远节点,甚至采用多节点协同服务,这体现了算法的灵活性。

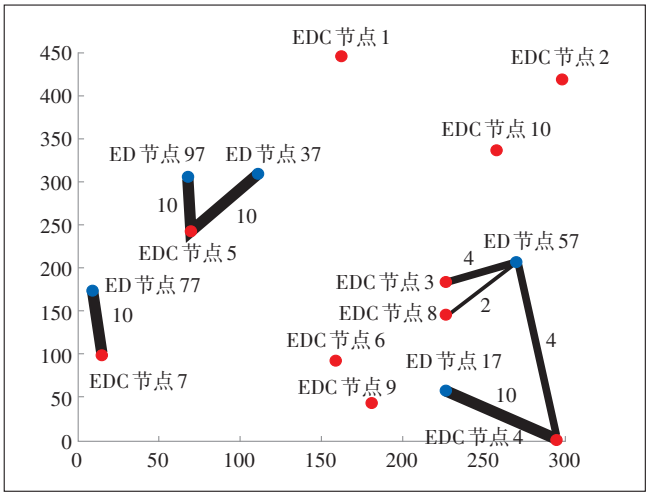

图4　边缘节点到边缘数据中心的调度示例

调度算法中正则系数ε的选取直接影响迭代次数,从而影响方案时延。如图5所示,增大正则系数ε能有效降低迭代次数,缩短计算耗时,但会增强信息熵效应,这与多EDC的调度需统筹多因子成本矩阵参数的需求相悖。通过对比分析,采用折中的正则系数值ε=0.002,此时的计算时延为0.319 ms,这对于ED节点创建Bucket操作来说基本无感。

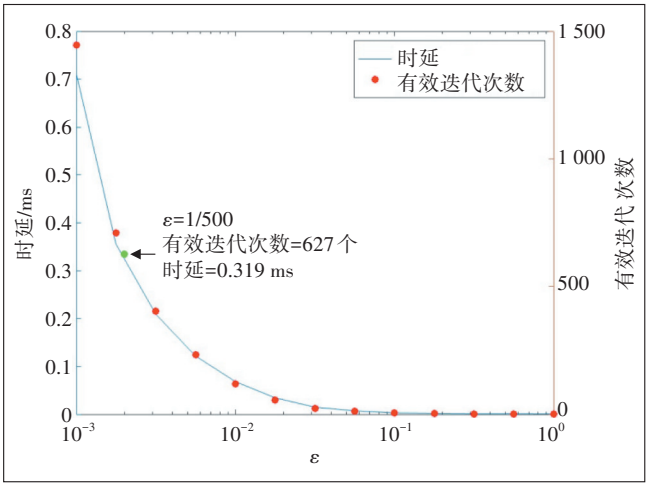

图5　不同ε下有效迭代时延和迭代次数趋势

基于前文定义的多EDC空间利用率均衡指标σ和多EDC写入性能指标L展开调度算法效果评估分析。

图6展示了3种调度策略(MF-Sinkhorn算法、原始Sinkhorn算法及生产环境常用的最近邻策略)在持续写入场景下对EDC空间利用率的调度效果。结果表明,改进算法使得数据写入更加均衡,在高负载EDC节点3、8、9、10的新增写入量显著降低,同时低利用率EDC节点1、2、4承担了更多的数据写入。多EDC节点在调度后利用率数据的标准差可以反映节点间均衡性,标准差越小则多EDC的综合空间利用率越高,可减少扩容等运维操作频次。如图7所示,相较于原始Sinkhorn算法,MF-Sinkhorn算法标准差下降2.78%;与最近邻策略相比,标准差降低了2.91%。实验证明,MF-Sinkhorn算法通过智能化的负载均衡机制,既避免了资源闲置,又提升了整体的系统效能。

