计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(9):271−278 [doi: 10.15888/j.cnki.csa.008111]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

淤地坝监测场景下边缘计算协作式任务调度方法①

章新川[1], 胡　炜[2], 王怀军[2], 安　洋[2], 李军怀[2]

[1](陕西省水利厅, 西安 710048)
[2](西安理工大学 计算机科学与工程学院, 西安 710048)
通讯作者: 王怀军, E-mail: wanghuaijun@xaut.edu.cn

摘　要: 针对目前我国西北地区淤地坝实时监测问题, 研究了淤地坝监测与预警任务的调度方法. 为避免淤地坝坝体隐患发现不及时, 提高预警系统的时效性, 本文考虑了任务卸载至边缘服务器的平均等待时间, 提出了一种淤地坝监测场景下边缘计算协作式任务调度方法. 根据任务计算量、边缘服务器计算能力等信息建立计算任务完成时间模型, 然后采用模拟退火算法优化计算任务卸载位置, 设计了一种多个边缘计算服务器相互协作的任务调度策略. 实验结果表明, 该方法有效降低了监测任务的计算时间, 提高了监测预警的时效性.

关键词: 淤地坝监测; 边缘计算; 任务调度; 任务完成时间模型

引用格式: 章新川,胡炜,王怀军,安洋,李军怀.淤地坝监测场景下边缘计算协作式任务调度方法.计算机系统应用,2021,30(9):271–278. http://www.c-s-a.org.cn/1003-3254/8111.html

Collaborative Task Scheduling Strategy for Mobile Edge Computing in Dam Monitoring

ZHANG Xin-Chuan[1], HU Wei[2], WANG Huai-Jun[2], AN Yang[2], LI Jun-Huai[2]

[1](Department of Water Resources, Shaanxi Province, Xi'an 710048, China)
[2](Faculty of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: Concerning the real-time monitoring problem of warping dams in northwest China, this work studies the scheduling method of warping dam monitoring and early warning tasks. To avoid the delay in discovering the hidden trouble of warping dams and improve the timeliness of the warning system, this study considers the average waiting time from task unloading to edge servers and proposes a collaborative task scheduling method based on edge computing in warping dam monitoring. A task completion time model is built according to the task computation, computing power of edge servers, and other information. Then, a simulated annealing algorithm is used to optimize the unloading position of computing tasks. A task scheduling strategy is designed in which multiple edge computing servers cooperate. Experimental results show that this method can greatly reduce the calculation time of monitoring tasks and improve the timeliness of monitoring and early warning.

Key words: warping dam monitoring; edge computing; task scheduling; task completion time model

1　引言

淤地坝作为坝区资源开发利用和经济建设的基础工程, 在黄土高原水土保持中的地位尤为显著[1]. 同时也存在相应的溃坝风险, 因溃坝事件导致的水土流失、生态破坏时有发生, 严重的溃坝事件会危害坝区居民生命安全. 目前淤地坝在信息化管理、险情预测方面, 存在数据利用率低、数据记录方式落后等问题. 为解决这些问题, 坝区远程监测、实时评估等在线实

① 基金项目: 陕西水利科技项目 (2020slkj-17); 国家自然科学基金 (61971347); 西安市科技计划 (2020KJRC0093)
Foundation item: Water Conservancy Science and Technology Project of Shaanxi Province (2020slkj-17); National Natural Science Foundation of China (61971347); Science and Technology Program of Xi'an Municipality (2020KJRC0093)
收稿时间: 2020-12-15; 修改时间: 2021-01-18; 采用时间: 2021-02-07; csa 在线出版时间: 2021-09-02

时监测模式逐渐成为淤地坝防护的研究热点.

在 5G、物联网技术高速发展的背景下, 采用“云-边-端”方式实现对淤地坝的实时监测、传输、处理和风险预警, 具有重要意义. 通过部署在坝区的水尺、雨量计、视频等传感器采集数据, 并经过边缘节点预处理后上传至云端统一管理和风险评估与预警. 近几年, 有学者通过结合坝区传感数据建立淤地坝溃坝模型, 将采集数据作为参数输入溃坝风险模型进行预警[2], 实现对所覆盖坝区的 24 小时智能监测. 然而, 通常情况下淤地坝基础溃坝模型计算复杂度较高, 也会是否在汛期而进一步提升. 同时由于淤地坝数量多、分布广, 一般按照行政区划进行管理, 因此采用“云-边-端”方式可以有效保证监测与预警任务执行的实时性.

本文考虑了实际场景中各因素对计算任务完成时间的影响, 建立了相应的数学模型, 并利用模拟退火 (Simulated Annealing, SA) 算法能够快速提供一个较为精确的近似解[3], 对淤地坝的计算任务调度问题进行求解.

