第58卷 第1期
2022年1月

南京大学学报(自然科学)
JOURNAL OF NANJING UNIVERSITY
(NATURAL SCIENCE)

Vol. 58, No. 1
Jan., 2022

DOI:10.13232/j.cnki.jnju.2022.01.015

面向IoT数据工作流的分割与调度方法

秦生辉[1,2],赵卓峰[1,2*],杨中国[1,2]

(1.北方工业大学信息学院,北京,100144;2.大规模流数据集成与分析技术北京市重点实验室,北京,100144)

摘 要:物联网数据是当前一类典型的大数据,其应用正成为诸多行业领域的热点,围绕物联网数据的应用往往可以被表示为由一组大数据处理与分析任务构成的工作流.与传统工作流不同的是,IoT(Internet of Things)环境下这种数据驱动的工作流具有数据来源分散、数据规模大、云边协同分布执行等特点,给IoT数据工作流的执行带来了数据流控制管理、数据传输调度等方面的诸多挑战.针对IoT数据工作流的执行约束和数据传输优化问题,提出一种面向IoT数据工作流的分割与调度优化方法.首先对IoT数据工作流的执行约束条件、边缘节点负载以及数据传输量进行建模,进而以数据传输和执行时间优化为目标设计一种云边架构下IoT数据工作流的分割算法和子工作流执行调度算法.通过基于WorkflowSim的仿真实验结果表明,提出的算法与典型的HEFT和MINMIN算法相比,可以在保障边缘节点执行约束和负载均衡的条件下有效降低IoT数据工作流的执行时间.

关键词:IoT工作流,分割调度,数据传输优化,多目标优化,物联网服务

中图分类号:TP301 **文献标志码**:A

Segmentation and scheduling method for IoT data workflow

Qin Shenghui[1,2], Zhao Zhuofeng[1,2*], Yang Zhongguo[1,2]

(1. Institute of Information Technology, North China University of Technology, Beijing, 100144, China;
2. Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, Beijing, 100144, China)

Abstract: IoT (Internet of Things) data is a typical type of big data at present, and its application is becoming a hot spot in many industries. Applications surrounding IoT data can often be expressed as a workflow composed of a set of big data processing and analysis tasks. Differing from traditional workflows, this data-driven workflow in the IoT environment has the characteristics of scattered data sources, large data scale, and cloud-side collaborative distributed execution, which brings data flow control management, data transmission scheduling and many other challenges to the execution of IoT data workflows. Therefore, this article proposes a segmentation and scheduling method for IoT data workflow execution with considering the execution constraints and data transmission optimization problems. The method first models the execution constraints, edge node load and data transmission volume of IoT data workflow in a coherent way. And then, a segmentation algorithm for IoT data workflow and a sub-workflow execution scheduling algorithm under cloud-edge architecture with the goal of data transmission and execution time optimization are designed. The results of simulation experiments based on WorkflowSim show that compared with the typical HEFT and MINMIN algorithms, our algorithm effectively reduce the execution time of IoT data workflow under the conditions of guaranteeing edge node execution constraints and load balancing.

Key words: IoT workflow, split scheduling, data transmission optimization, multi-objective optimization, Internet of Things services

基金项目:国家重点研发计划(2018YFB1402500),国家自然科学基金国际合作与交流项目(62061136006),北京市自然科学基金(4202021)

收稿日期:2021-09-22

*通讯联系人,E-mail:edzhao@ncut.edu.cn.

随着物联网建设的发展,逐渐产生并积累了大量各种各样的物联网数据,围绕这些数据的各类智能应用成为当前的热点.在物联网场景下,传感器数据、采集的实时视频数据等需要经过多个处理任务最终为智能决策服务.这些处理任务涉及数据采集、分布式数据查询和个性化数据分析等任务,这些任务可以以工作流的形式进行组织表示,这种形式的工作流被称为IoT(Internet of Things)数据工作流[1].同时,由于物联网应用系统网络拓扑分散、数据传输代价高等特点以及低处理时延和一定的数据隐私保护等需求,如何借助云边协同的方式对IoT数据工作流任务进行有效地划分和调度就成为此类IoT应用的关键.

区别于传统工作流,针对IoT数据处理的工作流具有诸多新的特点:IoT数据工作流的执行环境不再是集中的节点或集群,而是需要在IoT云边端网络架构下进行执行调度;IoT数据工作流任务涉及大量数据采集和处理工作,任务之间的数据流流量也较大;参与IoT数据工作流执行的节点资源及负载能力也不同.这些都给IoT数据工作流执行调度的研究带来了新的挑战[2].

例如,在智能电网业务中,终端传感器及监控设备不断产生大量数据,尤其是终端设备,产生的数据量大且有效信息较少,在传输过程中还会占用大量网络资源[3].而在大多数业务场景下,由于数据源等业务约束条件,部分任务需要指定节点执行,云边协同模式给这种业务场景提供了较好的执行环境.为了严格遵守约束条件,流程分割是一个有效的方法,在此基础上进行优化,可以有效地避免大量数据传输造成的时延.

为了应对上述挑战,一个基本的思路就是对IoT数据工作流进行有效切分并调度到合适的IoT节点执行,但将工作流切分成任务进行调度的方式往往会破坏工作流任务执行的关联依赖,而实际应用中往往存在一些业务约束[4],规定工作流的局部连续任务必须调度特定节点执行.所以,这部分任务应以工作流的形式进行调度执行,这样既可以保证局部连续任务的工作流模型不受破坏,保障任务之间的关联关系,又可以遵循业务约束将其调度到特定节点.基于这种方法的优势,工作流分割的概念被提出,即约束条件下的工作流分割.

