云环境下数据中台能耗感知的微服务任务调度算法

谢裕清[1] 王 渊[2,3] 江 樱[1] 李 浩[4] 王永利[4*]

[1](国网浙江省电力有限公司信息通信分公司 浙江 杭州 310000)

[2](南瑞集团有限公司(国网电力科学研究院有限公司) 江苏 南京 211100)

[3](江苏瑞中数据股份有限公司 江苏 南京 211100)

[4](南京理工大学计算机科学与工程学院 江苏 南京 210094)

摘 要 针对云计算环境下数据中台具有依赖关系的微服务任务调度导致能耗浪费问题,根据虚拟资源节点的能耗表现,构建相应的能耗效益函数,在此基础上,提出一个融入能耗优化的启发式微服务任务调度算法。算法主要由两个子算法构成:任务映射算法和串行任务合并算法。任务映射算法为每个工作流任务选择最优虚拟机类型,串行任务合并算法用于合并工作流中两个满足约束条件的子任务。仿真实验表明,该算法与同类算法相比,在同等条件下具有较好的能耗优化性能。

关键词 数据中台 云计算 能耗 微服务 启发式算法 任务调度

中图分类号 TP311 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2022.12.046

ENERGY AWARE TASKS SCHEDULING FOR MICROSERVICE IN DATA CENTER PLATFORM CLOUD ENVIRONMENT

Xie Yuqing[1] Wang Yuan[2,3] Jiang Ying[1] Li Hao[4] Wang Yongli[4*]

[1](*Information & Telecommunication Branch, State Grid Zhejiang Electric Power Corporation, Hangzhou 310000, Zhejiang, China*)

[2](*State Grid Electric Power Research Institute Co., Ltd., Nanjing 211100, Jiangsu, China*)

[3](*China Realtime Database Co., Ltd., Nanjing 211100, Jiangsu, China*)

[4](*School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, Jiangsu, China*)

Abstract In order to solve the dependent tasks scheduling problem which did not take higher energy consumption into consideration of data center platform under cloud computing environment, the energy effectiveness function was constructed according to the energy consumption performance of virtual resource nodes. A novel heuristic tasks scheduling algorithm for microservice which considered energy consumption was proposed based on the function. The algorithm consisted of two sub-algorithms: task mapping algorithm and sequence task merging algorithm. Task mapping algorithm was utilized to select optimal virtual machine type for each task and sequence task merging algorithm was used to merge two tasks which satisfies constraints in workflow. Simulation results show that this algorithm has a better performance in comparison with similar algorithms under the same conditions with respect to energy consumption.

Keywords Data center platform Cloud computing Energy consumption Microservice Heuristic algorithm Tasks scheduling

0 引 言

数据中台的理念源自阿里巴巴“构建规范定义的、全域可连接萃取的、智慧的数据处理平台”,数据中台的建设目标是实现前台数据的高效分析和应用[1]。数据中台的机制是通过云计算等数据技术,对海量数据进行采集、计算、存储、加工,并将数据的标准

收稿日期:2020-04-04。国家电网科技项目(5211XT190033)。**谢裕清**,博士,主研领域:信息技术。**王渊**,工程师。**江樱**,高工。**李浩**,讲师。**王永利**,教授。

和口径进行统一。数据中台将数据存储为经统一之后所形成的标准数据,构成大数据资产层,进而为客户提供高效服务。

随着数据中台建设与计算机网络云计算服务的发展,数据中台中越来越多的用户开始向云计算提交服务请求,利用数据中台提供的虚拟资源来完成任务,而云服务供应商需要根据云计算的特点、用户的偏好、实际的情况为用户动态地提供调度服务。在数据中台体系中,由于云平台任务调度面临的是一个非定常多项式(Non-deterministic Polynomial,NP)完全问题,引起了众多学者的关注,尤其是云平台下的多任务调度问题成为目前云计算领域研究的一个热点。

数据中台下的云数据中台用户越来越多,在扩大云平台规模的同时,为了提高调度的灵活性并且避免不必要的能耗浪费,云服务供应商提出了微服务架构[2]。微服务架构中的服务是指细粒度的、相互间松耦合的、可独立开发部署的组件。采用微服务架构设计的系统由一组通过轻量级接口实现的通信服务组成。

由于微服务架构将应用系统分为多个细粒度项目,原本单体式应用中的一系列操作可能需要多个服务协同完成,因此需要设计一个任务调度服务来处理关联多个服务的任务,比如在某一操作完成后执行另一类别的一个操作[3-4]。云计算数据中台具有异构性、动态性和地理分布性等特点,微服务工作流任务在云平台执行时,不但要保证任务执行时间满足截止期限制,云服务提供商还要最大限度优化数据中心能耗,否则可能违反 SLA(Service Level Agreement)协议。因此,如何设计启发式调度算法并应用到调度系统中来优化任务调度问题中的目标约束成为众多学者研究的核心问题。