2 相关研究

2.1 淤地坝安全问题研究

顾艳玲等[4] 基于熵理论将不同赋权方法进行有机集成, 明确了坝体安全评估特征值权重指标, 并通过最大化模型参数香农熵, 保证估计结果的相对稳健. 黄黎明等[5] 基于工程风险清单、风险搜集、大数据风险分析三方面并结合物联网技术建立质量风险管理框架, 得到以物联网技术实现水利工程风险管理的重要经验. 王乃欣等[6] 利用 3Ds Max 等建模技术对淤地坝溃坝进行仿真, 为研究溃坝时的救灾抢险、应急管理做出了重要的参考依据. 黄肖[7] 利用 MIKE 软件建立了一套关于坝系布局与洪水关系的预测模型, 实现模拟研究洪水作用下淤地坝的调控作用. 李平等[8] 将现有判断水库群发溢洪方法与贝叶斯网络有机结合, 并根据场景特征增加了上下游水坝溃坝关系因子, 建立了完整的贝叶斯溢洪评估网络.

2.2 边缘计算任务调度研究

Seid 等[9] 提出了一种基于无人机的 F-RAN 体系结构新方向, 该结构采用分布式强化学习 (DRL) 算法对边缘设备进行学习, 有效提高了计算任务卸载效率, 并优化了计算资源分配. Liu 等 [10] 建立了一种基于优化边缘服务器资源的分配算法, 提高了移动边缘计算架构的计算资源利用率. Zhou 等[11] 通过模拟退火算法解决了 D2D 资源分配成本高的问题, 在资源分配组合中引入模拟退火算法, 有效降低资源分配成本和任务平均计算时间. Liao 等[12] 提出了一种有效的低复杂度启发式算法, 可以在可接受水平内保证边缘计算网络的计算资源保持, 降低时间成本并最大化用户任务卸载量. Anousha 等[13] 对传统 Min-Min 调度算法进行改进, 充分利用原算法在策略制定速度上的优点, 并避免原算法负载均衡性能不足、任务执行时间跨度高的缺点, 最终改善传统 Min-Min 算法性能.

对淤地坝进行实时监测, 并将其数据作为参数输入至溃坝模型, 由此产生淤地坝计算任务. 为提高淤地坝风险预警的时效性, 需要对计算任务进行合理调度, 缩短计算任务完成时间. 各淤地坝计算任务卸载位置的精确求解属于整数线性规划中的 NP 难问题. 此类问题虽存在精确最优解, 但计算量极大, 耗时极长.

通常分布式的边缘计算具有高隐私、低成本、易部署等优势, 但其计算资源、存储容量都有所限制. 在边缘计算中, 由于其计算资源受到限制, 从单一角度出发的应用计算切分模式将变得不可实施. 多用户模式情景下, 网络资源和服务器资源的竞争十分普遍. 此时计算任务卸载决策需要结合边缘计算资源分配决策[14], 通过减少应用完成时间的方式, 提高系统整体服务水平[15].

3 边缘计算场景下协作式任务调度策略

面向淤地坝监测任务调度策略实现思路如图 1 所示, 分为 3 个步骤: 信息获取、建立任务完成时间模型、模拟退火算法求最优解.

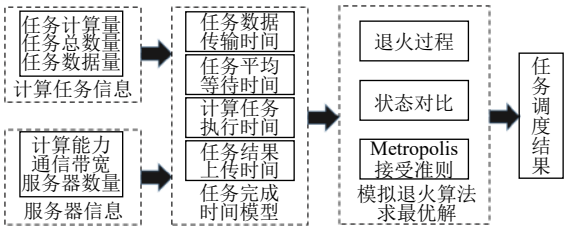

图 1 基于模拟退火算法的任务调度策略实现思路图

(1) 信息获取. 该过程为任务调度服务器收集节点产生的计算任务信息和边缘服务器计算资源信息. 其中计算任务信息包括: 计算任务总量、所有任务的计算量、传输计算任务的数据量; 边缘服务器信息包括: 服务器计算能力、基站与节点间的通信带宽、服务器数量.

(2) 建立任务完成时间模型. 建立任务完成时间模

型, 并代入任务信息与服务器信息, 计算所有任务在某个边缘服务器上的完成时间, 该服务器的运行时间即为完成卸载至此的所有计算任务时间.

(3) 模拟退火算法规划卸载位置. 随机分配计算任务至服务器上, 根据任务完成时间模型计算每一个服务器的运行时间, 并取服务器中最大完成时间作为系统任务完成时间. 降低算法温度, 同时对任务分配位置进行随机调整, 计算新分配方式下的任务完成时间, 并依据 Metropolis 接受准则接受新解. 重复迭代直至达到终止条件, 输出任务最优分配位置.

