陈逸飞, 陈庆奎. 一种面向并发 AI 数据流边缘处理集群的资源配置算法[J]. 智能计算机与应用,2025,15(11):1-8. DOI: 10.20169/j.issn.2095-2163.251101

一种面向并发 AI 数据流边缘处理集群的资源配置算法

陈逸飞, 陈庆奎
(上海理工大学 光电信息与计算机工程学院, 上海 200093)

摘 要: 面对多个 AI (Artificial Intelligence)数据流需要并发处理以及边缘处理集群中各个单元的处理能力不同的情况,如何充分利用边缘集群中的各类资源,降低 AI 数据流任务的处理时间和能耗,是一个具有重要意义的课题。本文设计了一个面向并发 AI 数据流的边缘集群架构,并对并发 AI 数据流以及边缘处理集群中的处理单元资源进行建模,提出了一种面向并发 AI 数据流边缘处理集群的资源配置算法 DLBE-PSO,该算法是基于粒子群的多目标优化算法,采用混沌映射与非线性递减结合的方式来更新权重,提高算法后期的局部搜索能力,同时对任务处理时间、边缘集群能耗、边缘集群负载均衡这 3 个关键指标进行分析。

关键词: 并发 AI 数据流; 多目标优化; 粒子群; 任务处理时间; 能耗; 负载均衡
中图分类号: TP393 **文献标志码**: A **文章编号**: 2095-2163(2025)11-0001-08

An algorithm for resource allocation in concurrent AI data stream edge processing clusters

CHEN Yifei, CHEN Qingkui
(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract: Facing the situation where multiple AI (Artificial Intelligence) data streams need concurrent processing and the processing capabilities of each unit in the edge processing cluster are different, how to make full use of various resources in the edge cluster and reduce the processing time and energy consumption of AI data stream tasks is a topic of great significance. This paper designs an edge cluster architecture for concurrent AI data streams, models the concurrent AI data streams and the processing unit resources in the edge processing cluster, and proposes a resource allocation algorithm DLBE-PSO for the edge processing cluster of concurrent AI data streams. This algorithm is a multi-objective optimization algorithm based on particle swarm. It adopts the combination of chaotic mapping and nonlinear decrement to update the weights and improve the local search ability in the later stage. Meanwhile, it analyzes the three key indicators of task processing time, energy consumption of edge clusters, and load balancing of edge clusters.

Key words: concurrent AI data streams; multi-objective optimization; particle Swarm; task processing time; energy consumption; load balancing

0 引 言

物联网迅猛发展,各种终端智能设备产生的数据不断激增,对服务器处理和分析数据的能力提出了巨大的挑战。为了提高用户体验并节省诸如存储、计算等资源,边缘计算的概念被提了出来[1]。为用户提供了强大的计算能力以支持移动计算等领域,主要优势在于可以减少传输到云端的数据量,从而进一步减轻网络带宽和云端的压力[2]。同时,人工智能(AI)技术近年也处于发展中[3]。但由于云计算模式存在着延时较高的特性,AI 技术无法满足部分实时计算的需求。有些终端设备计算能力十分有限,导致无法处理高数据量的 AI 任务。为了满足 AI 对计算的高性能需求,本文采用边缘计算的方式

基金项目: 国家自然科学基金(61572325); 上海重点科技攻关项目(19DZ1208903)。
作者简介: 陈逸飞(1999—),男,硕士,主要研究方向:边缘计算,并行计算。
通信作者: 陈庆奎(1966—),男,博士,教授,博士生导师,主要研究方向:计算机集群,并行计算,人工智能等。Email:Chenqingkui@usst.edu.cn。
收稿日期: 2024-03-14

来处理 AI 数据流。同时由于 AI 数据流会出现在不同的终端设备,这必然会形成大量的并发数据流[4],那么如何在边缘处理单元资源有限的情况下执行并发的 AI 数据流任务便成为研究热点[5],这其中存在着诸多难点:

1)边缘集群中处理单元的计算资源有限,无法在保证 AI 数据流计算结果准确的情况下,大幅降低并发任务的完成时间。此时如果需要保证数据流计算的 GPU 等资源的充足,将导致任务完成时间不可控。相反,过分追求降低 AI 数据流任务的时间,将导致边缘集群网络拥塞和算力不足等问题。

2)边缘集群的总能耗与集群中处理单元的运行时间以及处理计算的数据量有关。当待处理的数据量增大或是边缘集群运行时间增大,产生的能耗也会随之增长,因此如何提升边缘集群能量效率也是一个重要问题。

3)同一个处理单元会有多个维度的资源,一旦出现多维资源的分配不合理,会造成单个处理单元可被分配任务个数的减少以及整个边缘集群不同维度资源利用效率的降低,有可能会出现某些处理单元长久不被分配 AI 数据流任务,但同时某些处理单元一直在超负荷工作的情况,这大大影响了处理单元的可用性,说明 AI 数据流任务的不合理配置会导致诸如数据流到达速率与计算速率不一致的资源不匹配问题或是集群能耗成本增加的问题。