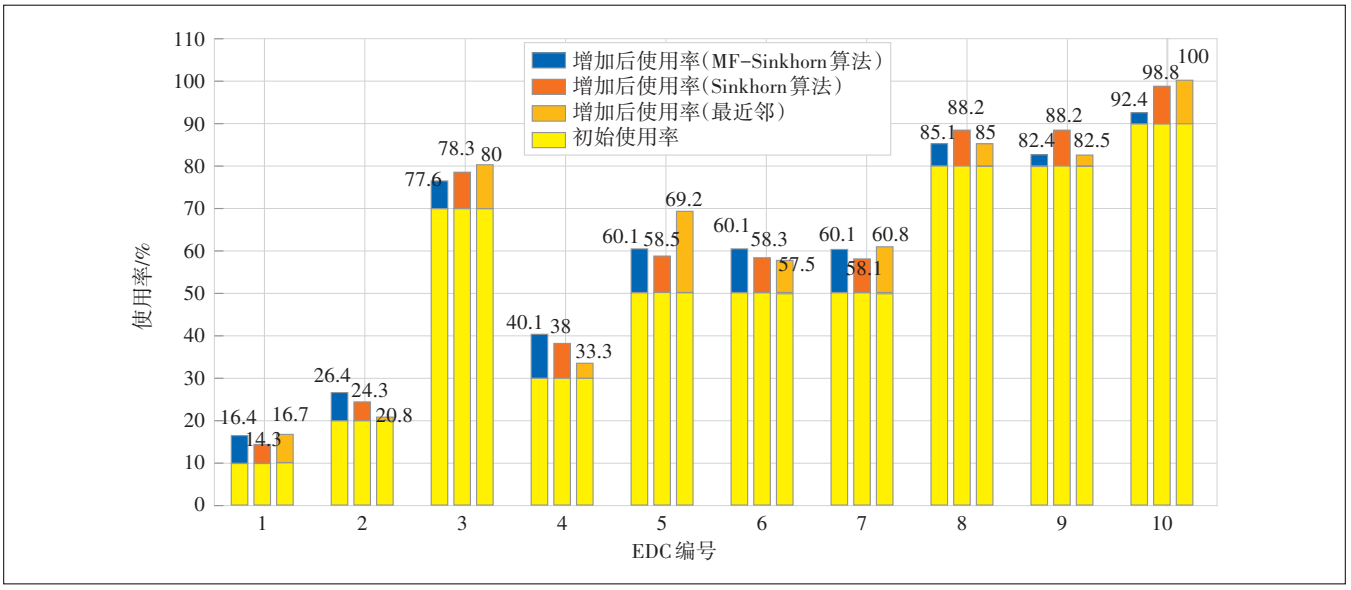

图6　多EDC空间利用率变化对比

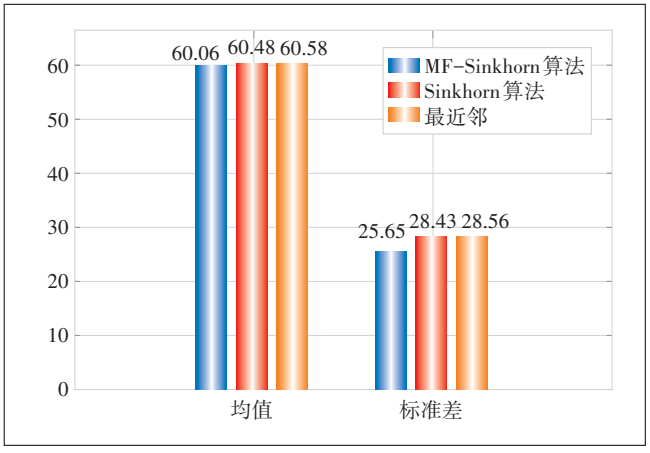

图7　多EDC空间利用率标准差对比

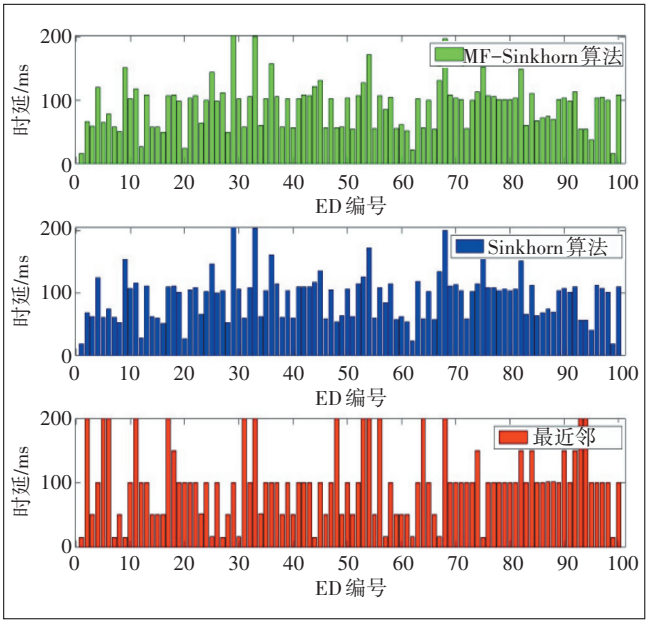

图8　多EDC综合写入平均时延情况

在调度算法对性能的影响方面,图8展示了各算法对多EDC写入时延的影响效果,其中,横轴为ED节点编号,纵轴为各ED节点侧实测综合写入平均时延。

多EDC综合写入平均时延和方差如图9所示,较原始Sinkhorn算法,改进后的MF-Sinkhorn算法使平均时延降低了2.13 ms(约2.28%);与生产使用的最近邻策略相比,降低5.93 ms(约6.01%)。多EDC间时延的均衡度可使用标准差指标进行评估,标准差越小,则EDC间的平均时延差距越小,在综合平均时延下降的前提下,说明MF-Sinkhorn算法更好地规避了长尾出现频次。实验证明,该调度算法通过动态平衡跨数据中心的负载压力,既缩短了端到端响应时间,又增强了存储系统的服务稳定性。

4　结束语