文献[5]提出一种基于二次分派问题 QAP 的调度算法,在任务-资源映射阶段和资源频率分配阶段中同步进行能量优化;文献[6]提出了一个多目标遗传进化算法用于云平台中的任务调度,认为优化问题中的多个目标函数都是相互冲突的,目的是找到这些多目标折中的非支配 Pareto 解集。文献[7]针对云环境下数据密集型任务的调度问题,设计了一个基于二叉树建模的调度方法,同时满足用户 QoS(Quality of Service)需求[8]。

现有云计算环境下的工作流任务调度的研究大多针对独立任务或元任务的特殊形式,忽视了任务间的数据关联与优先约束关系,不能反映应用任务间的实际特征。同时缺少调度大规模复杂的微服务工作流的工作,无法充分利用微服务工作流提高应用的灵活性和敏捷性。

鉴于此,本文提出一种利用 Docker 容器融入能耗感知的微服务工作流任务调度算法,综合考虑了微服务任务间的数据依赖和优先级依赖,引入了能耗效益函数。在此基础上,设计两个子算法——任务映射算法和任务合并算法,用于协助云资源调度器调度管理提交到云数据中台的微服务工作流任务,最小化数据中台能耗的同时,满足工作流任务截止期限制,以实现用户 QoS 需求。仿真实验表明,在同等条件下,与经典的 EES(Enhanced Energy-Efficient Scheduling)[9] 和 EHEFT(Enhancing Heterogeneous Earliest Finish Time)[10] 等算法相比,在任务截止期长度变化以及子任务个数变化等环境下,本文算法具有较好的综合性能。

本文的贡献如下:

(1) 提出使用 Docker 容器调度云数据中台的微服务工作流,相比于直接使用 IaaS 虚拟机,可有效提高计算资源的利用率。

(2) 综合考虑微服务任务间的数据依赖和优先级依赖,体现微服务工作流执行的灵活性。

(3) 引入了能耗效益函数,满足不同的工作流截止期约束条件下,以最小化计算资源租赁成本为目标,有效提高云环境下数据中台调度大规模微服务工作流的经济效益。

1 问题描述及建模

数据中台上基于微服务的应用程序调度可以通过三个组件的属性来表征,即应用程序模型、系统模型和性能模型。应用程序模型定义基于业务工作流调度的应用程序结构;系统模型描述执行应用程序的 VM 和基础网络;性能模型负责调度优化等任务。这种由三部分组成的视图与定义经典调度问题的表示法一致。

应用模型中基于微服务的工作流模型采用有向无环图(Directed Acyclic Graph,DAG) $W=(T,E)$,其中顶点 $T=\{t_1,t_2,\cdots,t_n\}$ 代表 n 个任务,一对相邻任务 t_i 和 t_j 之间的执行依赖性由它们之间的有向边表示,可以附加权重 $w_{i,j}$ 表示从 t_i 传递到 t_j 的数据大小。任务 t_i 从其每个先前功能接收数据输入并执行预定义的业务逻辑。成功执行后,任务 t_i 的输出数据将立即发送到其后续任务。如图 1 所示,t_i 从 t_j 和 t_k 接收输入数据执行其自身的功能,并在完成执行后将输出数据 $w_{i,p}$ 发送至 t_p。此处不考虑应用程序模型中功能的计算开销。系统模型 IaaS 支持运行基于微服务的应用程序的基础环境,其中在互连的物理计算机上保留、部署和执行虚拟机。本文将这样的云环境建模为完全连

接的有向图 G,顶点由 VM 集合组成,边由虚拟机之间的一组有向链接组成,VM 虚拟机 v_i 和 v_j 之间的每个通信链路 $l_{i,j}$ 与带宽(BW) $b_{i,j}$ 和最小链路延迟(MLD) $d_{i,j}$ 相关联。应用程序的每个功能都由微服务实现,每个微服务都需要部署在一个容器中,该容器的计算能力定义为四元式 $p_{ci}=(c,f,r,d)$,其中:c 表示 CPU 内核数;f 表示每一 CPU 核的频率;r 表示 RAM 总大小;d 表示总磁盘空间。功能 t_i 被映射到微服务 ms_i 并由微服务 ms_i 实现,然后微服务 ms_i 被封装在容器 c_i 中,而计算能力 p_{ci} 部署在 VM v_y 上。