模拟退火算法 (SA) 由 Metropoils 准则和退火过程两部分组成[16]. 前者实现了 SA 算法如何在处于局部最优解时发生概率突跳并最终获得全局最优解, 是算法的基础; 后者则是 SA 算法寻找最优解的迭代过程. Metropoils 准则是一种以概率接收新解, 而不是以完全确定规则的重要采样方法. 得益于 Metropoils 准则, SA 算法可跳出局部最优解, 并寻找全局最优解.

4 任务完成时间模型

坝系包括多个坝区, 每个坝区采集节点都会定时产生一个计算量不同的计算任务, 坝系周边有若干个计算能力不同的边缘服务器与坝区连接. 该场景下, 所有节点产生的计算任务地位平等, 所有计算任务信息、边缘服务器计算能力信息可知. 其变量标识如表 1 所示.

表 1 边缘计算网络变量标识表

参数	含义
$Node_i$	第i个采集节点
ES_j	第j个边缘计算服务器
Num	淤地坝数量、计算任务数量
M	坝区边缘服务器个数
c_i	完成第i个任务所需要的计算量(GB)
d_i	计算任务传输数据量(MB)
f_i^E	边缘计算服务器j的计算能力(GB/S)
B_j	边缘服务器ES_j与信息节点$Node_i$的无线传输速率(MHz)
l^i	整数变量, 计算任务T_i在边缘计算服务器的位置, $l^i \in \{1,2,\cdots M\}$
λ	任务到达率
μ	边缘计算服务器的服务率
P_i	排队系统的状态概率
$\overline{L_q}$	平均队长
$\overline{t_{\text{wait}}}$	平均等待时间
P_{success}	任务卸载成功率

计算任务由坝区节点卸载至连接的边缘服务器上进行计算, 任务完成时间共分为 3 个部分: 任务数据传输时间、任务平均等待时间、任务在边缘服务器上的执行时间、计算结果由边缘服务器向中心服务器上传时间. 任务完成时间模型如下.

(1) 任务数据传输时间t_i^{trans}: 由采集节点与服务器连接基站间的无线数据传输速率B_j, 及任务传输数据大小d_i决定, 如式 (1):

$$t_i^{\text{trans}} = \frac{d_i}{B_j} \tag{1}$$

(2) 考虑到每个边缘服务器的计算资源是有限的, 任务量较多的时候会出现任务卸载等待及拒绝现象. 接下来, 分析任务卸载时的平均等待时间和卸载成功率.

假设, 任务到达率为λ, 将 $M/M/s/N/\infty$多服务器排队模型[17] 运用于任务卸载中, 卸载任务到达边缘计算服务器后以 队列的形式保存, 遵循先到达先服务的准则. 系统负荷水平 ρ 如式 (2):

$$\rho = \frac{\lambda}{s \cdot \mu} \tag{2}$$

记 P_i 表示排队系统的状态概率, 其中, $0 \le i \le N+s$, 如式 (3):

$$P_i = \begin{cases} \dfrac{(s\cdot\rho)^i}{i!} \cdot P_0, & 0 \le i < s \\ \dfrac{(s\cdot\rho)^i}{(s!)\cdot s^{(i-s)}} \cdot P_0, & s \le i < N+s \end{cases} \tag{3}$$

根据正则性条件, 如式 (4):

$$\sum_{i=0}^{N+s} P_i = 1 \tag{4}$$

在 $\rho \neq 1$ 的情况下, 计算出状态概率 P_0, 如式 (5):

$$P_0 = \frac{1}{\sum\limits_{i=0}^{s-1} \frac{1}{i!} \cdot \left(\frac{\lambda}{\mu}\right)^i + \frac{1}{s!} \cdot \frac{1}{1-\rho} \cdot \left(\frac{\lambda}{\mu}\right)^s \cdot \left(1-\rho^{(N-s+1)}\right)} \tag{5}$$

根据 P_0 进一步计算得到队列中等待处理的任务的平均队长$\overline{L_q}$和平均等待时间$\overline{t_{\text{wait}}}$, 如式 (6)、式 (7), 其中平均等待时间根据排队论[18] 计算得到, 该参数表示从任务到达边缘计算服务器到被成功处理所需的平均等待时间.

$$\overline{L_q} = \frac{\left(\frac{\lambda}{\mu}\right)^s \cdot \rho \cdot P_0}{(s!)\cdot(1-\rho)^2} \cdot \left[1-\rho^{(N-s+1)} - (1-\rho)\cdot(N-s+1)\cdot\rho^{(N-s)}\right] \tag{6}$$

$$\overline{t_{\text{wait}}} = \frac{\overline{L_q}}{\lambda} \tag{7}$$

(3) 计算任务执行时间t_i^{job}: 由任务计算量c_i, 及边缘计算服务器的计算能力f_j^E决定, 如式 (8):

$$t_i^{\text{job}} = \frac{c_i}{f_j^E} \tag{8}$$

(4) 结果上传中心服务器时间: 由于边缘服务器所连接的坝区基站功率极大, 上传结果数据量相对较小, 因此选择对该部分时间忽略不计.