基于上述思路,本文针对IoT数据工作流的执行约束和数据传输优化问题,提出一种面向IoT数据工作流的分割与调度优化方法.首先对IoT数据工作流的执行约束条件、边缘节点负载以及数据传输量进行建模,进而以数据传输和执行时间优化为目标设计一种云边架构下IoT数据工作流的分割算法和子工作流执行节点调度算法.算法将约束条件下的工作流依据数据传输量进行优化分割得到子工作流,进而对于子工作流根据节点资源及负载和业务约束情况设计调度算法,从而优化IoT数据工作流在云边协同模式下的数据传输、执行时间及执行节点负载.

1 相关工作

近年来,IoT数据量飞速增长,与其相关的工作流调度也成为一个非常重要的问题,许多学者对工作流调度进行了大量研究.文献[5—7]考虑用户对于时间的约束、服务质量的约束以及隐私保护,通过多目标优化提高了服务质量.Chen et al[8]提出一种方法,可以为工作流的每个任务寻找最合适的资源以满足用户需求.Dong et al[9]在任务优先级约束条件下实现了执行时间的最小化.陈俊宇和刘茜萍[10]提出一种云环境下基于阶段划分的数据密集型工作流调度方法,基于数据依赖将工作流划分为多个阶段,然后对任务进行调度,优化工作流的执行时间.李万清等[11]提出面向安全和能耗感知的服务工作流调度方法,在降低移动应用调度风险率的同时最小化设备能耗.王柳婧等[12]提出一种基于多约束图分割的工作流调度算法,通过有向边的修剪,在所有维度上实现权重和的均衡,得到最小化的任务间数据传输量,降低通信代价.刘晓霞和李芳[13]提出一种期限分割的工作流调度代价优化算法,将工作流任务的调度过程划分为四个阶段:工作流分层、期限分割、任务选择和实例选择,降低工作流执行代价.薛凡[14]提出一种基于有向无环图分割的工作流调度算法,分割阶段确保任务的执行顺序依赖并对任务进行重分配,有效优化调度效率和调度代价.Pei et al[15]提出一种基于云异构平台的数据密集型工作流调度采用图划分和强化学习的云调度算

法,通过图划分算法将数据依赖性强的任务聚类成块以降低任务执行过程中的成本和时间开销.

以上研究大多考虑工作流的执行时间成本和诸多约束条件并以任务为实例进行调度,优化数据传输时延,也有效降低了移动边缘环境下工作流任务调度的隐私泄露风险.同时,也提出了一些基于约束工作流任务依赖和数据依赖的分割调度算法,但多是基于单目标的分割优化,对数据密集型且具有多约束条件的工作流并不适用.

工作流的本质是一种自动化的执行流程,其任务与任务间有相应的依赖关系,IoT数据工作流的调度约束往往在某几个相邻依赖的任务之间,即此部分关联任务有相同的业务约束和数据源依赖.因此本文提出一种IoT数据工作流分割调度方法,基于资源约束、业务约束和数据传输进行分割优化,并根据业务约束、节点资源及节点负载等进行调度优化,在保证业务约束的同时通过工作流分割的方法将服务调度之间的数据传输最小化,并保留工作流的局部依赖关系;然后根据约束条件对分割后的工作流进行调度优化,使节点负载保持均衡的同时最小化工作流的执行时间.

2　研究内容

针对目前IoT数据工作流调度领域存在的问题,本文将基于分割和调度两方面进行研究,最终通过实验证明该研究方案的有效性.整体的方案架构如图1所示.

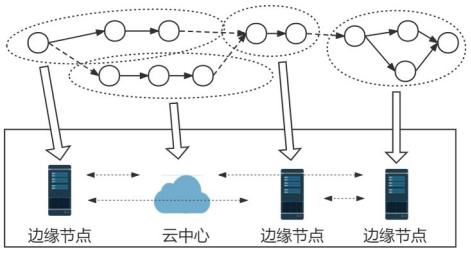


图1　IoT数据工作流分割调度方案

Fig. 1　IoT data workflow segmenting and scheduling scheme

2.1　IoT数据工作流分割　基于任务的调度和基于工作流的调度有很大的不同.IoT数据工作流的部分任务有一定的业务约束,如部分连续任务由于执行环境约束或者为了保护一些隐私数据需要在特定节点执行,在这种情况下,工作流的部分连续任务需要调度到同一节点执行.若基于任务调度,在调度过程中就需要考虑每个任务的开始时间、结束时间以及两个连续任务之间的数据传输时间,以保证各个任务执行的先后关系;若基于工作流调度,这部分任务构成局部子工作流,在调度过程中,只需考虑此子工作流的开始时间以及结束时间,从而既满足约束条件又可以保证此部分任务的关联关系.

本文首先根据业务约束条件确定工作流分割位置的初始值,然后对工作流的分割位置进行优化.任务之间的数据传输量是一个重要的考虑因素,各个子工作流之间的数据传输量将极大影响后继工作流的开始时间,进而影响整个工作流的执行时间,故以各任务之间的数据传输量作为分割位置的优化目标,在约束条件下基于任务之间的数据传输量进行优化,得到工作流的最优分割位置,以确保约束条件下各个子工作流之间的数据传输量最小.