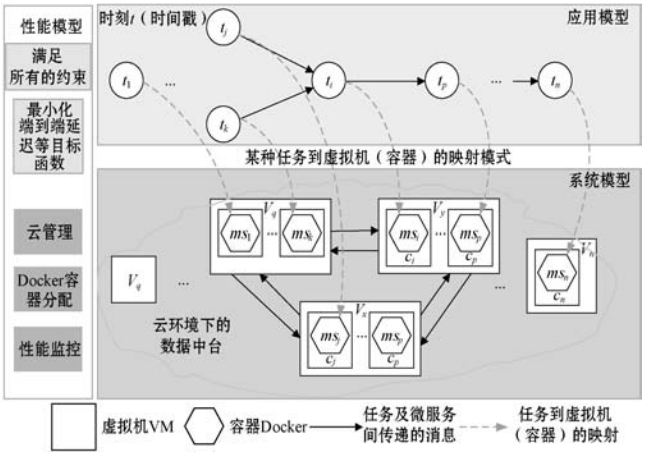

图 1　云数据中台上基于微服务的应用程序调度模型

在 IaaS 中,云提供商以互连 VM 的形式租用计算资源。通常,提供具有不同计算能力的不同类型的 VM,根据计算能力 c、f、r、d 满足不同的应用程序需求。不同的 VM 类型以不同的价格收费。以 Amazon EC2.3 为例,提供五种按需实例类型的配置和价格,Amazon 采用分步计费模式,其中 VM 不足 1 小时使用时间会被四舍五入到 1 小时。本文的工作针对单一公共云中的计算环境,在该公共云中物理机通过高带宽 Intranet 互连,因此不专门考虑数据传输成本。更笼统地说,考虑一个具有 l 个 VM 类型的公共云环境 $vm_1,vm_2,\cdots,vm_l$,其中每种类型的 vm_k 都与成本和性能相关的属性相关联,并定义为算力、最大功率、带宽三元组。图 1 给出了在示例系统模型中由 v_y 表示的一个 vm_k 类型的 VM 实例和由 v_q 和 v_x 表示的两个 vm_j 类型的 VM 实例。每个通信链路(例如,从 v_x 到 v_y)是有方向的,并与带宽($b_{x,y}$)和最小链路延迟($d_{x,y}$)相关联。

性能模型主要负责在公共数据中台云中执行基于微服务的应用程序调度,满足用户指定的预算约束、功耗约束下实现最小的端到端延迟(MED)目标。本文扩展了主要考虑能耗因素,将它们扩展到具有函数间依赖性的应用程序模型中,以识别和量化公共云的功耗成分。使用微服务和容器的一个关键优势是快速复制和迁移的便利性,因此将复制的微服务作为不同的实例运行,每个微服务都可以实现不同应用程序所需的相同功能。

与基于虚拟机的云服务相比,本文设计的基于容器的数据中台微服务模型的技术特征及优势为:

1)微服务易于被开发人员理解、修改和维护。各个微服务可被独立部署,各个微服务之间是松耦合的,每个微服务都很小,是有功能意义的服务,无论是在开发阶段或部署阶段都是独立的,有助于聚焦指定的业务功能或业务需求。2)基于容器的微服务运行占用的空间少启动速度快且便于迁移。与虚拟机相比,容器中的微服务可以共享操作系统内核,不需要在主机上构建多个虚拟机拥有完整的副本,占用的空间更少,启动速度也更快。每个微服务都可以实现不同应用程序所需的相同功能,可以将复制的微服务作为不同的实例运行,有利于快速复制和迁移。

为简便起见,假设每个虚拟机只对应一个容器。

定义 1　微服务工作流模型。通常情况下,一个微服务工作流 $W=(T,E)$ 可以用一个有向无环图(DAG)来表示,如图 2 所示。其中 $T=\{t_1,t_2,\cdots,t_n\}$ 为工作流中的任务集合,DAG 中一个顶点代表一个任务,E 是 DAG 中边的集合,代表任务之间的依赖关系。若任务 t_i 和任务 $t_j(t_i,t_j\in T,t_i\neq t_j$,且 $1\leqslant i<j\leqslant n)$ 之间存在依赖 $(t_i,t_j)\in E$,则意味着任务 t_i 必须在任务 t_j 之前完成,任务 t_i 称为父任务,任务 t_j 称为子任务。在一个 DAG 中,只有一个任务的父任务全部完成之后,该任务才能开始执行。任务 t_i 的子任务集合可以用式(1)表示。