(5) 任务完成时间: 由任务数据传输时间t_i^{trans}、任务平均等待时间$\overline{t_{\text{wait}}}$和计算任务执行时间$t_i^{\text{job}}$共同组成, 如式 (9):

$$t_i^{\text{total}} = \left(\frac{d_i}{B_j} + \frac{\overline{L_q}}{\lambda} + \frac{c_i}{f_{l^i}^E}\right) \tag{9}$$

其中, l^i表示计算任务$Task_i$的卸载位置, $l^i \in \{1, 2, \cdots, M\}$, $l^i=M$ 即表示计算任务T_i在编号为 M 的边缘服务器执行计算, 则任意一个边缘服务器完成卸载任务的总时间如式 (10):

$$t_j^{ES} = \sum_{i \in O_{l^i}} t_i^{\text{total}} = \sum_{i \in O_{l^i}} \left(\frac{d_i}{B_j} + \frac{\overline{L_q}}{\lambda} + \frac{c_i}{f_{l^i}^E}\right) \tag{10}$$

那么系统任务完成时间为所有服务器中的最长运行时间, 如式 (11):

$$\begin{cases} \max\left(t_1^{ES}, t_2^{ES}, \cdots, t_M^{ES}\right) \\ \quad = \max\left(\sum\limits_{i \in O_1}\left(\frac{d_i}{B_j} + \frac{\overline{L_q}}{\lambda} + \frac{c_i}{f_1^E}\right), \sum\limits_{i \in O_2}\left(\frac{d_i}{B_j} + \frac{\overline{L_q}}{\lambda} + \frac{c_i}{f_2^E}\right), \cdots, \right. \\ \quad \left. \sum\limits_{i \in O_M}\left(\frac{d_i}{B_j} + \frac{\overline{L_q}}{\lambda} + \frac{c_i}{f_M^E}\right)\right) \\ \text{s.t. } i \in [1, Num],\ \sum\limits_{j=1}^{M} O_j = Num,\ j \in [1, M] \end{cases} \tag{11}$$

其中, $i \in O_j$表示在编号为 j 的边缘服务器上运行的所有计算任务.

任务卸载成功率表示为任务被边缘计算服务器成功处理并接受服务的概率, 如式 (12):

$$P_{\text{success}} = 1 - \frac{\left(\frac{\lambda}{\mu}\right)^N \cdot P_0}{(s!) \cdot \lambda^{(N-s)}} \tag{12}$$

为最小化计算任务完成时间, 将原问题转化为有约束的目标函数如式 (13):

$$\begin{cases} t_{\text{target}} = \min(\max(t_1^{ES}, t_2^{ES}, \cdots, t_M^{ES})) \\ \text{s.t. } C: P_{\text{success}} \geq P_{\text{success}}^* \end{cases} \tag{13}$$

其中, 约束C要求任务的卸载成功率超过目标约束值P_{success}^*.

5 基于模拟退火算法的任务卸载方法

基于 SA 算法最优化计算任务卸载位置, 即求目标函数 (13) 最优解的过程. 设置算法初始温度T_0、终止温度T_{fin}、温度衰减系数α以及每个温度下的迭代次数β. 并假设存在 n 个淤地坝同时产生一个计算信息不同的计算任务, 生成任务计算量矩阵C_{task}与传输数据量矩阵D_{data}, 分别如式 (14)、式 (15) 所示.

$$C_{\text{task}} = [c_1\ c_2\ \cdots\ c_i\ \cdots\ c_n]^{\text{T}},\ i = 1, 2, \cdots, n \tag{14}$$

$$D_{\text{data}} = [d_1\ d_2\ \cdots\ d_i\ \cdots\ d_n]^{\text{T}},\ i = 1, 2, \cdots, n \tag{15}$$

结合式 (1)、式 (8) 生成数据传输时间模型矩阵A_{trans}、任务执行时间模型矩阵A_{jobs}, 如式 (16)、式 (17) 所示.

$$A_{\text{trans}} = \begin{bmatrix} \frac{1}{b_1} & 0 & 0 & 0 \\ 0 & \frac{1}{b_2} & 0 & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \frac{1}{b_m} \end{bmatrix}, i = 1, 2, \cdots, m \tag{16}$$

$$A_{\text{jobs}} = \begin{bmatrix} \frac{1}{f_1^E} & 0 & 0 & 0 \\ 0 & \frac{1}{f_2^E} & 0 & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \frac{1}{f_m^E} \end{bmatrix}, i = 1, 2, \cdots, m \tag{17}$$