本文设计了一种面向并发 AI 数据流边缘处理集群的资源配置算法,将并发的 AI 数据流任务分配在边缘处理集群中的不同单元上,达到了在资源受限的情况下合理权衡各个单元的可用资源,降低了边缘集群处理能耗和数据流处理时延,实现了资源配置的全局平衡。

1 相关工作

随着边缘 AI 的兴起,如何在资源受限的边缘处理单元上进行 AI 数据流的计算成为研究热点[6]。提供边缘 AI 计算服务离不开集群的资源管理,其中最核心的就是资源的配置,包括异构资源的统一配置,资源的合理分配调度和优化等。而资源配置问题通常被划入组合优化问题的范畴,Simon 等证明了这类问题大多是 NP(Nondeterministic Polynomial time)问题和 NPC(NP-Complete,NP 完全问题)问题[7]。当前比较主流的资源配置研究方案有:以降低计算时延为目标的资源配置方案、以降低边缘系统能耗成本为目标的资源配置方案、以达到集群之间负载平衡为目标的资源配置方案等。

为了充分利用边缘集群处理单元的计算能力,减少任务的响应延迟,目前有很多基于时延的算法,比如首次拟合(First-Fit)算法、最佳拟合(Best-Fit)算法、最短任务优先(Shortest Job First,SJF)算法等。文献[8]提出了基于队列分组优先级的时延调度算法(BDS);文献[9]提出基于将时间片和优先级结合的动态优先级算法(DDPQs)。边缘集群负载均衡是指边缘集群处理单元的各类资源和并发 AI 数据流之间达到充分匹配。常用的负载均衡算法也有很多,其中静态负载均衡算法有无状态调度的轮循调度算法、加权轮循调度算法等[10];动态负载均衡算法有有状态调度的最小连接数算法、加权最小连接数算法、最快响应速度算法等[11]。文献[12]提出了一种放置算法,通过在边缘服务器转移数据流,最大限度地满足资源总需求并且平衡各个处理单元之间的负载。在考虑能耗问题时,需要首先考虑能耗的度量模型和测量方法,进一步研究能效优化方法,降低某项运行指标以带来能源的明显节约[13]。文献[14]针对集群的能耗问题进行研究,提出了一种支持动态电压缩放算法(DFVS)的集群调度算法,最大限度的减少服务器能源的浪费。

尽管上述算法解决资源配置单个目标时效率很高,但对于解决并发数据流多目标配置问题时却并不适用,因为计算时延、系统能耗、集群负载均衡这 3 个目标互相影响甚至会有冲突,不存在一个可以同时使每个目标最优的解决方案。

2 并发 AI 数据流边缘集群架构

2.1 边缘集群架构搭建

本文通过不同的边缘设备抓取图片,之后存储并进行 AI 模型的计算。由于数据流具有并发性且需要大量计算资源以及存储资源,因此本文用若干个复合功能处理单元设计边缘集群架构,其中每个单元都包括 Kafka 抓取模块负责从 AI 图片流中并行地抓取、消费数据;存储中间件负责存储 AI 数据流原始信息;AI 服务模块负责提供 AI 数据流算力的支持;通信缓冲区负责根据集群情况调度各个 AI 数据流;资源监控模块负责记录数据流任务完成总的时间,整体边缘集群的能耗情况和负载情况,便于做资源配置策略的调整,面向并发 AI 数据流边缘集群架构如图 1 所示。

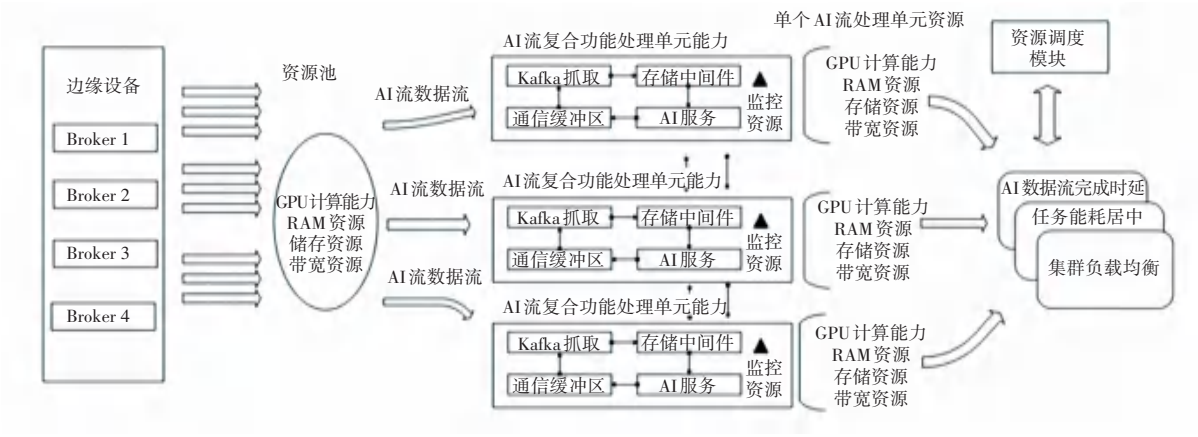

图 1　面向并发 AI 数据流边缘集群架构

Fig. 1　Concurrent AI data stream edge cluster architecture

1)若干个复合处理单元构建面向并发 AI 数据流的边缘集群架构,形成关于 GPU 计算能力、RAM 资源、存储资源、带宽资源的资源池,统计同一周期内每个 AI 数据流任务所需的各类总资源;