2.2　子工作流调度　IoT数据工作流的执行环境主要有两种:边缘协同模式和云边协同模式.边缘协同模式指应用在某边缘服务器提出形成工作流,由于边缘节点的位置优势,工作流在附近边缘节点调度执行;云边协同模式指应用在云端提出,不同业务可以调度到特定边缘节点执行,而对于业务量较大或者计算资源需求较高的子工作流可以在云中心执行.总体上,子工作流的调度基于业务约束,该业务约束指子工作流必须调度到特定的某个节点或某几个节点,在此约束的基础上需要考虑节点资源及负载,将子工作流优先调度到资源能力较高、负载较小的节点上执行.

2.3　总结　通过以上分析,得到如下结论:

(1)IoT数据工作流的分割主要考虑以约束条件下工作流各任务之间的数据传输量作为分割位置的优化目标.

(2)子工作流的调度主要考虑约束条件下子工作流执行节点的计算资源及负载.

本文考虑IoT数据工作流的分割调度优化问题,通过WorkflowSim仿真平台进行实验,证明该方法的可行性及有效性.

3 IoT 数据工作流系统架构及相关定义

3.1 系统架构 如图 2 所示,在物联网环境下,IoT 数据工作流系统涉及的数据存储和计算的主要参与者为云计算中心服务器集群和边缘计算节点,云中心和边缘节点均由多个虚拟机组成,其中边缘节点的计算能力有限,但 IoT 设备接收的数据可直接存储在最近的边缘服务器上,此数据传输时延可忽略不计,暂且认为边缘节点可以满足其附近 IoT 设备所接收数据的存储容量. 此外,对于同一业务场景,其多个边缘节点距离较近,故其之间的数据传输时延较小. 云计算中心服务器集群具有较强的计算能力,可以满足较大规模的计算任务,但边缘 IoT 设备的数据传输至云中心需要较高的传输开销.

图 2 物联网云边系统架构

Fig. 2 IoT cloud-edge system architecture

3.2 相关定义

3.2.1 任务间的数据传输 分割前的 IoT 数据工作流采用有向无环图 DAG (Directed Acyclic Graph)描述,记为 ODAG,如图 3 所示. 记为 $OG=(T,E)$,其中,T 为任务顶点的集合,$T=\{T_1,T_2,T_3,\cdots,T_n\}$ 表示云工作流由 n 个依赖任务组成;E 为有向边的集合,$E=\{(T_i,T_j,Data_{ij}):(T_i,T_j)\in T\}$.$(T_i,T_j,Data_{ij})$ 表示任务和任务之间的依赖关系,任务 T_i 是任务 T_j 的前驱任务,任务 T_j 是任务 T_i 的后继任务,$Data_{ij}$ 表示任务 T_i 传递给任务 T_j 的数据量大小.

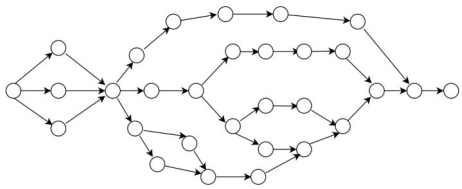


图 3 分割前的 IoT 数据工作流 DAG

Fig. 3 IoT data workflow DAG before segmentation

3.2.2 子工作流之间的数据传输 在 IoT 数据工作流分割之后形成子工作流,也采用有向无环图描述,记为 SDAG,即将每个子工作流作为一个顶点,将各个子工作流之间的依赖关系作为一条边,从而将 ODAG 化简,如图 4 所示. 记为 $OG=(P,R)$,其中,P 为子工作流的集合,$P=\{P_1,P_2,P_3,\cdots,P_m\}$ 表示原工作由 m 个子工作流组成;R 为表示各子工作流之间的依赖关系的边的集合,$R=\{(P_i,P_j,Data_{ij}):(P_i,P_j)\in P\}$. $(P_i,P_j,Data_{ij})$ 表示各子工作流之间的依赖关系,工作流 P_i 是工作流 P_j 的前驱工作流,工作流 P_j 是工作流 P_i 的后继任务,$Data_{ij}$ 表示工作流 P_i 传递给工作流 P_j 的数据量大小.

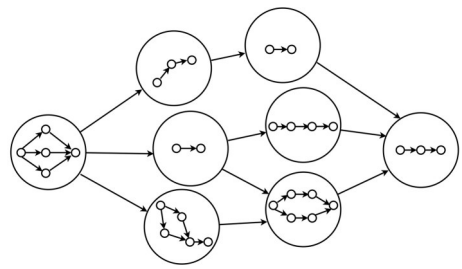


图 4 分割后的 IoT 数据工作流 DAG

Fig. 4 Segmented IoT data workflow DAG

3.2.3 执行时间 分割之前的 IoT 数据工作流的执行时间取决于其执行环境模式、子工作流之间的数据传输量和节点资源计算能力.