通过随机函数生成一个 $n \times m$ 的 0-1 矩阵记为初始解向量矩阵Z_{old}, 其中定义解向量矩阵$Z_{\text{old}} = (z_{ij})_{n \times m}$, $z_{ij} \in \{0, 1\}$, $\sum\limits_{i=1}^{m} z_{aj} = 1$, 其形式如式 (14), 向量矩阵 Z 表示所有设备的计算方式及任务卸载位置. 解向量$j = 1, 2, 3, \cdots, m$表示计算任务是否选择在边缘计算服务器 $1, 2, \cdots, m$ 上执行. z_{ij}=0 表示计算任务T_i不会选择该方式执行, z_{ij}=1 表示计算任务T_i选择该计算方式执行计算.

$$Z=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&0&0&0\\0&0&0&1&1\\0&0&1&0&0\end{bmatrix},z_{ij}\in\{0,1\},\sum_{j=1}^{M}z_{aj}=1 \quad (18)$$

式 (18) 表示任务 1 在编号为 1 的边缘服务器上运行, 任务 2 在编号为 2 的边缘服务器上运行, 任务 3 在编号为 5 的边缘服务器上运行, 任务 4、5 在编号为 4 的边缘服务器上运行.

结合线性代数相关内容与式 (11), 获取完成时间跨度矩阵T_{time}, 如式 (19).

$$\begin{aligned}T_{\text{time}}&=D_{\text{data}}*Z_{\text{old}}*A_{\text{trans}}+C_{\text{task}}*Z_{\text{old}}*A_{\text{jobs}}\\&=\left[t_1^{ES}\ \ t_2^{ES}\ \ \cdots\ \ t_i^{ES}\ \ \cdots\ \ t_M^{ES}\right]^T,i\in\{1,2,\cdots,M\}\end{aligned} \quad (19)$$

其中, t_i^{ES}, 表示第 i 个服务器完成卸载至此所有任务的运行时间, 令T_{time}矩阵中最大值$t_{\max}^{ES}$为系统任务完成时间t_{old}. 同时随机交换原解向量矩阵Z_{old}中 3 个元素的位置, 生成新的解向量矩阵Z_{new}, 并计算该解矩阵下的任务完成时间t_{new}. 并求出完成时间跨度差, 如式 (20).

$$\Delta t=t_{\text{new}}-t_{\text{old}} \quad (20)$$

若 $\Delta t<0$, 则一定接受新解Z_{new}, 若 $\Delta t>0$, 以 Metropoils 准则接受新解Z_{new}, 如式 (18) 接受新解Z_{new}. 判断此次迭代是否达到该温度下迭代次数上限, 若没有达到次数上限, 则重复进行位置交换与完成时间; 若达到该温度下的迭代次数上限, 则降低算法温度, 如式 (21).

$$T_0=T_0*\alpha \quad (21)$$

判断是否达到规定的最低温度; 若未达到最低温度则降低当前温度, 重置迭代次数, 并重复上述迭代过程, 若达到规定的最低温度, 则输出当前解向量矩阵, 算法结束. 如图 2 所示, 表示一次迭代过程.

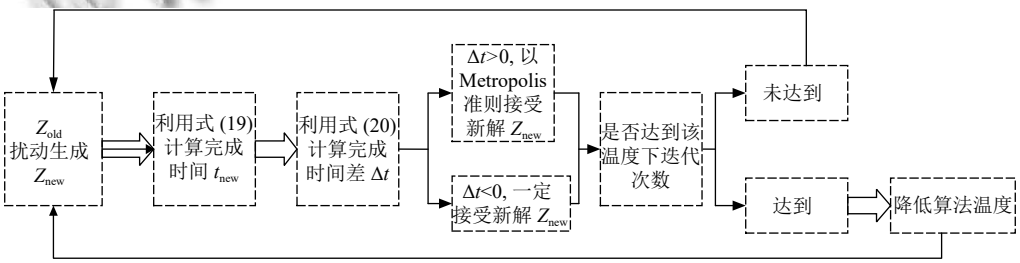

图 2 模拟退火算法制定任务卸载位置迭代流程

6 实验仿真与结果分析

6.1 性能评价标准

利用模拟退火算法建立计算任务调度策略的主要目的为降低计算任务完成时间即任务执行时间跨度, 使得系统的风险预警更具备时效性. 此外, 为综合评定算法性能, 增加任务总执行时间、计算任务调度策略制定时间以及系统负载均衡共同作为算法性能评价标准.

各指标的物理意义分别如下:

(1) 任务执行时间跨度: 即系统第一个任务开始至最后一个任务完成的时间跨度.

(2) 任务总执行时间: 即所有任务在边缘服务器上的执行时间总和.

(3) 计算任务调度策略制定时间: 即任务调度服务器收集到计算任务信息后, 制定所有计算任务卸载位置的总时间.