2)采用一定的资源配置策略,通信缓冲区会通知各个复合处理单元对应的并发 AI 数据流序列号;

3) AI 数据流接入对应的复合处理单元,通过 Kafka 管道方式接收图片数据,将数据临时存储,以便在一定时间周期内进行 AI 模型的计算;

4)通过资源监控模块管理复合处理单元的资源状态信息,利用各个复合处理单元的通信缓冲区进行多种资源配置算法切换,保证任务总完成时间、负载情况和能耗情况这 3 个指标可以达到均衡优化的效果;

5)各个复合处理单元根据计算模型进行模型计算,并将计算出的结果与上述 3 个指标值存储和记录下来,不断对边缘集群架构进行资源配置算法的调优。

2.2　资源分配问题描述

将当前并发的若干个 AI 数据流定义为 $\{D_1, D_2, D_3, \cdots, D_n\}$,数据流处理所需要的资源会放在资源池中,假设池中每个 AI 数据流处理所需的资源由一个五元组 $D_i = \{R_{\mathrm{id}}, R_{\mathrm{gpu}}, R_{\mathrm{ram}}, R_{\mathrm{hd}}, R_{\mathrm{bw}}\}$ 表示,其中 R_{id} 是每一个数据流任务的标识位,R_{gpu} 是进行该数据流任务所需要的 GPU 计算能力,R_{ram} 是完成该数据流任务所需要的内存占用,R_{hd} 是存储这批数据流任务所需的硬盘空间,R_{bw} 是传输数据流任务所需的带宽资源。

边缘处理集群中的 m 个复合处理单元表示为 $N = \{N_1, N_2, N_3, \cdots, N_m\}$,每个处理单元的资源表示为 $N_j = \{r_{\mathrm{id}}, r_{\mathrm{gpu}}, r_{\mathrm{ram}}, r_{\mathrm{hd}}, r_{\mathrm{bw}}\}$,其中 r_{id} 是每个处理单元的标识位,r_{gpu} 是每个处理单元的 GPU 计算能力,r_{ram} 是每个处理单元的内存占用,r_{hd} 是每个处理单元的存储资源,r_{bw} 是每个处理单元的带宽资源。

本文假设同一批次的并发 AI 数据流按照一定的策略分配到边缘集群中不同复合处理单元,即 n 个 AI 数据流会按照顺序到达 m 个复合处理单元,构造出一个 $n \times m$ 的矩阵,如下式:

$$\boldsymbol{S} = [s_{i,j}] = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,m} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & s_{n,m} \end{bmatrix} \tag{1}$$

其中,$s_{i,j}$ 表示数据流 i 在进行资源配置的时候被分配到了 j 单元上执行。

若 $s_{i,j} = 0$ 则表示 AI 数据流 i 还未分配到任何一个单元上;若 $s_{i,j} = 1$ 则表示 AI 数据流 i 完全被分配在 j 单元上执行。考虑到进行资源配置时 AI 数据流 i 可以进一步拆分到不同的单元,$s_{i,j}$ 的取值为 $[0,1]$。该矩阵满足下式约束:

$$\sum_{j=1}^{m} s_{i,j} = 1 \tag{2}$$

2.3　多目标资源配置数学模型

在并发 AI 数据流资源配置的过程中,每个复合处理单元的计算能力 GPU 资源、存储能力 ROM 资源等都是不同的,因此依据不同的资源配置方案来分配 AI 数据流任务,效果的差异非常大。本文考虑的并非单目标问题,而是基于多目标优化进行资源配置。因此建立数学模型时,不仅仅考虑并发数据流的任务总执行时间是否较小,同时也将资源配置后边缘集群架构的总能耗考虑在内,并确保整个集群的负载达到均衡,但这 3 种资源配置目标存在一

定的冲突。

并发 AI 数据流多目标资源配置的数学模型描述如下：

定义 1 任务总执行时间

一个集群中的所有复合处理单元对于 AI 数据流任务的时间消耗，本文主要考虑处理数据的时间 T_{process} 和数据从边缘设备传输到集群中服务器所用的耗时 T_{transfer} 以及根据一定的资源分配策略进行分配后每个 AI 数据流在处理单元上所需要的等待时间 T_{wait}。

那么单个数据流 D_i 在集群复合处理单元上实际任务总执行时间 T_{total}，公式如下：

$$T_{\text{total}} = T_{\text{process}} + T_{\text{transfer}} + T_{\text{wait}} \tag{3}$$

