



控制与决策

Control and Decision

ISSN 1001-0920, CN 21-1124/TP

《控制与决策》网络首发论文

题目: 边缘计算环境中基于规划图的服务组合修复方法
作者: 高志浩, 李静, 祝铭, 刘彤阳, 鲁瑞
DOI: 10.13195/j.kzyjc.2023.1599
收稿日期: 2023-11-16
网络首发日期: 2024-03-07
引用格式: 高志浩, 李静, 祝铭, 刘彤阳, 鲁瑞. 边缘计算环境中基于规划图的服务组合修复方法[J/OL]. 控制与决策. <https://doi.org/10.13195/j.kzyjc.2023.1599>



网络首发: 在编辑部工作流程中, 稿件从录用到出版要经历录用定稿、排版定稿、整期汇编定稿等阶段。录用定稿指内容已经确定, 且通过同行评议、主编终审同意刊用的稿件。排版定稿指录用定稿按照期刊特定版式(包括网络呈现版式)排版后的稿件, 可暂不确定出版年、卷、期和页码。整期汇编定稿指出版年、卷、期、页码均已确定的印刷或数字出版的整期汇编稿件。录用定稿网络首发稿件内容必须符合《出版管理条例》和《期刊出版管理规定》的有关规定; 学术研究成果具有创新性、科学性和先进性, 符合编辑部对刊文的录用要求, 不存在学术不端行为及其他侵权行为; 稿件内容应基本符合国家有关书刊编辑、出版的技术标准, 正确使用和统一规范语言文字、符号、数字、外文字母、法定计量单位及地图标注等。为确保录用定稿网络首发的严肃性, 录用定稿一经发布, 不得修改论文题目、作者、机构名称和学术内容, 只可基于编辑规范进行少量文字的修改。

出版确认: 纸质期刊编辑部通过与《中国学术期刊(光盘版)》电子杂志社有限公司签约, 在《中国学术期刊(网络版)》出版传播平台上创办与纸质期刊内容一致的网络版, 以单篇或整期出版形式, 在印刷出版之前刊发论文的录用定稿、排版定稿、整期汇编定稿。因为《中国学术期刊(网络版)》是国家新闻出版广电总局批准的网络连续型出版物(ISSN 2096-4188, CN 11-6037/Z), 所以签约期刊的网络版上网络首发论文视为正式出版。

边缘计算环境中基于规划图的服务组合修复方法

高志浩¹, 李静¹, 祝铭^{1†}, 刘彤阳¹, 鲁瑞²

(1. 山东理工大学 计算机科学与技术学院, 山东 淄博 255000; 2. 澳大利亚国立大学 工程与计算机科学学院, 堪培拉 2601)

摘要: 在云计算和边缘计算环境中, 通过提取和组合可用的服务, 以满足用户需求已成为常见的做法。然而, 当前的方法难以应对由于用户需求变化或外部环境变动导致的组合失效问题。为了应对这一挑战, 本文提出了一种边缘计算环境中基于规划图的服务组合及其修复方法。本文首先结合移动路径模型和规划图方法完成服务组合过程。通过规划图的构建, 本文可以有效地评估和选择适合用户需求的服务组合。当服务集合发生变化或用户目标更改时, 本文的方法能够在现有的规划图基础上生成新的解决方案, 以满足用户的需求。这种修复方法能够实时适应云边环境中的变化, 提高系统的灵活性和可靠性。经过实验测试发现本文提出的修复方法相较于重新规划具有更好的性能表现, 并证明了修复方法在组合失效问题上的有效性和实用性。

关键词: 服务组合; 规划图; 服务组合修复; 边缘计算; 服务质量; 服务时延

中图分类号: TP338.8

文献标志码: A

DOI: 10.13195/j.kzyjc.2023.1599

A Repairing Service Composition Method based on Planning Graph under Edge Computing

GAO Zhi-hao¹, LI Jing¹, ZHU Ming^{1†}, LIU Tong-yang¹, LU Rui²

(1. School of Computer Science and Technology, Shandong University of Science and Technology, Zibo 255000, China; 2. School of Engineering and Computer Science, Australian National University, Canberra 2601, Australia)