(4) 系统负载均衡: 评价一个网格计算平台是否分配均衡, 本文将采用各个节点的标准方差进行评价. 其评价公式如式 (22), 其中 M 表示边缘服务器个数, X_i表示第 i 个边缘服务器被调用的次数, $\overline{X_i}$表示所有边缘服务器平均被调用的次数, σ 表示系统负载标准方差, 其值越小, 表明系统负载越均衡.

$$\sigma=\sqrt{\frac{\sum_{i=1}^{M}(X_i-\overline{X})^2}{M}} \quad (22)$$

6.2 结果分析

实验中设置计算任务量c_i在区间 [5, 20] 随机生成, 传输数据量d_i在区间 [10, 50] 随机生成. 边缘服务器的计算能力在区间 [2, 3] 随机生成, 节点与边缘服务器的通信带宽B_j=1000. SA 算法初始温度T_0=200, 终止温度$T_{\min}$=3, 每个温度下的迭代次数为 30, 冷却率取 α=0.99.

(1) 不同任务数量下, SA 算法与 Min-Min 算法的

性能比较

为研究两种算法在任务数量不同时的性能差异. 实验边缘服务器数量设置为 50, 任务数量在区间 [100, 1000] 以 100 递增取值, 共计 10 组实验, 每组实验重复 10 次取平均值. 图 3 的 4 个子图分别表示两种算法在不同任务数量下的 4 种评价指标性能差异.

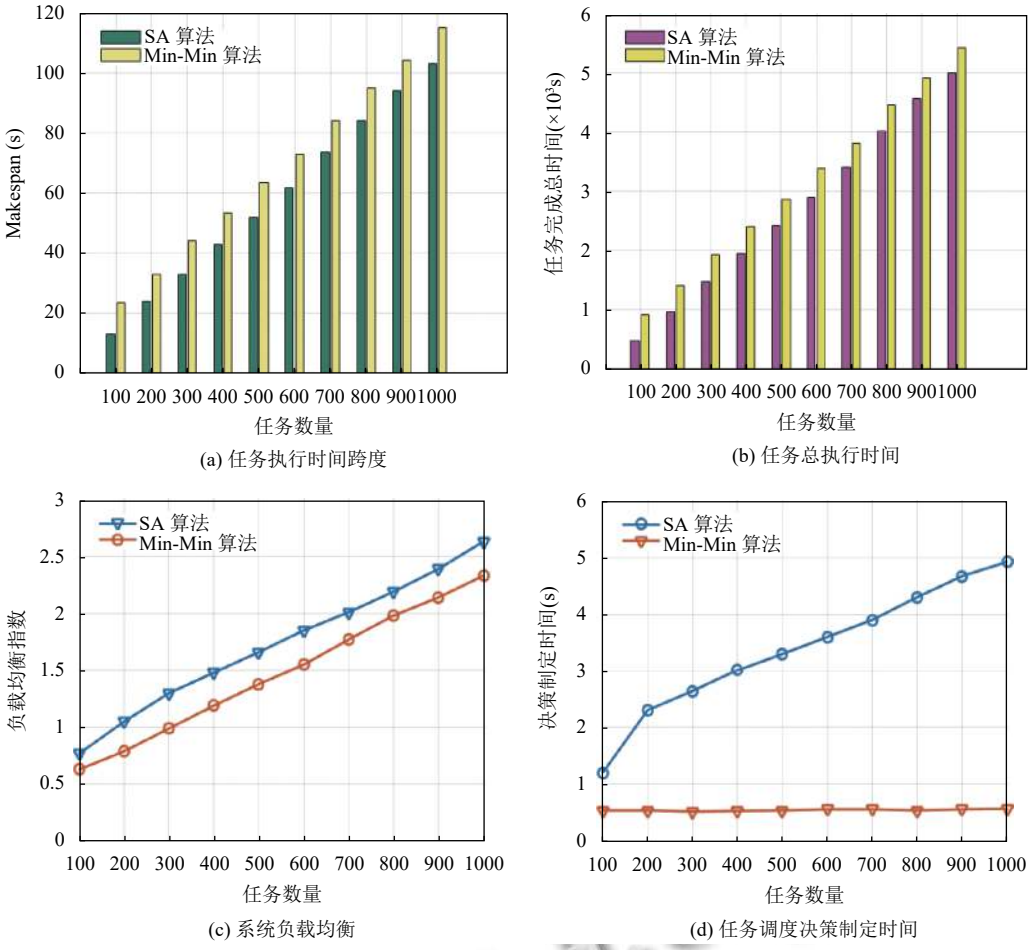

图 3 两种算法在不同任务数量下的综合性能

由图 3(a)、图 3(b) 可知, 采用 SA 算法进行任务调度, 可有效降低系统任务执行时间跨度、任务执行时间总和, 且当计算任务量越小, SA 算法求出的解越接近精确解. 其原因是算法初始化后, 迭代次数将被确定, 解矩阵越小, SA 算法越接近穷举法, 因此算法的时间性能越好. 由图 3(c) 可知, 为降低任务执行时间跨度, SA 算法将会为计算能力较强的服务器分配更多的计算任务, 因此其算法的负载均衡性能会有所损失. 由图 3(d) 可知, 虽然 SA 算法存在较为明显的时间性能优势, 但其任务调度策略制定时间相比于 Min-Min 算法也会增加, 且利用 SA 算法的制定调度决策的时间将呈现线性增加, 贴合 SA 算法的时间复杂度 O($kl(n)$), 其中 k 为每个温度下的迭代次数, $l(n)$ 为问题规模的多项式函数.