由于一个复合处理单元的资源会被用来处理多个 AI 数据流，所以分配到同一个单元的 AI 数据流也要按照先后顺序依次计算，因此每个 AI 数据流在处理单元等待的时延也必须考虑在内。假设分配到集群复合处理单元 N_j 上的数据流队列为 $L_j = \{D_1', D_2', \cdots, D_i', \cdots, D_q'\}$，其中，$D_i'$ 表示这个数据流在集群复合处理单元 N_j 上处于数据流队列的第 i 位，那么数据流 D_i' 的等待时延公式如下：

$$T_{\text{wait}}^{i} = \begin{cases} 0, & i = 1 \\ T_{\text{wait}}^{i-1} + T_{\text{process}}^{i-1} + T_{\text{transfer}}^{i-1}, & i > 1 \end{cases} \tag{4}$$

那么 n 个并发数据流在 m 个复合处理单元上的总完成时间，公式如下：

$$\text{execution} = \sum_{i=1}^{n} \max_j \sum_{j=1}^{m} s_{ij} \times T_{\text{total}} \tag{5}$$

定义 2 边缘集群总能耗

集群复合处理单元的能耗可以分为处理单元静态产生的能耗、计算 AI 数据流任务时的动态能耗以及边缘设备进行数据流传输时产生的能耗。本文通过获取实时电流的方式来计算每个复合处理单元的实时能耗[15]。处理单元 i 在单个周期（即任务总执行时间 T_{total}^{i}）内产生的能耗的计算公式为：

$$\text{energy} = \int_{t_0}^{t_{\text{total}}} p_i(I_i(t))\,\mathrm{d}t \tag{6}$$

其中，$p_i(I_i(t))$ 为时间间隔 $[t_0, t_{\text{total}}]$ 中的处理单元的能耗，可以通过下式得到：

$$p_i(I_i(t)) = p_i^{\text{static}}(I_i(t)) + p_i^{\text{dynamic}}(I_i(t)) + p_i^{\text{transfer}}(I_i(t)) \tag{7}$$

其中，$p_i^{\text{static}}(I_i(t))$ 是处理单元 i 的静态能耗，即在该单元闲置（等待 AI 数据流）的时候产生的能耗；$p_i^{\text{dynamic}}(I_i(t))$ 是处理单元 i 的动态能耗，包括处理单元从等待状态到分配了若干个 AI 数据流进行计算的阶段，这期间能耗会随着执行时间而发生变化；$p_i^{\text{transfer}}(I_i(t))$ 是处理单元每次传输数据产生的能耗；$I_i(t)$ 是处理单元 i 随时间变化的实时电流情况。

服务器的能耗与其执行任务时的电流经常呈一种线性关系[16]，如下式所示：

$$\begin{cases} p_i^{\text{static}}(I_i(t)) = U_i \cdot I_{\text{static}} \\ p_i^{\text{dynamic}}(I_i(t)) = U_i \cdot I_{\text{dynamic}} \\ p_i^{\text{transfer}}(I_i(t)) = U_i \cdot I_{\text{transfer}} \end{cases} \tag{8}$$

其中，I_{static} 是处理单元等待数据流的时候产生的电流；I_{dynamic} 是处理单元正在进行 AI 数据流计算的时候产生的电流；I_{transfer} 是进行数据流传输的时候产生的电流。

定义 3 边缘集群负载均衡率

边缘集群中各处理单元的 GPU 计算能力、内存大小、存储容量和网络带宽等参数都存在着差异。因此当 AI 数据流并发到来时，需要综合考虑集群中各个复合处理单元的性能差异，尽可能使各个处理单元各项资源的利用率较为平均，避免出现某一复合处理单元的利用率过高而有的单元处于空闲的情况，因此还需要确定一个负载均衡的指标来衡量资源分配时边缘集群的负载水平。

对于 AI 数据流 i 分配在单个复合功能单元 j 上，那么该单元上某项资源的使用率的计算公式如下：

$$U^k = \frac{R_{i\text{指标}}}{r_{j\text{指标}}} \tag{9}$$

其中，$R_{i\text{指标}}$ 是 AI 数据流 i 在资源池中所需的某个维度的资源，而 $r_{j\text{指标}}$ 是复合功能单元 j 所拥有的对应维度的资源。

那么，对单一复合功能单元的所有资源的综合负载的计算公式如下式：

$$L_j = \sum_{k=1}^{\text{sum}} \lambda^k U^k \tag{10}$$

其中，sum 表示计算负载均衡率时要考虑的资源种类（本文中是 4 种），λ^k 是各项资源的权重，满足下式约束：

$$\sum_{k=1}^{\text{sum}} \lambda^k = 1 \tag{11}$$

本文提出的边缘集群架构的资源整体负载的值 LBI 可以用下式来计算：

$$\text{LBI} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(L_j - \frac{1}{m}\sum_{j=1}^{m} L_j\right)^2} \tag{12}$$

LBI 的值越低说明该集群在处理并发 AI 数据

流时各个处理单元之间的负载越均衡。

3　基于粒子群算法的资源配置策略

3.1　粒子群算法

在边缘集群资源配置时，并发 AI 数据流的完成时间、集群的负载均衡指标以及最后的能耗是由数据流与处理单元的映射关系所决定的，不同的映射关系代表不同的资源配置方案。不同约束条件下多维度资源耦合配置是一种较复杂的非统一优化问题，处理这类问题通常使用遗传算法、粒子群算法等[17]。粒子群从随机解开始，通过不断更新自身的速度和位置，最终趋向于一个稳定的收敛解，这种算法相比较而言具有更优越的全局收敛性，是一种基于种群的随机搜索算法，通过群体中粒子间的合作与竞争产生的群体智能指导进行优化搜索[18]。本文提出的 DLBE-PSO 算法是基于改进粒子群算法实现的，主要结合并发 AI 数据流资源建立数学模型，针对边缘集群复合处理单元进行多目标资源配置。