Abstract: In cloud and edge computing environments, it is common to extract and combine available services to meet user needs. However, current methods struggle to cope with composition failures caused by changes in user needs or external environment. To address this challenge, this paper proposes a planning-based service composition and repair method in edge computing environments. We first combine the mobile path model and planning graph method to complete the service composition process. The construction of the graph allows for efficient evaluation and selection of service compositions that suit user demands. When the service set changes or user goals are modified, our method can generate new solutions based on the existing planning graph to meet user needs. This repair method can adapt to changes in real-time in the cloud-edge environment, enhancing system flexibility and reliability. Experimental tests have shown that the proposed repair method outperforms replanning, demonstrating its effectiveness and practicality in addressing combination failures.

Keywords: service composition; planning graph; repair service composition; edge computing; Qos; service delay

0 引言

近年来, 随着科技的发展, 越来越多的人开始使用智能设备^[1]。与此同时, 许多对时延要求较高以及需求密集型应用的数量也在迅速增加。传统的云计算在面对数据爆炸和更高的服务质量要求方面已经不能完全满足用户的需求。此外, 云计算中心化架构也存在单点故障风险, 并且难以满足对敏感性和低延迟的需求^[2]。

为了应对这些挑战, 移动边缘计算技术 (Mobile Edge Computing, MEC) 被提出^[3,4], 其采用分布式架构, 将计算资源分散到多个边缘设备上, 并且可以将计算、存储和网络资源放置在离用户更近的位置, 使得用户可以在较短的响应时间内访问到各种所需的服务。其中用户移动过程会提交服务请求, 数据、算力、资源等都可被包成服务, 供用户按需请求。在云边环境中, 由于用户需求的复杂性, 单个服

收稿日期: 2023-11-16; 录用日期: 2024-01-02.

基金项目: 国家自然科学基金 (62002205), 嵌入式与网络计算湖南省重点实验室开放基金 (20220106), 教育部高等学校科学研究发展中心中国高校产学研创新基金 - 新一代信息技术创新项目 (2022IT048)

[†]通讯作者. E-mail: zhu_ming@sdu.edu.cn.

务往往无法满足要求, 因此需要通过整合和共享多个服务资源来进行服务组合。然而, 服务间关系复杂且用户需求各异, 这给服务组合和规划带来了很大的挑战。为应对这个问题, AI 规划方法^[5-8]应运而生。Li 等人^[9]提出了一种关系数据库方法, 用于自动化服务组合。该方法预先生成和存储所有可能的服务组合, 并在用户请求时通过查询数据库返回最优的服务。Zhu 等人^[10]提出了一种综合服务质量 (Quality of Service, QoS) 和修改规划图的 Web 服务组合方法, 以应对复杂性和耗时性。通过模糊逻辑处理不确定性并支持决策, 该方法在服务修剪和解生成方面进行优化, 取得了更好的解决方案。Han 等人^[11]提出一种改进的多目标灰狼优化 (improved multi-objective grey wolf optimization, IMOGWO) 算法来求解 Web 环境下的服务组合问题。然而, 这些方法只能提供一次性的解决方案, 无法适应服务或用户需求的变化。

为了应对动态变化的服务和用户需求的挑战, 研究者们提出了多种方法。其中, Friedrich 等人^[12]提出了一种自愈方法处理服务流程中的异常, 并使用基于模型的方法修复故障活动。Church 等人^[13]研究了服务组合中的故障问题, 并提出了一个形式化的模型和相应的修复建议, 设计了一种支持程序员调试服务故障的技术。Eder 等人^[14]提出了一种架构, 将活动和实例视为可管理的资源, 用于修复故障流程。然而, 这些方法无法适用于移动边缘环境。因此, 本文使用规划图方法描述云边环境下的服务组合问题, 并引入修复算法减少用户等待时间和延迟。