本文提出了一种面向边缘智能的新型边缘存储调度体系,在边缘存储调度架构方面,在ED层与EDC层之间通过动态协同机制替代传统静态映射方式,新增EDC选举机制,在主EDC上调用MF-Sinkhorn算法,实现边缘设备节点与边缘数据中心Bucket新建位

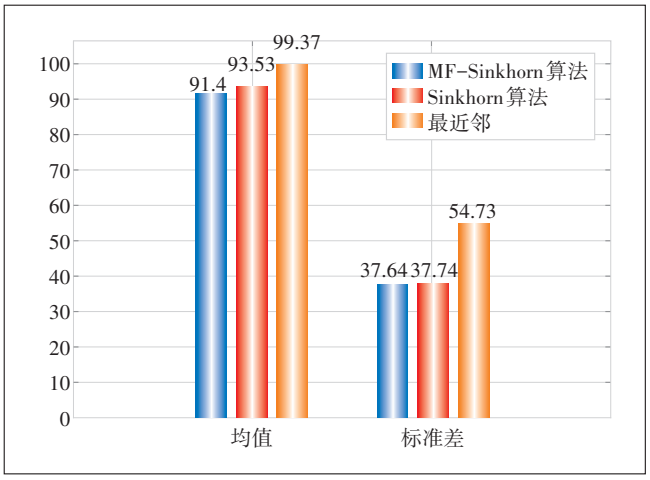

图9　多EDC综合写入平均时延和方差示意

置的智能映射计算。仿真实验结果表明,该方案的多EDC空间利用率均衡度提升2.91%,能够有效缓解高负载EDC的压力,延长关键节点的服务寿命。在性能方面,多EDC性能提升6.01%。综上所述,MF-Sinkhorn算法创新性地将动态选举机制与最优传输理论相结合[21],实现了多EDC负载均衡和性能的双重提升,为边缘智能场景下的存储资源调度开辟了新的技术路径。

参考文献:

[1] Deng S G, Zhao H L, Fang W J, et al. Edge intelligence: the confluence of edge computing and artificial intelligence[J]. IEEE Internet of Things Journal, 2020, 7(8): 7457-7469.