在粒子群算法中，每一个粒子的位置向量都代表一种配置边缘集群资源的方案。每个粒子的位置向量都是 n 维的，其中 n 也表示并发 AI 数据流队列中的数据流数。n 维向量中每个坐标的取值范围为 $[1, m+1)$，m 是集群中处理单元的数量。粒子位置向量的每个坐标表示数据流将被分配到的处理单元编号，取坐标的整数部分作为数据流与处理单元的映射结果。若边缘集群中处理单元 $m=6$，并发 AI 数据流个数 $n=10$，某一粒子的位置向量(4.2, 3.3, 2.1, 4.5, 5.2, 2.6, 6.6, 2.5, 1.4, 2.7)，并发数据流与复合处理单元的映射关系见表 1 。

表 1　并发数据流与复合处理单元的映射关系

Table 1　Mapping relationship of concurrent data streams and composite processing units using particle swarm optimization algorithm

AI 数据流序号	粒子位置	复合处理单元编号
1	4.2	4
2	3.3	3
3	2.1	2
4	4.5	4
5	5.2	5
6	2.6	2
7	6.6	6
8	2.5	2
9	1.4	1
10	2.7	2

则粒子群算法得到的 AI 数据流调度方案为：{D9}→N1，{D3、D6、D8、D10}→N2，{D2}→N3，{D1、D4}→N4，{D5}→N5，{D7}→N6。

3.2　粒子群的初始化

初始化时，首先确定全局粒子速度惯性，并随机生成 n 个初始粒子，这些粒子组成了一个大小为 n 的原始种群。每个粒子的初始位置取 $[1, m+1)$。对于粒子 i，其速度的 n 维向量可以表示为 $V_i=\{v_{i1}, v_{i2}, \cdots, v_{in}\}$。由于位置为连续的变量，而并发 AI 数据流的配置为离散的组合，因此考虑把连续的变量处理成离散的资源配置组合。每个粒子的初始位置和速度公式如下：

$$x_i^1 = \text{rand} \times (x_{\max} - x_{\min}) + x_{\min} \tag{13}$$

$$v_i^1 = \text{rand} \times (v_{\max} - v_{\min}) + v_{\min} \tag{14}$$

其中，rand 为 0 到 1 之间均匀分布的随机数；$x_{\max}$ 为位置向量的最大值；$x_{\min}$ 为位置向量的最小值；$v_{\max}$ 为速度的最大值；$v_{\min}$ 为速度的最小值。

3.3　适应度计算

本文边缘集群架构针对处理并发 AI 数据流主要考虑的是任务总执行时间、集群负载均衡和能耗这 3 个指标，因此设计的适应度函数公式如下：

$$\text{fitness} = f(x_i^t) = [f_{\text{execution}}(x_i^t), f_{\text{LBI}}(x_i^t), f_{\text{energy}}(x_i^t) \tag{15}$$

其中，$f_{\text{execution}}(x_i^t)$ 表示并发 AI 数据流任务总执行时间的适应度函数；$f_{\text{LBI}}(x_i^t)$ 为边缘集群负载均衡指标的适应度函数；$f_{\text{energy}}(x_i^t)$ 为边缘集群能耗的适应度函数。

由于适应度函数的 3 个指标不统一，需要将 3 个子适应度函数进行数据预处理，再根据配置策略赋予多个子目标不同的权重值，最后转换为单个目标。

数据预处理是将子适应函数进行取对数，再取倒数的操作。

数据预处理后的任务总执行时间的适应度函数：

$$f_{\text{execution}}(x_i^t) = \frac{1}{\log_2 \text{execution}(x_i^t)} \tag{16}$$

其中，$\text{execution}(x_i^t)$ 表示经过 t 次迭代后的粒子群中第 i 个解的任务总执行时间。

数据预处理后的负载均衡指标的适应度函数如下式：

$$f_{\text{LBI}}(x_i^t) = \frac{1}{\log_2 \text{LBI}(x_i^t)} \tag{17}$$

其中，$\text{LBI}(x_i^t)$ 表示经过 t 次迭代后的粒子群中第 i 个解的负载均衡指标。

数据预处理后的边缘集群能耗的适应度函数如下式：

$$f_{\text{energy}}(x_i^t) = \frac{1}{\log_2 \text{energy}(x_i^t)} \tag{18}$$

其中，$\text{energy}(x_i^t)$ 表示经过 t 次迭代后的粒子群中第 i 个解的能耗情况。

加权处理后的适应度函数如下式：

$$\text{fitness} = f(x) = \alpha f\text{execution}(x) + \beta f\text{LBI}(x) + \gamma f_{\text{energy}}(x) \tag{19}$$