规划图是一种图形化工具, 能够很好地建模服务组合问题, 其能够有效表示服务之间的依赖关系和约束条件, 并描述服务的属性和行为。这种方法的有效性已在 Web 服务环境中得到验证^[15]。然而, 他们的方法有局限性, 该方法并没有进一步考虑服务 QoS 对整个维修过程的影响。本文将其改进并应用在云边环境中来描述边缘环境中的服务组合。当服务不可用、损坏或用户需求发生变化时, 规划图的一部分仍然有效, 因此本文通过修复原有规划图来解决这一问题, 而不是只提供一次性的解决方案, 避免了重新规划 (replanning)。规划图作为一种灵活且高效的工具, 能够应对云边服务环境中的不确定性和变化。所以, 使用规划图可以有效解决云边服务中的服务组合和规划问题并且应对不断变化的服务环境。

这种基于规划图的修复方法能够提供灵活且高效的服务组合解决方案, 适应云边服务环境中的动态变化和不确定性。本文的具体贡献总结如下:

(1) 本文考虑了在云和边缘计算中的移动服务组合问题, 并使用规划图来描述这一问题, 其可以描述服务之间的关系, 并且这种应用规划图的方法在云边环境中还没有被探讨过。

(2) 本文提出了一种基于规划图的服务组合修复方法。在移动边缘计算环境中, 服务可能会因网络变化、资源限制或其他因素而发生故障或中断。本文的修复方法能够动态地调整服务组合, 以适应变化的环境, 并尽快恢复系统的正常运行。

1 背景知识

1.1 规划图

规划图 PG 是一个有向无环的水平图^[16], 交替包含问题层 P_i 和动作层 A_i 。其中 A_i 层包含动作 a , P_i 层中包含 a 的前提条件 $preconds(a)$ 与 a 的后续结果 $effects(a)$ 。通过组合问题层和动作层, 可以形成规划图的结构。

首先进行前向扩展, 规划图逐渐从初始状态 S_0 向目标状态 $goal$ 拓展。在前向扩展的过程中, 问题层和动作层不断扩展, 建立问题状态和动作之间的联系。当规划图满足终止条件时, 即问题层中包含全部目标集合或最后两层问题层的命题层相同, 前向扩展过程停止。规划图构建完成后, 使用反向搜索来寻找满足目标的服务组合解决方案。反向搜索从目标状态开始, 逆向回溯到初始状态并跟踪每个状态的动作, 最终可以得到一个满足目标的组合方案。

1.2 服务定义

本文将服务 w 定义为一个 4 元组 (p, w_{in}, w_{out}, q) 其中 p 表示服务 w 所部署的服务器位置, 它可以在云服务器或者边缘服务器。 w_{in} 表示服务 w 所需要的输入参数。 w_{out} 为服务 w 的输出参数。 q 表示服务 w 的服务质量 (QoS)。本文中使用响应时间 (T) 作为衡量服务 QoS 的标准。

在这里假设一个人脸验证的服务, 其位于云服务器 (p), 响应时间为 $12ms(q)$, 用户需要提供面部视频信息 (w_{in}), 之后服务返回用户所需的信息 (w_{out}), 此外本文定义这些输入输出信息的数据大小为 $Datasize$, 这些输入输出信息的集合称为 $Concept$, 大小单位为 Mb。

1.3 服务组合的定义

在云边环境中为了能更好的满足用户需求,我们往往需要将多个服务的资源进行共享和利用,即进行服务组合。本文定义一个服务组合问题如下:

- W 是一组有限的服务集合。
- C_{in} 是输入参数的有限集合。
- C_{out} 是输出参数的有限集合。
- Q 表示服务质量标准集合。

服务组合问题可以映射到规划图中,表示如下:

- $w \leftrightarrow a$
- $w_{in} \leftrightarrow preconed(a)$
- $w_{out} \leftrightarrow effects(a)$
- $C_{in} \leftrightarrow S_0$
- $C_{out} \leftrightarrow goal$

这里给出两个服务 w_i, w_j , 它们之间有三种服务执行结构^[10], $w_i; w_j$ 表示的顺序执行结构, $w_i || w_j$ 表示并行执行结构, $w_i | w_j$ 表示选择执行结构。