(2) 不同服务器数量下, SA 算法与 Min-Min 算法的性能比较

为研究两种算法在边缘服务器数量不同时的性能差异. 实验边缘服务器数量在区间 [30, 100] 以 10 递增取值, 共计 8 组实验, 每组实验重复 10 次取平均值. 图 4(a) 至图 4(d) 分别表示两种算法在不同服务器数量下的 4 种评价指标性能差异.

由图 4(a) 可知, 增加参与计算的边缘服务器可有效降低两种算法的任务执行时间跨度, 且边缘服务器的数量越多, SA 算法的性能优势将越明显, 其原因为:

在增加边缘服务器数量时, 虽然两种算法均会降低任务完成时间跨度, 但由于 SA 算法更加精确, 因此下降趋势较 Min-Min 算法更加明显. 由图 4(b) 可知, 当参与计算的服务器增加时, Min-Min 算法将会分配给计算能力较差的服务器分配计算任务, 因此任务执行时间总和将会增长. 通过图 4(c) 可知, 增加服务器数量会改善系统整体的负载均衡性能, 但本文采用 SA 算法制定任务调度策略的目的是降低任务执行时间跨度, 势必会提高计算能力较强服务器的负载, 因此系统负载均衡性能将会有所不足. 由图 4(d) 可知, 增加边缘服务器数量将会扩大 SA 算法解矩阵规模, 提高 SA 算法时间复杂度, 导致求解时间增加.