每个子流程都会被分配到某个云中心或边缘节点的虚拟机,每个子流程都有执行的开始时间和结束时间,记为 $sched(P_i)=(vm_j,SP_{P_i},EP_{P_i})$,其中,$vm_j$ 表示子工作流 P_i 分配到 vm_j 类型的虚拟机上执行,SP_{P_i} 表示子工作流 P_i 的开始时间,EP_{P_i} 表示子工作流 P_i 的结束时间. 子工作流 P_i 的开始

时间SP_{P_i}由P_i所有的前驱子工作流执行完毕后将数据传输到子工作流P_j所在虚拟机的时间决定,子工作流P_i的结束时间EP_{P_i}等于P_i的执行时间加上其开始时间SP_{P_i},如式(1)和式(2)所示:

$$EP_{P_i}=SP_{P_i}+\frac{work(P_i)}{P_{vm_j}} \tag{1}$$

$$SP_{P_i}=\max\left\{EP_{P_h}+\frac{Data(P_{hi})}{b}:P_h\in pre(P_i)\right\} \tag{2}$$

EP_{P_h}为前驱工作流P_h的结束时间,$\frac{Data(P_{hi})}{b}$为数据从P_h传输到P_i的时间.b为节点之间的数据传输速率:边缘协同模式下数据传输速率较高,即b较大;云边协同模式下由于节点距离较远,数据传输速率较低,即b较小.

$work(P_i)$为工作流P_i的指令数,P_{vm_j}为vm_j每秒处理的指令数.从第一个子工作流开始执行到最后一个子工作流完成的时间段称为整个IoT数据工作流的完成时间,因而整个IoT数据工作流的完成时间如式(3)所示:

$$makespan=\max\left\{EP_{P_h}:P_i\in P\right\} \tag{3}$$

4　IoT数据工作流分割调度方法

对于IoT数据工作流,通过基于约束条件下的分割调度方法,实现最小化IoT数据工作流执行时间的目标.本节首先对于执行环境及约束条件进行详细分析,抽象建模,形成系统模型和分割调度的约束模型;其次使用粒子群优化算法设计IoT数据工作流调度算法解决多目标优化问题.

4.1　IoT数据工作流分割调度问题　IoT数据工作流的调度具有诸多约束和可优化点,以下将给出具体模型.

4.1.1　业务约束　业务约束指数据隐私保护约束,即受约束任务的数据源或运行场景固定,必须在指定的一个或多个节点上执行.本文在业务约束的基础上进行IoT数据工作流的分割优化和调度优化.

将调度范围的位置进行独热编码,每个地理位置都是一个长度为N的串,N为所有的位置个数,该串中只有一个数为1,其他数均为0,如图5所示.在这种表示方式下,如果需要数据的位置决定计算任务的位置,就必须满足式(4):

$$location_{\text{data}}\cdot location_{\text{task}}=1 \tag{4}$$

其中,$location_{\text{data}}$是受到约束的数据集的位置向量,$location_{\text{task}}$是对应该数据处理任务的位置向量.

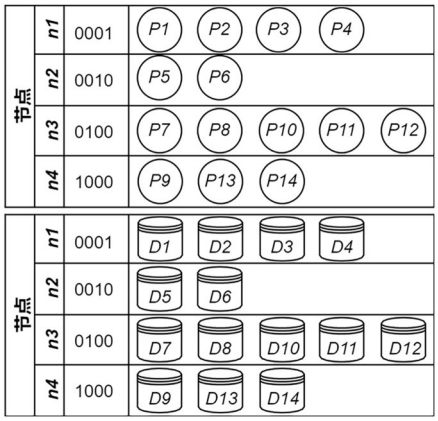


图5　业务约束规则

Fig. 5　Business constraint rules

4.1.2　节点资源　边缘节点具有离数据源近的特点,但其计算能力有限,节点资源的计算能力是影响子工作流调度的重要因素.考虑边缘节点的计算能力不能满足某些子工作流的执行,表示为$R_{P_k}\leqslant C_i$,如图6所示.

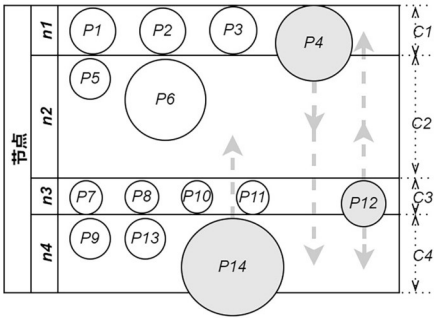


图6　节点资源约束规则

Fig. 6　Node resource constraint rules

4.1.3　节点负载　云中心集群和边缘节点集群作为IoT数据工作流的执行载体,必须保证其稳定正常运行,节点负载均衡可有效保障整个系统的正常平稳运行,所以在调度过程中保证集群各节点的负载均衡对于工作流执行的稳定性具有重要意义.节点负载如图7所示.

计算资源节点j(j=1,2,…,K,K为资源节点总数)上分配的子工作流数量来表示资源节点

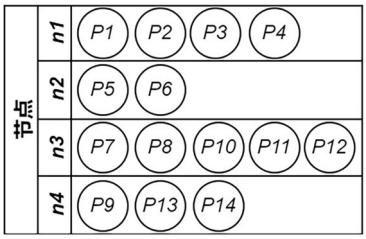


图 7 节点负载示意图

Fig. 7 Node load balancing diagram

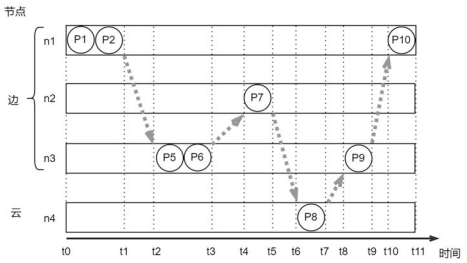


图 8 IoT 数据工作流执行时间示意图

Fig. 8 IoT data workflow execution time diagram

执行子工作流的频度,用 N_j 表示. 假定每个集群中的各个节点计算能力相同,则单个集群的资源节点利用率如式(5)所示:

$$r=\frac{\sum_{j=1}^{k}N_j}{K\cdot\max\limits_{j\leqslant K}N_j}\times 100\% \tag{5}$$

这里采用均衡度来表示. 在优化的过程中,需要让 r 尽可能靠近 1.