其中，α,β,γ 分别为3个目标对应的权重值，且 $\alpha+\beta+\gamma=1$，$\alpha \in [0,1]$，$\beta \in [0,1]$，$\gamma \in [0,1]$。

3.4 计算个体、群体的极值并更新历史最优值和位置

先对种群中的每个粒子进行计算，得到解的适应度函数值，再将适应度函数值与该解的历史最优值比较，找到个体极值 pbest；将适应度函数值与群体历史最优值进行比较，找到群体极值 gbest；通过公式(20)更新每个粒子速度，通过公式(21)更新每个粒子的位置，经过一次次迭代更新，使得粒子最终趋向于最优解。

$$v_i^{k+1} = \omega v_i^k + c_1 \text{rand}_1 \times (\text{pbest}_i - x_i^k) + c_2 \text{rand}_2 \times (\text{gbest} - x_i^k) \tag{20}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{21}$$

其中，v_i^{k+1} 为第 i 个粒子在第 $k+1$ 次迭代的速度；c_1、c_2 分别为粒子个体的学习因子和粒子群体的学习因子；rand_1、rand_2 表示0到1之间均匀分布的随机数；pbest_i 表示第 i 个粒子的历史最优位置；gbest表示全局最优的位置；x_i^k 表示第 i 个粒子第 k 次迭代的位置。

3.5 更新惯性权重

惯性权重值在粒子群算法中有至关重要的作用，其取值的合理性会严重影响粒子群算法在全局搜索和局部搜索的平衡。而传统的粒子群算法在迭代的过程中通常是采用固定不变的惯性权重值。为了应对算法迭代后期局部搜索能力不足，本文采用了一种将混沌 Sine 映射与非线性递减相结合的方式更新惯性权重值。非线性递减可以使惯性权重值随着迭代次数的增加逐渐减小，且递减的形式是由快到慢的。这种策略能够在算法的不同阶段灵活调整粒子的行为，即在前期增强算法的全局搜索能力，后期增强局部搜索能力，更好地适应问题的复杂性，提高算法的精度和收敛性[19]。混沌 Sine 映射作为经典的混沌映射具有良好的遍历性。当在更新惯性权重值的过程中引入混沌映射，可以增强算法的随机性，如下式：

$$\omega^k = S(t) \times \omega_{\max} - (\omega_{\max} - \omega_{\min}) \times (t/T) \tag{22}$$

其中，t 为当前迭代次数；T 为最大迭代次数；ω^k 为第 k 次迭代时的权重值；$\omega_{\max}$、$\omega_{\min}$ 分别为惯性权重的最大值和最小值，这里的 $\omega_{\max}$ 取0.9，$\omega_{\min}$ 取0.3；$S(t)$ 为混沌 Sine 映射，公式如下：

$$\begin{cases} S(t) = \mu \times \sin(S(t-1) \times \pi) \\ S(0) = \text{rand},\ \mu \in (0,1] \end{cases} \tag{23}$$

代入混沌 Sine 映射后的惯性权重变化如图2所示。

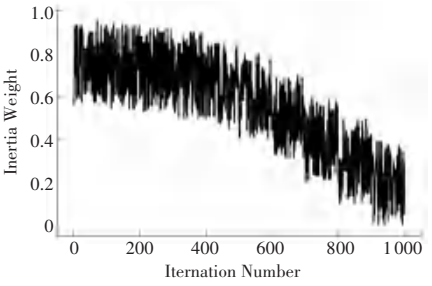

图2 惯性权重变化图

Fig. 2 Inertia weight variation

3.6 算法描述

本文提出的 DLBE-PSO 算法具体的流程如下：

1)设定并发数据流个数以及每个数据流的数据规模，设定边缘集群的处理单元个数以及每个单元的 GPU 资源、内存资源、存储资源以及带宽资源；

2)粒子群初始化，即粒子位置初始化，粒子速度初始化；

3)由于每个粒子的状态为并发 AI 数据流情况下边缘集群中处理单元的资源配置方案，判断是否合理即判断每个 AI 数据流任务最后是不是被分配的若干个处理单元完全完成及判断每个复合处理单元在同一时刻各项资源利用率有没有超过100%；

4)对该粒子群的解进行适应度函数计算；

5)找到个体粒子极值 pbest 和群体粒子极值 gbest，更新最佳适应度；

6)判断是否满足迭代条件。若满足，则输出全局最优的边缘集群处理单元配置策略以及该策略下的适应度值；若不满足，则执行步骤8)。

7)动态更新惯性权重；

8)更新粒子速度向量信息，更新粒子位置向量信息，产生新一代粒子群，转到步骤3)，判断解是否合理。

最后，产生的解位置向量 $x_{t+1,i}$ 还需要进行边界条件处理，当其超过解集的位置向量约束的范围即超过粒子的最大取值或是低于粒子的最小取值，则将解重新赋值为约束范围内的随机值，再将位置向