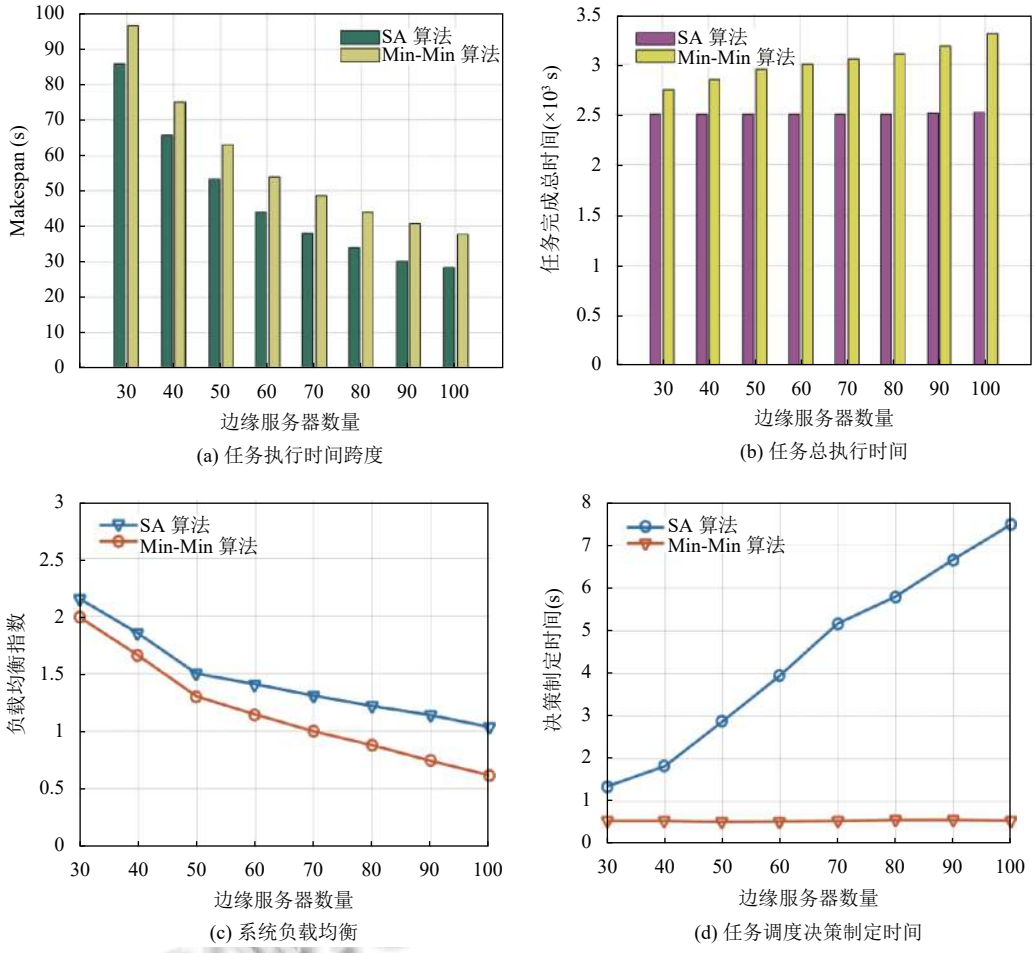

图 4 两种算法在不同服务器数量下的综合性能

7 结论与展望

针对淤地坝监测与预警系统时效性问题, 设计了基于模拟退火的多个边缘计算服务器相互协作任务调度方法. 根据任务调度分析, 建立任务完成时间模型. 并说明了在淤地坝计算任务的执行没有优先级时, 应用模拟退火算法制定调度策略可以达到该场景下降低完成时间跨度的任务调度目的, 并给出了调度优化问题的解决步骤. 最后通过仿真实验, 验证了基于模拟退火的调度方法符合场景需求, 可以达到预期的效果.

参考文献

1 高雅玉, 田晋华. 淤地坝建设研究进展分析. 中国科技成果, 2019, 20(9): 34–37. [doi: 10.3772/j.issn.1009-5659.2019.09.017]

2 党维勤, 郝鲁东, 高健健, 等. 基于“7·26”暴雨洪水灾害的淤地坝作用分析与思考. 中国水利, 2019, (8): 52–55. [doi: 10.3969/j.issn.1000-1123.2019.08.020]

3 Attiya I, Elaziz MA, Xiong SW. Job scheduling in cloud computing using a modified Harris hawks optimization and simulated annealing algorithm. Computational Intelligence

and Neuroscience, 2020, 2020: 3504642.
4 顾艳玲, 顾冲时, 李波. 大坝安全综合评价多指标权重确定方法研究. 水电能源科学, 2008, 26(4): 88–90, 94. [doi: 10.3969/j.issn.1000-7709.2008.04.028]
5 黄黎明, 张可, 龚寻, 等. 大数据视角下水利工程质量风险管理. 水利经济, 2017, 35(6): 66–70, 82. [doi: 10.3880/j.issn.1003-9511.2017.06.013]
6 王乃欣, 王志坚, 梁小卫. 基于虚拟现实的淤地坝溃决模拟. 水资源与水工程学报, 2017, 28(5): 162–167. [doi: 10.11705/j.issn.1672-643X.2017.05.27]
7 黄肖. 基于 MIKE 模型的淤地坝系布局对流域洪水过程的影响. 东北水利水电, 2019, 37(1): 50–52.
8 李平, 黄跃飞, 李兵. 基于贝叶斯网络的梯级水库连溃风险. 水科学进展, 2018, 29(5): 677–684.
9 Seid M, Anokye S, Sun GL. Machine learning based unmanned aerial vehicle enabled fog-radio access network and edge computing. ZTE Communications, 2019, 17(4): 33–45.
10 Liu M, Mao YM, Leng SP. Distributed caching in information-centric cellular networks with full duplex communication. IET Communications, 2019, 13(2): 223–231. [doi: 10.1049/iet-com.2018.5130]
11 Zhou FH, Wu YP, Hu RQ, *et al*. Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems. IEEE Journal on Selected Areas in Communications, 2018, 36(9): 1927–1941. [doi: 10.1109/JSAC.2018.2864426]
12 Liao YZ, Shou LQ, Yu Q, *et al*. Joint offloading decision and resource allocation for mobile edge computing enabled networks. Computer Communications, 2020, 154: 361–369. [doi: 10.1016/j.comcom.2020.02.071]
13 Anousha S, Ahmadi M. An improved min-min task scheduling algorithm in grid computing. 8th International Conference Grid and Pervasive Computing. Seoul, Republic of Korea. 2013. 103–113.
14 Xhafa F, Kilic B, Krause P. Evaluation of IoT stream processing at edge computing layer for semantic data enrichment. Future Generation Computer Systems, 2020, 105: 730–736. [doi: 10.1016/j.future.2019.12.031]
15 张开元, 桂小林, 任德旺, 等. 移动边缘网络中计算迁移与内容缓存研究综述. 软件学报, 2019, 30(8): 2491–2516. [doi: 10.13328/j.cnki.jos.005861]
16 Zhang DE, Li WL, Wu XH, *et al*. Application of simulated annealing genetic algorithm-optimized Back Propagation (BP) neural network in fault diagnosis. International Journal of Modeling, Simulation, and Scientific Computing, 2019, 10(4): 1950024. [doi: 10.1142/S1793962319500247]
17 游卓浩. 基于 M/M/n 排队模型的云资源调度策略研究 [硕士学位论文]. 成都: 电子科技大学, 2014.
18 Zukerman M. Introduction to queueing theory and stochastic teletraffic models. http://dl.icdst.org/pdfs/files/b5e99f272c56fd2b21a52c4d0d03be3e.pdf, [2021-01-24].