此外,本文使用多个集群(云中心集群及各个边缘集群)的平均负载 $\frac{r}{N}$(N 为集群数量)作为优化目标.

4.1.4 数据传输时延 由于云中心集群和边缘节点位置分布不一致,云中心距离边缘节点较远,同一业务场景的边缘节点之间距离较近. 计算方式如式(6)所示:

$$T_{hi}=\begin{cases}0 & l_h^i=c_1\\ \dfrac{Data\left(P_{hi}\right)}{b} & l_h^i=c_2\\ d+\dfrac{Data\left(P_{hi}\right)}{b} & l_h^i=c_3\end{cases} \tag{6}$$

其中,

c_1:子流程 P_h 与 P_i 部署在相同的边缘节点;

c_2:子流程 P_h 与 P_i 分别部署在不同的边缘节点;

c_3:子流程 P_h 与 P_i 分别部署云端及边缘节点.

d 为受节点距离影响而增加的额外的传输时延,本文认为边缘节点之间无额外传输时延. 图 8 为数据传输时延及执行时间示意图.

4.2 分割算法 业务约束是分割位置优化的前提,当任务的 $location_{\text{task}}$ 相等时,则这些任务属于同一个约束集合. 在约束条件的基础上给出以下分割优化算法,根据数据传输量对业务约束集合进行进一步优化,得到新的任务集合.

算法步骤如下:

(1)将与约束任务有关联关系的无约束任务并入约束任务集合:将每个约束任务集合中的内部数据传输量的平均值作为一个数据传输量的阈值 W,从约束任务集合的任务遍历 ODAG 中的无约束任务,若两个相邻任务之间的数据传输量大于该阈值 W,则将后继任务并入该约束任务集合,并继续遍历,若小于 W,则此次遍历结束. 从下一个约束任务集开始遍历,重复以上步骤.

(2)未并入的有关联关系的无约束任务组成新的任务集合:从 ODAG 的开始节点遍历,将连续的无约束任务组成新的任务集合.

(3)在任务集合中进一步优化分割位置:若该任务集合允许被调度到多个服务节点,则可以进一步根据阈值 W 将该集合分成多个.

4.3 子工作流执行调度方法 本文的研究目标是最小化 IoT 数据工作流的执行时间,并且使各个集群的节点利用率趋于均衡.

4.3.1 多目标优化调度 一般地,多目标优化问题是在多种约束条件下,为求解的多目标函数寻找一组决策变量值和目标函数值[16]. 比如,一个多目标最小化问题包括 n 个决策变量、r 个优化目标以及 m 个约束条件,就可以表示为式(7):

$$\begin{cases}\min f(x)=\left[f_1(x),f_2(x),\cdots,f_n(x)\right]\\ g_i(x)\geqslant 0\quad (i=1,2,\cdots,k)\\ h_j(x)=0\quad (j=1,2,\cdots,l)\end{cases} \tag{7}$$

其中,r 表示需要优化的目标数量. 本文的调度优化中仅有一个优化目标:IoT 数据工作流执行时间. 问题中的每个解决方案 x 都包含两个变量:一个变量包含 n 个值 $\left(x_1,x_2,\cdots,x_n\right)$,$n$ 是一个 IoT

数据工作流实例中分割之后的子流程个数$(n=|P|)$,$x_i=S(P_i)$表示调度虚拟资源x_i上的子工作流$(P_i\in P)$;另一个变量是大小为n的数列,表示子工作流被调度的顺序.g_i和h_j是问题的约束条件.如果x均达到这些约束条件的要求,就可以将这个x记为其一个可行解.因此,问题的可行解集合可以表示为式(8):

$$\Omega=\{x\mid g_i(x)\geqslant 0(i=1,2,\cdots,k),\ h_j(x)=0(j=1,2,\cdots,l)\} \tag{8}$$

在此前的分割优化过程中已将具有相同约束条件的任务优化分割到同一子工作流中,因此在调度过程中,子工作流的约束条件即为它们指定服务节点进行调度.则调度优化算法的目标可以表示为式(9):

$$\begin{cases} f(x)=\min(Tw_{\text{total}},Sl_{\text{total}}) \\ S(p_i)=r_m \quad p_i\in P,r_m\in dc_p \end{cases} \tag{9}$$

其中,Tw_{total}和Sl_{total}指算法的优化目标,即IoT数据工作流的执行时间和系统的平均负载;$S(p_i)$是算法的约束条件,p_i和dc_p分别是具有业务约束的子工作流和为其指定的服务节点,r_m是服务节点dc_p上子工作流p_i所需要的资源.

4.3.2　基于粒子群的子工作流调度算法　为了解决上述多目标优化问题,采用粒子群优化(Partical Swarm Optimization,PSO)算法[17-19].PSO采用群体智能搜索模式在搜索空间上逐步寻找解决优化问题的一个解决方案,通过该算法将得到子工作流和虚拟机的最佳映射.