$$(t_i)=\{t_j\mid(t_I,t_j)\in E\}\tag{1}$$

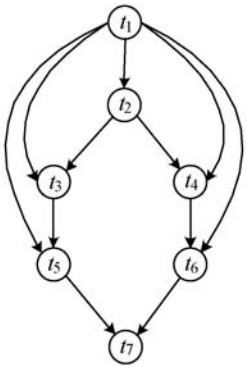

图 2　一个 DAG 工作流模型

定义 2　数据中台虚拟机模型。数据中台提供 l 种类型的虚拟机的类型集为 $VM=\{vm_1,vm_2,\cdots,vm_k\cdots,vm_l\}$,每种类型 VM 的虚拟机的特性都可以用($c$,$p^{\max}$,$B$)三元组表示,其中:$c$ 是虚拟机的计算能力,用每秒运算的浮点数(MFLOPS)衡量;$p^{\max}$ 是虚拟机的最大功率;B 为带宽。不同类型的虚拟机之间满足如下约束关系:

$$\frac{p_k^{\max}}{c_k} < \frac{p_{k+1}^{\max}}{c_{k+1}} \tag{2}$$

$$\text{s.t. } c_1 < c_2 < \cdots < c_k \wedge p_1^{\max} < p_2^{\max} < \cdots < p_k^{\max}$$

式中:所有虚拟机实例的计算能力和最大功率均按升序排序。式(2)就意味着,虚拟机之间随着计算能力的增强,虚拟机的功耗将随之增加,且增加幅度超过虚拟机计算能力的增强幅度。

定义 3 基于容器的数据中台虚拟机微服务任务调度问题可描述为:在数据中台中,将微服务工作流的 n 个子任务分配给 m 个异构的虚拟资源节点执行($m<n$)。其中,任务集为 $T=\{t_1,t_2,\cdots,t_n\}$,$t_i(i=1,2,\cdots,n)$表示第 i 个子任务,虚拟资源节点集为 $V=\{v_1,v_2,\cdots,v_m\}$,$v_j(j=1,2,\cdots,m)$表示第 j 个虚拟资源节点。每个子任务只能在一个虚拟资源节点上执行。任务集 $T=\{t_1,t_2,\cdots,t_n\}$ 与虚拟资源节点集 $V=\{v_1,v_2,\cdots,v_m\}$ 的分配关系可用矩阵 $\boldsymbol{X}$ 表示为:

$$\boldsymbol{X}=\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{m2} & \cdots & x_{nm} \end{pmatrix} \tag{3}$$

$$\text{s.t. } x_{ij} \in \{0,1\} \wedge \sum_{i=1}^{n} x_{ij} = 1$$

$$i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,m\} \tag{4}$$

定义 4 虚拟机动态能耗。一台虚拟机 vm_k 的动态能耗可以由它的动态执行功耗 P_j^k 和它执行任务时间的乘积得到:

$$E_j^k = \lambda_k \cdot (v_j^k)^2 \cdot f_j^k \cdot \eta \cdot t_{\text{execution}} \tag{5}$$

虚拟机 vm_k 执行任务时,其动态执行功耗 P_j^k 可由下式获得:

$$P_j^k = \lambda_k \cdot (v_j^k)^2 \cdot f_j^k \cdot \eta$$

$$\text{s.t. } \eta = \begin{cases} 1 & \text{被分配到该虚拟机上执行} \\ 0 & \text{其他} \end{cases} \tag{6}$$

式中:λ_k 是和虚拟机 vm_k 有关的常量;v_j^k 是 vm_k 在服务级别为 j 时的供应电压;f_j^k 是 vm_k 在服务级别为 j 时的虚拟机频率;v_j^k 和 f_j^k 都可以测得[11]。

定义 5 虚拟机空闲能耗。当虚拟机处在空闲状态时,CPU 芯片会使用动态电压频率调整(Dynamic Voltage Frequency Scaling,DVFS)技术把 CPU 的电压和运行频率调整到一个较低的水平。因此,虚拟机处在空闲状态时的能耗可由式(7)计算。

$$E_i^k = \lambda_k \cdot (v_{\min}^k)^2 \cdot f_{\min}^k \cdot t_{\text{idle}} \tag{7}$$

式中:$v_{\min}^k$ 和 $f_{\min}^k$ 分别是虚拟机空闲状态时的供应电压和频率;t_{idle} 是虚拟机处于空闲状态的时间间隔。

定义 6 数据中台的总能耗。由数据中台中所有虚拟机实例的总能耗组成(动态能耗和空闲能耗),因此,云数据中台的总功耗 E_{total} 为:

$$E_{\text{total}} = \sum_k \sum_j E_j^k + \sum_k \sum_i E_i^k \tag{8}$$

定义 7 基于以上定义,本文的调度问题可以描述为:找到一个微服务工作流调度方案,使得数据中台执行工作流任务时产生的总能耗最小,且满足工作流任务的截止期限制。可以由式(9)定义。

$$\begin{aligned} &\min \quad E_{\text{total}} \\ &\text{s.t. } makespan \leqslant deadline(w) \end{aligned} \tag{9}$$