量向下取整数与复合处理单元一一映射。

4　实验

实验抓取上海市公交车的实时监控视频流，将周期性产生的视频数据转换为数据流，并利用目标检测算法 YOLO 计算公交车车厢拥挤度。

实验中共部署了 10 台服务器作为复合处理单元，搭建边缘集群，以进行完整的并发 AI 数据流的处理，并通过集群架构中的监控模块获取如 GPU 占用、内存占用、硬盘存储空间占用、网络带宽、任务完成时间、能耗等指标。其中一个边缘集群复合处理单元的详细配置见表 2。

表 2　边缘集群复合处理单元资源配置

Table 2　Resource allocation of edge cluster composite processing units

配置项	配置信息
CPU	Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz
操作系统	Centos7.0
内核版本	3.10.0- 123.el7.x86_64
网卡	I350-T4-4-port 1Gb/s PCI-e X4
内存	32 G
GPU	GTX1080+GTX1080
带宽/Mbps	1 000

为了证明本文提出的资源配置算法 DLBE-PSO 对于并发 AI 数据流的资源配置是有效的，实验着重从并发 AI 数据流任务总执行时间、边缘集群复合处理单元总的能耗以及各个处理单元负载均衡率这 3 个指标对不同资源配置算法的结果进行了多次对比实验。

具体实验步骤：并行 AI 数据流共 30 条即共 30 个 Kafka 抓取模块同时开始工作，每个 AI 数据流随机生成 100~1 000 张图片；每个 AI 数据流所需各项资源的具体值事先以同一标准量化，集群处理单元的计算能力控制在一定范围内，且各项资源利用率自主设定，通过监控模块观察各个处理单元的负载以及能耗情况。本次试验分别选择 6，8，10 个复合处理单元搭建边缘集群，重复进行 50 次并发 AI 数据流处理任务，统计 4 种资源配置算法各自的任务完成时间，能耗情况和负载均衡率的平均值。

在本文资源配置算法 DLBE-PSO 具体参数见表 3，这样设置有利于算法的收敛，可以达到较好的求解结果，其中补充参数为计算负载均衡指标时的各项资源的权重值 $\lambda_1 = 0.4$，$\lambda_2 = 0.2$，$\lambda_3 = 0.2$，$\lambda_4 = 0.2$。

表 3　DLBE-PSO 算法参数表

Table 3　DLBE-PSO algorithm parameters

参数	取值
种群规模 N	60
迭代次数	1 000
AI 数据流数量	30
惯性因子 ω	0.9，0.3
学习因子 c_1	0.73
学习因子 c_2	0.73
适应度权值 α	0.4
适应度权值 β	0.3
适应度权值 γ	0.3
速度值 v	2，-2

为验证本文提出的资源配置算法性能，选取了文献[11]提出的最小连接数算法、文献[9]提出基于将时间片和优先级结合的动态优先级算法 DDPQs、文献[14]提出的动态电压缩放算法 DFVS、普通粒子群算法 PSO 以及本文提出的 DLBE-PSO 算法作对比实验。并发 AI 数据流总的任务总执行时间，边缘集群能效情况和负载均衡指数分别如图 3~图 5 所示。

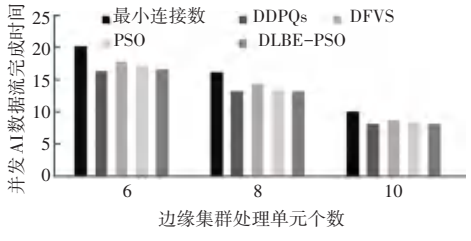

图 3　4 种资源配置算法下的任务总执行时间

Fig. 3　Total execution time of tasks under four resource allocation algorithms

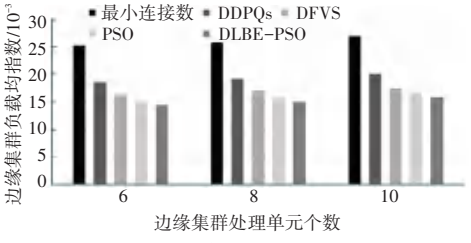

图 4　4 种资源配置算法下的边缘集群能效情况

Fig. 4　Energy efficiency of edge cluster under four resource allocation algorithms

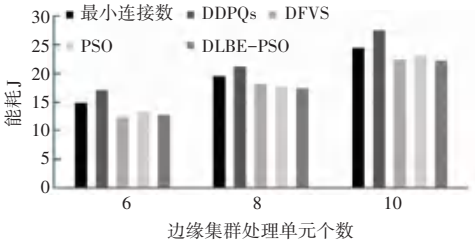

图 5　4 种资源配置算法下的边缘集群负载均衡指数

Fig. 5　Load balancing index of edge cluster under four resource allocation algorithms