类似其他进化算法,在PSO算法中,种群是求解空间中全部的粒子.粒子随机初始化,每个粒子都由一个适应值表示.在每一代优化时,适应值都由一个适应值函数进行评估,本文的适应值如式(10)所示:

$$Fitness=(rt\times 5+1000\times(1-al)\times 5)/10 \tag{10}$$

其中,rt为执行时间,al为平均负载.

每个粒子知道它自己的最佳位置P_{best}和到目前为止整个种群中最佳的粒子位置G_{best},因此在搜索过程中粒子总是趋向更好的搜索区域移动.一个粒子的P_{best}是该粒子迄今达到的最佳位置(适应值),G_{best}是对整个种群而言的最佳位置.粒子具有引导粒子移动的速度和评估粒子适应值的位置,每一次迭代过程中,粒子会通过追踪两个极值来更新自己.在每一代中,粒子的速度和位置会根据式(11)和式(12)更新:

$$V_i^{k+1}=\omega V_i^k+c_1r_1(P_{\text{best}\,i}-X_i^k)+c_2r_2(G_{\text{best}}-X_i^k) \tag{11}$$

$$X_i^{k+1}=X_i^k+V_i^{k+1} \tag{12}$$

其中,V_i^{k+1}是粒子i在第$k+1$次迭代时的速度,V_i^k是粒子i在第k次迭代时的速度,ω是惯性权重,c_1和c_2是加速系数,r_1和r_2是0到1之间的随机数,X_i^k是粒子i在第k次迭代时的位置,X_i^{k+1}是粒子i在第$k+1$次迭代时的位置,$P_{\text{best}\,i}$是第i个粒子目前为止达到的最好位置,G_{best}是对整个种群而言最佳的粒子位置.

算法过程如下:

(1)在搜索空间中,随机初始化粒子种群,每个粒子有自己的速度和位置.

(2)计算所有粒子的适应值,设置每个粒子的$P_{\text{best}\,i}$等于它当前的位置,设置G_{best}等于最好的初始粒子.

(3)根据式(11)和式(12)更新每个粒子的速度和位置.

(4)将每个粒子的位置映射到求解空间并根据适应值函数式(10)评估粒子的适应值.

(5)比较每个粒子当前适应值和其个体最佳适应值P_{best}.如果当前适应值更好,就更新P_{best}为当前的粒子.

(6)选取当前整个种群中拥有最佳适应值的粒子,如果当前粒子的适应值比G_{best}好,那么更新G_{best}为当前粒子.

符合停止条件输出G_{best}和它的适应值,否则,跳转到步骤(2)继续执行.正如式(11)和式(12)所示,最初的PSO用来解决连续的优化问题,但是任务分配问题是离散的,因此,速度更新的计算改为式(13):

$$V_i^{k+1}=\left\lfloor \omega V_i^k+c_1r_1(P_{\text{best}\,i}-X_i^k)+c_2r_2(G_{\text{best}}-X_i^k)\right\rfloor \tag{13}$$

其中,粒子速度的范围为$[-vmNum,vmNum]$,$vmNum$为虚拟机的数量,若速度更新后超出此范围则取其边界值;粒子位置的范围为

$[0, vmNum-1]$,若位置更新后超出此范围则取其边界值.

5 实验评估

5.1 实验设置 实验模拟IoT数据工作流在云边环境下的调度问题,模拟四个边缘节点以及一个云计算中心,并与工作流调度领域的经典算法HEFT算法和MINMIN算法[20]作对比,以验证本文优化方法的正确性及有效性.

5.2 平台 实验设计和算法实现基于WorkflowSim仿真平台,该仿真平台在CloudSim平台的基础上增加了工作流应用的资源配置和任务调度的支持.算法实现和相关对比算法的实现用Java编程语言完成.所有实验在同一配置(Intel(R)Core(TM)i7-9750CPU@2.60 GHz,16 G内存)的Windows10机器上进行.

5.3 WorkflowSim中数据中心和虚拟机配置 实验在五个不同地理位置的数据中心下进行,前四个数据中心代表四个边缘计算节点,剩下的一个数据中心代表云计算中心.前四个数据中心每个数据中心配置一台主机,第五个数据中心配置三台主机,如表1所示.边缘节点及云计算中心的参数设置如表2所示.

同时,实验还配置了三种类型的虚拟机,每种类型10个,随机分配给五个数据中心.虚拟机的参数配置如表3所示.

5.4 工作流实例配置 仿真实验采用WorkflowSim平台提供的工作流实例,包含两种形式的数据工作流,如图9所示.每种包含四种不同任务数量的工作流 $Qw\in\{30, 50, 100, 1000\}$,在这两个工作流的基础上根据IoT数据工作流特征进行了扩展,为每个任务设置了随机的约束条件,使该任务在某服务节点上具有数据和业务约束.约束规则见4.1.1.

5.5 结果分析 通过对比实验可以发现,对于第一种形式的工作流,其各个任务之间的数据依赖较复杂,所以在具有业务约束时,本文的分割调度优化方法对工作流执行时间的优化效果比HEFT算法更明显,与MINMIN算法效果较为接近,如图10所示.而对于第二种形式的工作流,其任务之间的数据依赖较简单,在业务约束条件下,本文的分割优化方法对于工作流执行时间的优化效果与HEFT,MINMIN算法较接近,如图11所示.