2 算法设计与实现

任务的调度问题面临的是一个 NP-难问题[12],本文的目标是设计启发式调度算法以最大限度优化数据中台产生的能耗。下面给出算法的实现以及伪代码。

2.1 任务映射算法

定义 8 一个任务的完工时间由两部分组成:该任务真实的执行时间和该任务的依赖数据需要传输的时间。可由式(10)计算得出。

$$T(t_i^k) = \frac{w_i}{c_k} + \frac{D_i}{B} \tag{10}$$

式中:$T(t_i^k)$ 为任务 t_i 在类型为 k 上的虚拟机的完工时间;w_i 为 t_i 的工作负载;D_i 为 t_i 依赖的数据量。

定义 9 能耗效益函数。能耗效益函数代表消耗单位的能耗下能完成的工作负载的大小。任务 t_i 在类型为 k 的虚拟机上执行时的能耗效益函数可以通过式(11)计算。

$$u(t_i^k) = \frac{w_i}{T(t_i^k) \cdot p_k} \tag{11}$$

式中:分子为任务 t_i 的工作负载;分母为执行任务 t_i 时产生的能耗。能耗效益函数值越大,表明该任务被映射到类型为 k 的虚拟机上执行时越节能。

微服务工作流任务与虚拟机分配映射算法伪代码:

算法 1 微服务工作流任务与虚拟机分配映射算法 TaskMapping

输入:微服务工作流任务 $t_i, i \in 1,2,\cdots,n$,虚拟机能耗效益函数集合。

输出:任务 t_i 与虚拟机映射关系 map。

1) 对于任务 t_i 初始化所有虚拟机类型的能耗效益函数集合 $u(t_i^k) = \frac{w_i}{T(t_i^k) \cdot p_k}$;

2) 将能耗效益函数值按降序排序,得到 $U(t_i) = \{u(t_i^1), u(t_i^2), \cdots, u(t_i^k), \cdots, u(t_i^l)\}$;

3) 初始化任务集 $R=\varnothing, k=1$;

4) **while** $k \leqslant l$ and $R \neq \varnothing$ **do**

5) 针对任务 t_i,选取能耗效益函数集合中的第 k 个虚拟机类型作为目标虚拟机类型,如果 t_i 能在其截止期限执行完成,则选取当前第 k 种虚拟机类型机为 VM_{opt},把任务 t_i 映射到该类型的虚拟机的容器上,即 $map: t_i \rightarrow VM_{opt}$;

6) 否则,把任务 t_i 放入 R 中,针对 R 中的每一个任务,选取该任务的能耗效益函数集合的下一个虚拟机类型作为目标虚拟机类型;

7) **end do**

虚拟机类型的能耗效益排序采用快速排序或者堆排序,最优时间复杂度为 $O(l \log l)$,其中 l 为虚拟机类型数目,步骤 5)到步骤 7)算法复杂度为 $O(nl)$,n 为微服务工作流任务个数。与已有任务虚拟机映射算法相比,本文提出的启发式微服务工作流任务与虚拟机分配映射算法具有复杂度低且满足综合能耗效益函数约束。

2.2 串行任务合并算法

定义 10 两个存在依赖关系的串行任务之间数据传输时间可由式(12)获得。

$$Time_{data}(t_i, t_j) = \begin{cases} \dfrac{data(t_i, t_j)}{\min(B_p, B_q)} & p \neq q \\ 0 & p = q \end{cases} \tag{12}$$

式中:$data(t_i, t_j)$ 表示有依赖关系的任务 t_i 和 t_j 之间传输的数据量。

从式(12)可以看出,如果两个串行任务分配到一台虚拟机上执行,那么这两个任务间的依赖数据传输时间就会被节约。因此,下面给出串行任务合并算法的伪代码。

算法 2 串行任务合并算法 TaskMerging

输入:微服务工作流任务 $t_i, i \in 1,2,\cdots,n$,虚拟机能耗效益函数集合。

输出:微服务工作流任务集。

1) 初始化空任务集合,记为 T。

2) 对工作流中的任务进行拓扑排序,然后把所有拓扑排序后的任务放入 T 中,$i=1$;

3) **while** 任务集合 T 不为空 **do**

4) **if** (t_i 只有一个直接后继任务且该后继任务只有一个前驱任务,或者 t_i 只有一个前驱任务且该前驱任务只有一个后继任务) then

5) 找到一个虚拟机类型 k,满足在($deadline(t_i) + deadline(t_{i+1})$)时间内能完成 t_i 和 t_{i+1};

6) **if** $e(t_{i,i+1}^{k}) \leqslant e(t_i^j) + e(t_{i+1}^{j+1})$ then

7) 合并 t_i 和 t_{i+1} 为一个新任务 t',

8) 更新 t' 的前驱任务、后继任务、工作负载和截止期限;