1.4 移动模型

随着边缘计算服务的普及程度不断提高。越来越多的服务请求发生在边缘环境中。而用户的移动路径对服务上传、选择以及服务组合结果的得到时间起着重要影响,最终也会影响到服务组合解决方案的质量,因此需要对移动路径进行建模。

假设用户初始距离基站的距离为 D , 移动的方向角为 θ , 且用户以一个恒定的速度移动, 在经过时间 t 之后到达最终位置, 此时用户和基站的距离为 d 。这里可以通过公式 (1)^[17] 计算出 d 的值。

$$d = \sqrt{(vt)^2 - 2Dvt \times \cos(\theta) + D^2} \quad (1)$$

其中, v 为用户的移动速度。本文将用户初始位置和移动速度结合起来, 将用户在时间 t 内的移动路径考虑进去, 从而得到用户和基站之间的实际距离。

通过对移动路径的建模, 本文可以更加准确地预测用户与基站之间的距离, 从而为服务上传、选择和服务组合的过程提供更合理的参考。这将有助于改善服务组合解决方案的质量, 并提高用户移动环境下的服务体验。

2 云边环境下服务组合

在这一节本文主要介绍云边环境中基于规划图的服务组合过程, 包括从用户上传到下载结果的公式计算方法。

图 1 中展示了本文服务组合方法的主要结构, 具体内容如下:

移动边缘和云环境: 主要模拟了云服务器以及边缘服务器的分布, 并且服务分布到各个服务器中。每个边缘服务器都连接到互联网, 并能够与其他服务器进行数据通信。同样, 每个云服务器都连接到互联网, 并能够与其他服务器传输数据。在移动边缘和云计算环境下, 用户的移动路径部分决定了上传、服务组合和下载的时间。因此, 本文对用户的移动路径进行建模。

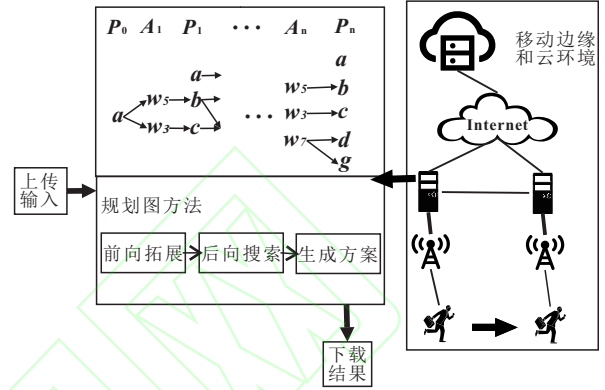


图 1 云边环境下基于规划图的服务组合

基于规划图的服务组合: 移动用户将他的请求以及输入上传到附近的基站, 接着将其上传到服务器。之后在云边服务环境中进行规划图的前向拓展过程, 直到找到满足用户需求的输出结果, 最后用户从靠近他的基站接收解决方案。

在云边环境中, 服务器之间的异构性是存在的。当用户使用这些服务时, 由于服务的差异, 它们在服务组合中所需的资源和时间也不同。在这里文本定义在云边环境中总共的响应时间为 T_{all} , 是用户从发出请求到接收到解决方案的总时间, 可由下式计算:

$$T_{all} = T_{upload} + T_{comp} + T_{download} \quad (2)$$

其中 T_{upload} 是指上传请求的时间, $T_{download}$ 是下载解决方案的时间, T_{comp} 是服务组合的响应时间, 对于组合时间本文将在案例说明中介绍具体的计算过程。

对于上传请求的时间, 包括用户请求传输到基站的时间 T_{u2b} 以及基站到边缘服务器的传输时间 T_{b2s} 。 T_{upload} 表示为:

$$T_{upload} = T_{u2b} + T_{b2s} \quad (3)$$

同样的对于下载时间, 如果最后一个任务的输出结果不在用户所处范围内的服务器中, 可能在边缘或者云服务器中, 则下载时间包括服务器之间的传输时间 T_{s2s} , 服务器到基站的传输时间 T_{s2b} , 基站