[2] 侯祥鹏,兰兰,陶长乐,等. 边缘智能与协同计算:前沿与进展[J]. 控制与决策,2024,39(7):2385-2404.

[3] Dai M H, Su Z, Li J L, et al. An energy-efficient edge offloading scheme for UAV-assisted Internet of things[C]//2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). Singapore: IEEE, 2020: 1293-1297.

[4] Ebrahimzadeh A, Maier M. Cooperative computation offloading in FiWi enhanced 4G HetNets using self-organizing MEC[J]. IEEE Transactions on Wireless Communications, 2020, 19(7): 4480-4493.

[5] 张雪晴,刘延伟,刘金霞,等. 面向边缘智能的联邦学习综述[J]. 计算机研究与发展,2023,60(6):1276-1295.

[6] Letaief K B, Shi Y M, Lu J M, et al. Edge artificial intelligence for 6G: vision, enabling technologies, and applications[J]. IEEE Journal on Selected Areas in Communications, 2022, 40(1): 5-36.

[7] Cui Y G, Cao K, Zhou J L, et al. Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023, 42(5): 1518-1531.

[8] 林琳. 开发边缘计算存储需要考虑的关键因素[J]. 计算机与网络,2021,47(10):43.

[9] Zhou Z, Chen X, Li E, et al. Edge intelliigence: paving the last mile of artificial intellingence with edge computing[J]. Proceedings of the IEEE, 2019, 107(8): 1738-1762.

[10] Liu Y, Yu H M, Xie S L, et al. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(11): 11158-11168.

[11] 曹守欣,赵琉涛,金翊. 基于对象存储的云存储系统研究[J]. 计算机科学与应用,2014,4(12):333-343.

[12] Thorpe M. Introduction to optimal transport[EB/OL]. [2026-05-19]. https://www.damtp.cam.ac.uk/research/cia/files/teaching/Optimal_Transport_Notes.pdf.

[13] Thornton J, Cuturi M. Rethinking initialization of the Sinkhorn algorithm[C]/Proceedings of the 26th International Conference on Artificial Intelligence and Statistics. Tangier: PMLR, 2023: 8682-8698.

[14] 刘铎,杨涓,谭玉娟. 边缘存储的发展现状与挑战[J]. 中兴通讯技术,2019,25(3):15-22.

[15] Mishra S K, Sahoo B, Parida P P. Load balancing in cloud computing: a big picture[J]. Journal of King Saud University - Computer and Information Sciences, 2020, 32(2): 149-158.

[16] Syed S. Optimizing cloud resource allocation with machine learning: a comprehensive approach to efficiency and performance[D]. Hofuf: King Faisal University, 2024.

[17] Bogachev V I. Kantorovich problem of optimal transportation of measures: new directions of research[J]. Russian Mathematical Surveys, 2022, 77(5): 3-52.

[18] Peyré G. Numerical optimal transport and its applications[EB/OL]. [2026-05-19]. https://mathematical-tours.github.io/book-basics-sources/ot-sources/TransportEN.pdf.

[19] Rout L, Korotin A, Burnaev E. Generative modeling with optimal transport maps [EB/OL]. [2026-05-19]. https://arxiv.org/abs/2110.02999.

[20] Cuturi M. Sinkhorn distances: lightspeed computation of optimal transport[C]//Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. Lake Tahoe: Curran Associates Inc., 2013: 2292-2300.

[21] Peyré G. Course notes on computational optimal transport[EB/OL]. [2026-05-19]. https://www.sambuz.com/doc/course-notes-on-computational-optimal-transport-pdf-document-709277.

作者简介:
杜量,工程师,硕士,主要从事算力网络、人工智能研究工作;谭跃,工程师,硕士,主要从事下一代互联网、算网融合技术、存储技术研究工作;邓玲,正高级工程师,硕士,主要从事算力网络、5G无线、传输网络研究工作;曾楚轩,高级工程师,博士,主要从事算力网络、人工智能研究工作;曹畅,正高级工程师,博士,主要从事下一代互联网、算网融合技术研究工作;杨杰,工程师,硕士,主要从事算力网络、超智算集群架构研究工作。