9) 将 t_i 从 T 中移除;

10) **else**

11) 不合并 t_i 和 t_{i+1},移除 t_i;

12) **end if**

13) **end if**

14) **end do**

图 3 给出串行任务合并过程示意图,算法复杂度为 $O(nl)$,其中 n 为微服务工作流任务个数。

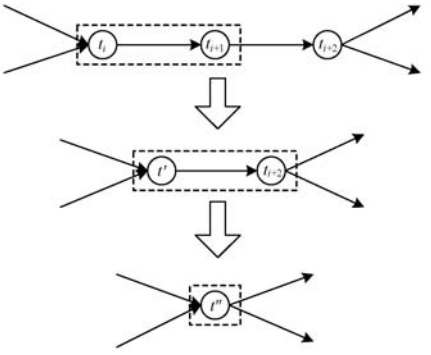

图 3　串行任务合并过程示意图

与已有的串行任务合并算法相比,本文提出的串行任务合并算法考虑到虚拟机能耗效益因素,并且关注存在依赖关系的串行任务合并,显著地节约了数据传输时间。

2.3 能耗感知的任务调度算法

本文提出的能耗感知的微服务工作流任务调度算法旨在优化云环境下数据中台执行微服务任务时产生的能耗,同时满足任务截止期限制。算法主要由以上提出的两个子算法构成。

算法 3 能耗感知的微服务工作流任务调度算法

输入:微服务工作流任务集合 $T=\{t_1, t_2, \cdots, t_n\}$,虚拟资源节点集 $V=\{v_1, v_2, \cdots, v_m\}$。

输出:任务与资源分配关系矩阵 $\boldsymbol{X}$。

1) **while** 未达到调度评价指标 **do**

2) 调用任务映射算法 TaskMapping;

3) 调用串行任务合并算法 TaskMerging;

4) **end do**

算法复杂度为 $O(nlm)$。

3 仿真实验及结果分析

为了验证本文算法在节约能耗方面的综合性能,在不同的仿真参数和性能指标下将其与同样为启发式任务调度算法的 EES 和 EHEFT 进行了综合对比分析。实验环境为 Windows 7 系统,Intel(R) Core 3.20 GHz CPU,1 TB 硬盘,8 GB RAM。仿真平台为澳大利亚墨尔本大学的网络实验基于 Java 语言开发的 Cloudsim 云仿真平台[13]。在 Cloudsim 仿真平台中,DataCenter 类代表数据中台,VirtualMachine 类代表虚拟资源节点,Cloudlet 类代表任务。

3.1 两个子算法之间节能性能比较

本实验旨在比较两个子算法随任务截止期变化时的节能性能。实验参数为:虚拟资源节点个数 $m=10$,任务数 $n=30$,虚拟资源节点类型数 $k=5$。实验结果如图 4 所示。

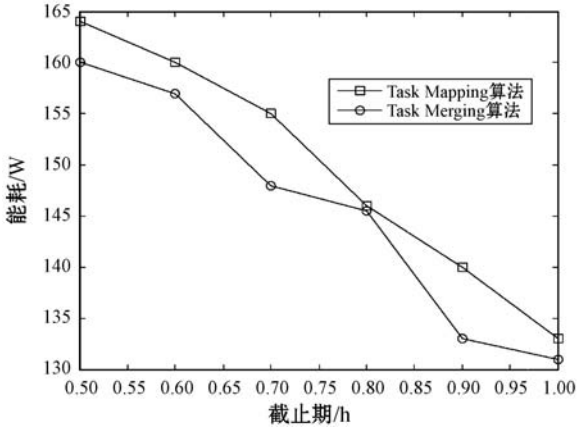

图 4 任务截止期变化对两个子算法能耗的影响

可以看出,随着任务截止期的变长,两者的能耗都在降低,且固定任务截止期时,任务合并算法的节能效果优于任务映射算法的节能效果,其原因在于,串行任务合并算法避免了数据依赖任务之间数据集的传输,而且任务合并后只需要租用更少的虚拟机实例。本文提出的能耗感知的任务调度算法在未达到调度评价指标情况下,综合两个子算法的节能优势,实现综合能耗最优化。