到用户的传输时间 T_{b2u} 。 T_{download} 表示为:

$$T_{\text{download}} = T_{s2s} + T_{s2b} + T_{b2u} \quad (4)$$

请求上传到基站 T_{u2b} 的时间和从基站 T_{b2u} 下载结果的时间计算如下:

$$T_{u2b} = \sum \text{Input}_{\text{DataSize}} / r(B, g) \quad (5)$$

$$T_{b2u} = \sum \text{Goal}_{\text{DataSize}} / r(B, g) \quad (6)$$

其中 $\sum \text{Input}_{\text{DataSize}}$ 表示上传的任务的大小, $\sum \text{Goal}_{\text{DataSize}}$ 表示组合目标结果的大小, $r(B, g)$ 表示可实现的数据传输速率, 依据香农公式, 其可由下面的式子计算得到:

$$r(B, g) = B \times \log_2(1 + tp \times g / \sigma^2) \quad (7)$$

其中 σ^2 为接收机处的噪声功率, tp 为设备的无线发射功率, g 为信道功率增益, 可计算为 $h_0 * d^{-\gamma}$, 其中 h_0 为接收功率, d 为设备与服务器之间的距离, 为路径损耗系数。

此外服务器之间的传输时间 T_{tran} 也取决于数据大小和带宽 BW_{s2s} , 可由下式计算:

$$T_{\text{tran}} = \text{Concept}_{\text{DataSize}} / BW_{s2s} \quad (8)$$

3 修复模型

3.1 服务修复定义

在现实世界中, 由于用户需求的改变或用户移动而带来的变化都可能导致可用服务和组合发生变化。为了解决这个问题, 文章^[15]提出了一种在 Web 服务环境中的修复方法。他们通过一种基于人工智能的方法来修复受损的规划图, 并获得更新的服务组合方案, 然而, 他们的方法有局限性。他们认为损坏或消失的服务产生的所有参数都必须修复。此外, 该方法没有进一步考虑服务 QoS 对整个修复过程的影响。

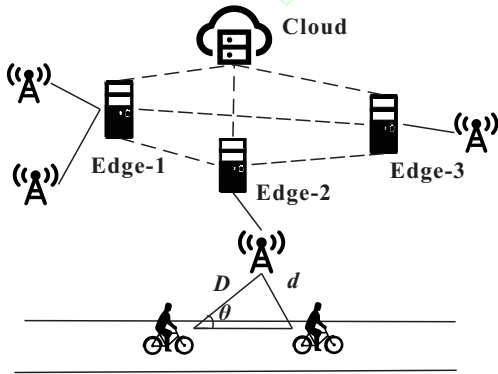


图2 案例图示

本文对该方法进行了改进与调整提出了一种规划图修复方法 (Repairing a Damaged Planning Graph for Service Composition, RDPGSC) 使其可以应用到

边缘计算环境中。本文的方法仍然基于原始计划, 因为本文相信尽管规划图已经被破坏, 但仍然有很多计划是可行的, 即大部分的服务还是可以被再次使用, 这可以为后续的工作提供帮助。

为了方便描述修复过程, 这里给出以下定义: G 表示被破坏后的规划图。 BP 表示在规划图中一些服务未满足的输入。 g 表示的是 $goal$ 集合中需要被修复的元素。 W 表示候选服务集合。

本文主要通过更新后的候选服务集合 W 中选择 w 修复规划图, 根据等式 (9) 选择服务。

$$f_1(G, w) = |g \cap w_{out}| + \alpha \cdot |P_m \cap BP \cap w_{out}| + |P_{m-1} \cap w_{in}| - |w_{in} - P_{m-1}| - \frac{(T_w - T_W^{\min}) \cdot \beta}{T_W^{\max} - T_W^{\min}} - w_{outsized} \cdot \delta \quad (9)$$