总体来看,随着边缘集群处理单元的增加,并发 AI 数据流任务总执行时间是一定会有所降低的,但是考虑到如集群各个处理单元之间的通信带宽等成本以及某些处理单元本身资源的限制,任务总执行时间提高有限;DFVS 这类以能耗为主要目标的配置算法常常可以降低处理单元功率,减少处理单元个数,但对并发 AI 数据流的任务总执行时间影响非常大。从实验结果来看,在并发数据流任务总执行时间占比较大时,DDPQs 算法和本文提出的改进粒子群算法是比较有优势的;与最小连接数算法、DFVS 算法以及基本的粒子群算法相比,本文提出的 DLBE-PSO 算法优势显著,任务总执行时间分别提高了 21.96%、7.34%、2.51%;在集群能耗方面,DLBE-PSO 算法比最小连接数算法、DDPQs 算法、PSO 算法分别提高了 12.90%、26.59%、4.21%,与 DFVS 算法相差不大;在负载均衡指数上,DLBE-PSO 算法也比最小连接数算法、DDPQs 算法、DFVS 算法、PSO 算法分别提升了 71.60%、27.54%、12.15%、4.56%。

5 结束语

本文研究在边缘集群环境下的资源配置问题,主要针对的场景是当并发 AI 数据流传输到集群时,综合考虑并发 AI 数据流任务总执行时间,边缘集群负载均衡以及集群能耗这三大指标,提出了一个面向并发 AI 数据流边缘处理集群的多目标资源配置算法 DLBE-PSO。最后的实验结果表明,本文设计的边缘集群架构以及提出的算法在基本保证负载均衡的情况下,将能耗控制在一定范围内,并在此区间找到了最小的任务总执行时间。当然,本文对各个目标约束以及同一处理单元不同维度资源的考虑并不充分,算法在任务总执行时间指标下与一些资源调度算法相比优势不够明显,后续将在建模的过程中考虑更多的目标约束,并采用深度学习算法着重对任务总执行时间这一指标下的资源配置做进一步的优化。

参考文献

[1] SHI W, CAO J, ZHANG Q, et al. Edge computing: Vision and challenges[J]. Internet of Things Journal, 2016, 3(5): 637-646.

[2] 赵梓铭, 刘芳, 蔡志平, 等. 边缘计算: 平台、应用与挑战[J]. 计算机研究与发展, 2018, 55(2): 327-337.

[3] SIL R, ROY A, BHUSHAN B, et al. Artificial intelligence and machine learning based legal application: The state-of-the-art and future research trends[C]// Proceedings of 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS). Piscataway, NJ: IEEE, 2019: 57-62.

[4] 黄东生, 陈庆奎. 一个并发 AI 数据流处理节点内的通信模型[J]. 智能计算机与应用, 2022, 12(11): 26-33.

[5] 孙笑科. 边缘计算网络中任务卸载和资源分配优化研究[D]. 北京: 北京交通大学, 2021.

[6] LI En, ZENG Liekang, ZHOU Zhi, et al. Edge AI: On-demand accelerating deep neural network inference via edge computing[J]. IEEE Transactions on Wireless Communications, 2020, 19(1): 447-457.

[7] SIMON U H. On approximate solutions for combinatorial optimization problems[J]. Siam Journal on Discrete Mathematics, 2006, 3(2): 294-310.

[8] DOVROLIS C, STILIADIS D, RAMANATHAN P. Proportional differentiated services: Execution differentiation and packet scheduling[J]. IEEE/ACM Transactions on Networking, 2002, 10(1): 12-26.

[9] 张登银, 许扬扬, 蒋娟. 基于时延的动态优先级调度算法[J]. 计算机技术与发展, 2011, 21(2): 162-165.

[10] GUTMAN D, BISTRITZ Y. Speaker verification using phoneme-adapted gaussian mixture models[C]//Proceedings of 2002 11th European Signal Processing Conference. Piscataway, NJ: IEEE, 2002: 1-4.

[11] 李坤, 王百杰. 服务器集群负载均衡技术研究及算法比较[J]. 计算机与现代化, 2009 (8): 7-10.

[12] TANG C, STEINDER M, SPREITZER M, et al. A scalable application placement controller for enterprise data centers[C]// Proceedings of the 16th International Conference on World Wide Web. 2007: 331-340.

[13] 宋杰, 李甜甜, 闫振兴, 等. 一种云计算环境下的能效模型和度量方法[J]. 软件学报, 2012, 23(2): 200-214.

[14] VON LASZEWSKI G, WANG L, YOUNGE A J, et al. Power-aware scheduling of virtual machines in dvfs-enabled clusters[C]// Proceedings of 2009 IEEE International Conference on Cluster Computing and Workshops. Piscataway, NJ: IEEE, 2009: 1-10.

[15] 方玉玲. 基于 GPU 能耗分析的集群可靠性方法[D]. 上海: 上海理工大学, 2018.

[16] MA D, BONDADE R. Enabling Power-efficient DVFS operations on silicon[J]. IEEE Circuits and Systems Magazine, 2010, 10(1): 14-30.

[17] 陆怀谷, 刘绍东, 张渊, 等. 基于自适应粒子群算法的增量配电网电源博弈规划[J]. 南京理工大学学报, 2021, 45(2): 150-157.

[18] TRELEA I C. The particle swarm optimization algorithm: Convergence analysis and parameter selection[J]. Information Processing Letters, 2003, 85(6): 317-325.

[19] EL-SHORBAGY M A, MOUSA A A, NASR S M. A chaos-based evolutionary algorithm for general nonlinear programming problems[J]. Chaos, Solitons and Fractals, 2016, 85: 8-21.