3.2 随任务截止期变化的性能比较

本实验的实验参数与实验 1 相同,主要考察随任务截止期变化时本文算法与 EES 和 EHEFT 之间的节能性能比较,实验结果如图 5 所示。

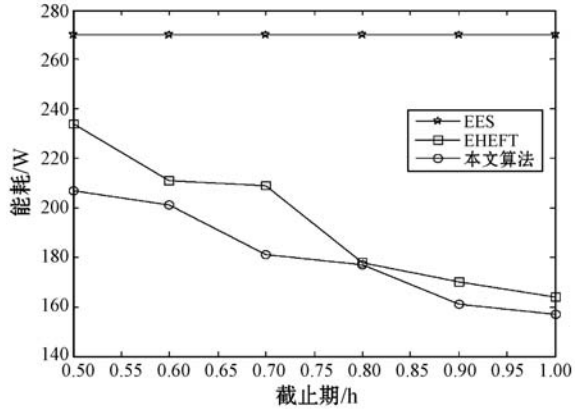

图 5 任务截止期变化对能耗的影响

可以看出,本文算法的节能性能最优,这说明本文算法在调度任务执行时对数据中台的能耗最敏感,因为本文算法的能耗效益函数总是在满足任务截止期的前提下,借助工作流任务与虚拟机分配映射算法和串行任务合并算法,选择能耗最优的虚拟机执行任务。EHEFT 次之,EES 的能耗保持不变且维持在一个较高水平,这是因为 EES 总是选择数据中台中性能最强的虚拟机执行任务,不考虑能耗因素。

3.3 随任务数变化的性能比较

实验参数为:虚拟资源节点个数 $m=30$,虚拟资源节点类型数 $k=5$。性能指标比较结果如图 6 所示。可以看出,随着任务数的扩充,本文算法、EES 和 EHEFT 的能耗都有所增加,但是固定任务数时,EES 和 EHEFT 产生的能耗均高于本文算法,说明在同等条件下,本文算法的能耗效益函数机制在执行任务时节约能耗性能和可扩展性均明显优于 EES 和 EHEFT。

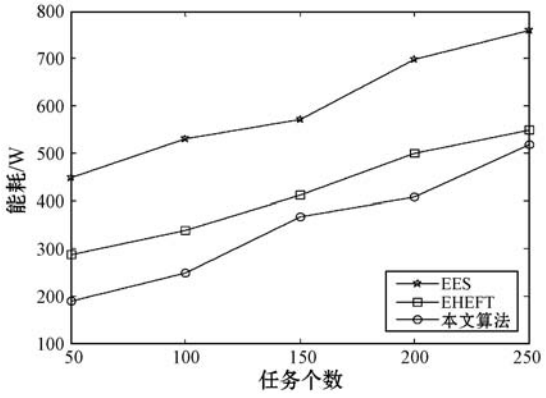

图 6 任务个数变化对能耗的影响

3.4 随虚拟资源节点异构性变化的性能比较

实验参数为:任务个数 $n=100$,虚拟资源节点个数 $m=30$。能耗比较变化趋势如图 7 所示。可以看出,随着虚拟资源节点异构性(虚拟机类型数)的变化,三个算法的能耗变化不明显,跟实验 2 类似,EES 能耗最高,EHEFT 次之,本文算法能耗最少。这说明本文算法对虚拟资源的异构性变化不敏感,但是节能性能最优。

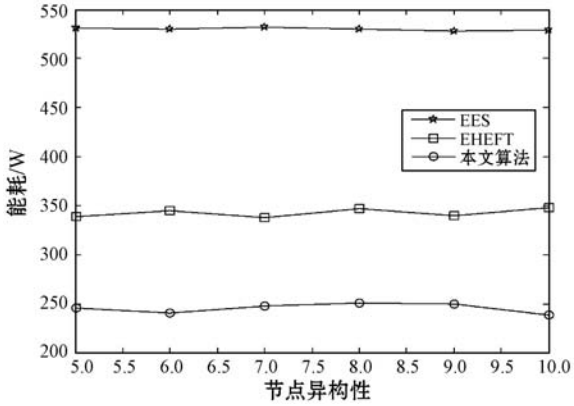

图 7 虚拟资源节点异构性变化对能耗的影响

综上,本文提出的能耗感知的微服务工作流任务调度算法与已有算法相比,在随任务截止期变化、随任务数变化、随虚拟资源节点异构性变化等几个方面均具有更好的性能。

4 结　语

能耗问题在基于微服务的数据中台云数据中心表现得越来越重要。本文针对云数据中台中依赖任务调度导致的能耗浪费问题,构建了能耗效益函数模型和云数据中台能耗模型,并提出一个包含了两个子算法的启发式微服务工作流任务调度算法对模型进行求解。仿真实验表明,本文算法在能耗管理方面具有较好的综合性能。下一步的工作将是考虑虚拟资源节点的综合成本问题,同时满足基于用户需求和云数据中台节能的多目标优化调度算法,以及采用相应的数学模型来刻画不同的用户需求。

参 考 文 献

[1] 李炳森,胡全贵,陈小峰,等. 电网企业数据中台的研究与设计[J]. 电力信息与通信技术,2019,17(7):29 - 34.