此外,对节点平均负载也做了较明显的优化.与HEFT和MINMIN算法相比,本文优化方法使节点平均负载更接近1,从而更好地保障了节点稳定性,如图12和图13所示.

表1 五个数据中心的配置

Table 1 Configuration of five data centers

数据中心	主机数量	宽带	说明
Datacenter_0	1	1.5e7	Edge
Datacenter_1	1	1.5e7	Edge
Datacenter_2	1	1.5e7	Edge
Datacenter_3	1	1.5e7	Edge
Datacenter_4	3	0.75e7	Cloud

表2 边缘节点及云计算中心的参数设置

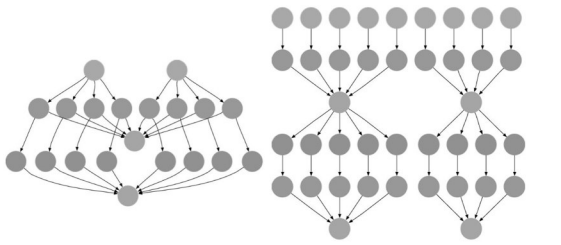
Table 2 Parameter configuration of edge nodes and cloud computing centers

参数\节点	边缘节点	云中心
Mips	3000	3000
Ram	1800	2048
Storage	10000	10000
Bw	10000	10000

表3 虚拟机的参数配置

Table 3 Parameter configuration of virtual machines

参数\虚拟	1	2	3
Size	1100	800	1100
Mips	512	512	512
Ram	1000	800	1200
Bw	1000	1200	800



(a) 工作流形式(1) (b) 工作流形式(2)

图9 仿真实验包含的两种工作流形式

Fig.9 Two workflow forms included in the simulation experiments

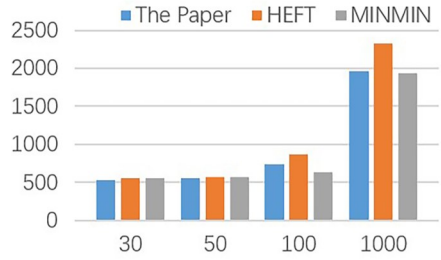


图10　本文算法和HEFT,MINMIN算法对工作流形式(1)的执行时间对比

Fig. 10　Execution time of our algorithm with HEFT and MINMIN algorithm for workflow form (1)

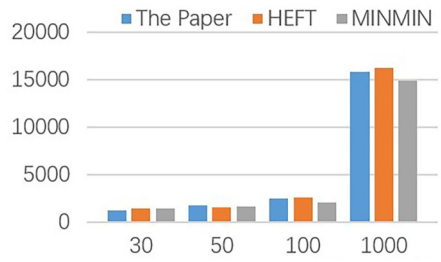


图11　本文算法和HEFT,MINMIN算法对工作流形式(2)的执行时间对比

Fig. 11　Execution time of our algorithm with HEFT and MINMIN algorithm for workflow form (2)

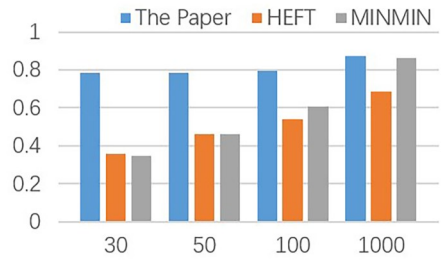


图12　本文算法和HEFT,MINMIN算法对工作流形式(1)的负载对比

Fig. 12　Load balancing of our algorithm with HEFT and MINMIN algorithm for workflow form (1)

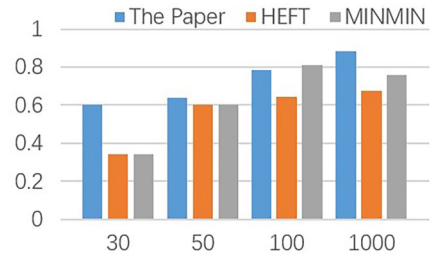


图13　本文算法和HEFT,MINMIN算法对工作流形式(2)的负载对比

Fig. 13　Load balancing of our algorithm with HEFT and MINMIN algorithm for workflow form (2)

6　结论

在云边协同环境下,IoT数据工作流数据量大、数据源分散,所以IoT数据工作流各任务之间的数据依赖较复杂,且调度时数据传输不可避免.本文通过在IoT数据工作流的多目标优化调度之前加入分割优化方法来优化数据传输,通过实验评估,该方法可以有效优化IoT数据工作流在业务约束条件下调度时的执行时间,有效提高了服务质量,并且使各个节点的平均负载更加均衡,使得系统工作更加稳定.未来将针对具体的IoT工作流做进一步的分割和调度优化.

参考文献

[1] 潘俊虹.基于工作流和QoS的物联网服务组合技术研究.武夷学院学报,2016,35(3):59－62.(Pan J H. A reserch on IoT web service composition technology based on workflow and QoS. Journal of Wuyi University,2016,35(3):59－62.)

[2] 田倬璟,黄震春,张益农.云计算环境任务调度方法研究综述.计算机工程与应用,2021,57(2):1－11.(Tian Z J,Huang Z C,Zhang Y N. Review of task scheduling methods in cloud computing environment. Computer Engineering and Applications,2021,57(2):1－11.)