[2] Bao L, Wu C, Bu X, et al. Performance modeling and workflow scheduling of microservice-based applications in clouds [J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(9):2114 - 2129.

[3] 万书鹏,易强,张凯,等. 基于微服务架构的新一代调控系统服务编排技术[J]. 电力系统自动化,2019,43(22):116 - 121.

[4] 唐溢. 基于微服务架构的任务调度系统的设计与实现[D]. 成都:西南交通大学,2019.

[5] 孙赫勇,文勃,郑丹,等. 移动云计算任务交互图的能效映射与调度[J]. 计算机应用与软件,2020,37(1):8 - 14,52.

[6] Dastjerdi A V, Buyya R. An autonomous reliability-aware negotiation strategy for cloud computing environments[C]// 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGRID),2012:284 - 291.

[7] Zhu Z, Zhang G, Li M, et al. Evolutionary multi-objective workflow scheduling in cloud[J]. IEEE Transactions on Parallel and Distributed Systems,2016,27(5):1344 - 1357.

[8] Zhao Q, Xiong C, Yu C, et al. A new energy-aware task scheduling method for data-intensive applications in the cloud[J]. Journal of Network and Computer Applications, 2015, 59:14 - 27.

[9] Li B, Song A M, Song J. A distributed QoS-constraint task scheduling scheme in cloud computing environment: Model and algorithm[J]. Advances in Information Sciences and Service Sciences,2012,4(5):283 - 291.

[10] Huang Q, Su S, Li J, et al. Enhanced energy-efficient scheduling for parallel applications in cloud [C]//12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGRID),2012:781 - 786.

[11] Thanavanich T, Uthayopas P. Efficient energy aware task scheduling for parallel workflow tasks on hybrids cloud environment[C]//2013 International Computer Science and Engineering Conference(ICSEC),2013:37 - 42.

[12] 林伟伟,吴文泰. 面向云计算环境的能耗测量和管理方法[J]. 软件学报,2016,27(4):1026 - 1041.

[13] Bajaj R, Agrawal D P. Improving scheduling of tasks in a heterogeneous environment[J]. IEEE Transactions on Parallel and Distributed Systems,2004,15(2):107 - 118.

(上接第 271 页)

[2] Li W, Sun S, Jia Y, et al. Robust unscented Kalman filter with adaptation of process and measurement noise covariances[J]. Digital Signal Processing,2016,48:93 - 103.

[3] Arasaratnam I, Haykin S. Cubature Kalman filters[J]. IEEE Transactions on Automatic Control,2009,54(6):1254 - 1269.

[4] Gordon N J, Salmond D J, Smith A F M. Novel approach to nonlinear/non-Gaussian Bayesian state estimation[J]. IEEE Proceedings F (Radar and Signal Processing), 1993, 140(2):107 - 113.

[5] 张勇刚,王刚,黄玉龙,等. 递推更新高斯粒子滤波器[J]. 控制理论与应用,2016,33(3):353 - 360.

[6] 李良群,姬红兵,罗军辉. 迭代扩展卡尔曼粒子滤波器[J]. 西安电子科技大学学报(自然科学版),2007,34(2):233 - 238.

[7] 杨丽华,葛磊,李保林,等. 强跟踪 UKF 粒子滤波算法[J]. 计算工程与设计,2015,36(9):2432 - 2436.

[8] 赵凯丽,高火涛,曹婷,等. 基于 CPFDE 的目标跟踪算法研究[J]. 现代雷达,2019,41(1):36 - 41.

[9] 刘丹,段建民,于宏啸. 基于自适应渐消 EKF 的 FastSLAM 算法[J]. 系统工程与电子技术,2016,38(3):644 - 651.

[10] 包子阳,余继周智能优化算法及其 Matlab 实例[M]. 北京:电子工业出版社,2016.

[11] 贺姗,师昕. 带渐消因子的容积卡尔曼滤波算法[J]. 科技与创新,2017(13):1 - 2.

[12] 王倩,曾庆军,张家敏,等. 基于自适应渐消无迹粒子滤波的 Unscented FastSLAM 算法[J]. 计算机应用研究,2019,36(5):1315 - 1318.

[13] 黄友锐. 智能优化算法及其应用[M]. 北京:国防工业出版社,2008.

[14] Zuccolotto M, Pereira C E, Fasanotti L, et al. Designing an artificial immune systems for intelligent maintenance systems [J]. IFAC-Papersonline,2015,48(3):1451 - 1456.

[15] 张廷军,郭毅锋,黄丽敏. 改进重采样的移动机器 SLAM 算法[J]. 计算机工程与设计,2019,40(11):3276 - 3281.