[3] 陆艺雯.基于智能电网的数据中心能源优化与任务调度方法研究.硕士学位论文.南京:南京航空航天大学,2019.(Lu Y W. Research on data center energy optimization and task scheduling methods based on smart grid. Master Dissertation. Nanjing:Nanjing University of Aeronautics and Astronautics,2019.)

[4] 张龙信,王兰,肖满生,等.异构云系统中预算成本约束下高效的工作流调度算法.小型微型计算机系统,2020,41(6):1182－1187.(Zhang L X,Wang L,Xiao M S,et al. Efficient work flow scheduling algorithm under cost budget constraint in heterogeneous cloud systems. Journal of Chinese Computer Systems,2020,41(6):1182－1187.)

[5] Paknejad P,Khorsand R,Ramezanpour M. Chaotic improved PICEA－g－based multi－objective optimization for workflow scheduling in cloud environment. Future Generation Computer Systems,2021(117):12－28.

[6] Meena J, Vardhan M. Cost‐effective heuristic workflow scheduling algorithm in cloud under deadline constraint. Recent Advances in Computer Science and Communications, 2020, 13(6): 1302—1317.

[7] Qian Y F, Jiang Y Y, Hossain M S, et al. Privacy-preserving based task allocation with mobile edge clouds. Information Sciences, 2020, 507: 288—297.

[8] Chen P, Xia Y N, Yu C. A novel reinforcement‐learning‐based approach to workflow scheduling upon infrastructure-as-a-service clouds. International Journal of Web Services Research, 2021, 18(1): 21—33.

[9] Dong T T, Xue F, Xiao C B, et al. Workflow scheduling based on deep reinforcement learning in the cloud environment. Journal of Ambient Intelligence and Humanized Computing, 2021, 12(12): 10823—10835.

[10] 陈俊宇,刘茜萍. 云环境下基于阶段划分的数据密集型工作流调度. 南京邮电大学学报(自然科学版), 2020, 40(4): 103—110. (Chen J Y, Liu X P. Data-intensive workflow scheduling based on phase division in cloud environment. Journal of Nanjing University of Posts and Telecommunications (Natural Science Edition), 2020, 40(4): 103—110.)

[11] 李万清,刘辉,李忠金,等. 移动边缘计算环境下面向安全和能耗感知的服务工作流调度方法. 计算机集成制造系统, 2020, 26(7): 1831—1842. (Li W Q, Liu H, Li Z J, et al. Security and energy aware scheduling for service workflow in mobile edge computing. Computer Integrated Manufacturing Systems, 2020, 26(7): 1831—1842.)

[12] 王柳婧,蒋一翔,徐元根. 基于多约束图分割机制的科学工作流调度. 计算机应用与软件, 2019, 36(10): 299—304. (Wang L J, Jiang Y X, Xu Y G. Scientific workflow scheduling algorithm based on multiple constraints graph division mechanism. Computer Applications and Software, 2019, 36(10): 299—304.)

[13] 刘晓霞,李芳. 云环境中期限分割下工作流调度代价优化仿真. 实验室研究与探索, 2018, 37(10): 136—141, 161. (Liu X X, Li F. Simulation on workflow scheduling cost optimization under deadline division in clouds. Research and Exploration in Laboratory, 2018, 37(10): 136—141, 161.)

[14] 薛凡. DAG分割模型下的云工作流调度策略. 计算机应用研究, 2019, 36(12): 3725—3728, 3734. (Xue F. Cloud workflow scheduling strategy in DAG partition model. Application Research of Computers, 2019, 36(12): 3725—3728, 3734.)

[15] Pei S J, Zhang Q G, Cheng X H. Workflow scheduling using graph segmentation and reinforcement learning. International Journal of Performability Engineering, 2020, 16(8): 1262—1270.

[16] 薛庆水,李凤英. 基于云环境的双QoS约束多目标工作流调度. 计算机工程与设计, 2019, 40(8): 2196—2203. (Xue Q S, Li F Y. Multi‐objective workflow scheduling with Bi‐QoS constraint based on cloud environment. Computer Engineering and Design, 2019, 40(8): 2196—2203.)

[17] 袁友伟,黄锡恺,俞东进,等. 移动边缘计算环境下服务工作流容错调度算法. 计算机集成制造系统, 2021, 27(6): 1693—1702. (Yuan Y W, Huang X K, Yu D J, et al. Fault-tolerant scheduling algorithm for service workflow in MEC environment. Computer Integrated Manufacturing Systems, 2021, 27(6): 1693—1702.)

[18] 钟诗奇,龚晓峰. 基于改进粒子群算法的云工作流调度. 电子设计工程, 2019, 27(20): 110—114. (Zhong S Q, Gong X F. Cloud workflows scheduling based on improved particle swarm optimization algorithm. Electronic Design Engineering, 2019, 27(20): 110—114.)

[19] 薛凡,吴志健. 基于粒子群优化的云工作流任务调度. 微电子学与计算机, 2018, 35(8): 122—127, 131. (Xue F, Wu Z J. Cloud workflow wasks scheduling based on particle swarm optimization. Microelec-tronics & Computer, 2018, 35(8): 122—127, 131.)

[20] 韩莉. 实时系统工作流的能量感知容错算法. 博士学位论文. 上海: 华东师范大学, 2020. (Han L. Fault-tolerant and energy-aware algorithms for workflows and real‐time systems. Ph. D. Dissertation. Shanghai: East China Normal University, 2020.)

(责任编辑 杨可盛)