公式 (9) 用来计算规划图中除了第 A_n 最后一层 (n 层) 的服务的适应度值。 $P_m (0 \leq m < n)$ 表示正在修复的当前层。 $|P_m \cap BP \cap w_{out}|$ 表示 w 的输出元素可以满足的在第 P_m 层中的 BP 集合里的所需要的参数的数量。 $|P_{m-1} \cap w_{in}|$ 表示的是在第 P_{m-1} 层中元素中不能满足 w 输入的数量, $|w_{in} - P_{m-1}|$ 表示 P_{m-1} 层中不能满足 w 输入的参数个数。 T_w 表示服务的响应时间, T_W^{\max}, T_W^{\min} 表示所有服务的最大响应时间和最小响应时间, 公式中这一部分表示对候选服务响应时间的归一化计算。 $w_{outsized}$ 表示服务需要传输到其他服务器的最大输出数据的大小。 α, β , 和 γ 分别为权重系数。

$$f_2(G, w) = |g \cap w_{out}| \cdot \alpha + |P_{n-1} \cap w_{in}| - |w_{in} - P_{n-1}| - \frac{(T_w - T_W^{\min}) \cdot \beta}{T_W^{\max} - T_W^{\min}} + isNear \quad (10)$$

公式 (10) 是被用来计算第 A_n 层即最后一层的服务的适应度值。在公式中 $|g \cap w_{out}|$ 表示候选服务 w 可以满足的目标集 ($goal$) 的得分。 $|P_{n-1} \cap w_{in}|$ 表示在第 P_{n-1} 层的元素中不能满足 w 输入的数量。 $isNear$ 用于判断最后一层服务层中所选的服务是否位于用户当前基站范围内的服务器中。若位于范围内, 则可以缩短数据传输时间。其数值为 0 或 1。

3.2 案例说明

本节中将使用一个简单的例子来展示在云边环境中使用规划图进行服务选择的过程以及服务修复的具体过程。

首先介绍规划图的前向拓展过程。图 2 的例子中为代驾司机接单的过程, 具体为司机接收订单后

上传自己的信息集合 S_0 , 获取从他的位置到客户位置以及到达客户的目的地路径信息。表 1、表 2 和表 3 分别为事例中的参数信息以及服务描述, 包

表 1 参数描述

参数名	参数值	参数名	参数值
σ^2	-60dBm/Hz	γ	2
B	256MHz	tp	500MW
h_0	-30dB	BW_{c2c}	5Gbps
BW_{e2e}	3Gbps	BW_{e2c}	1.5Gbps
BW_{e2b}	2Gbps	θ	30°

括服务的输入和输出以及这些数据的大小 (单位为 Mb)。首先代驾司机将初始输入上传到最近的基站中。开始时, 代驾司机距离基站最近为 100m。其中用户的初始输入集合 S_0 里为 *Routetodes*, *RoutetoC* 上传到基站的时间可以计算为:

$$T_{u2b} = \sum Input_{Datasize} / r(B, g) \\ = 853ms$$

另外边缘服务器到基站之间的传输速度 BW_{b2e} 为 2Gbps。因此输入上传到边缘服务器的时间可以计算为:

$$T_{b2e} = \sum Input_{Datasize} / BW_{b2e} = 3.5ms$$

这里可以得到 T_{upload} 的时间为:

$$T_{upload} = T_{u2b} + T_{b2e} = 856.5ms$$

在这之后我们开始计算在云边环境中基于规划图的服务组合的响应时间 T_{comp} 。此时初始输入 S_0 在 Edge-2 中, 接着服务 *exDinfo* 调用输入 *Dinfo*, 而服务 *exDinfo* 位于云服务器, 因此还需要计算 *Dinfo* 的传输时间 T_{tran} 为 1.3ms, 此外 *exDinfo* 的响应时间 T 为 12ms, 通过计算可得 *Dcredit*, *DID* 产生的时间为 13.3ms。

表 2 案例数据大小

Concept	Datasize	Concept	Datasize
<i>Dinfo</i>	2	<i>Routetodes</i>	5
<i>Dcredit</i>	0.1	<i>GPSinfo</i>	5
<i>DID</i>	0.1	<i>Cloact</i>	2
<i>Dsite</i>	5	<i>Cdest</i>	1
<i>Faceinfo</i>	5	<i>Drivepath</i>	5
<i>Cphone</i>	5	<i>Csite</i>	1
<i>Cdestination</i>	1	<i>RoutetoC</i>	5
<i>Cinfo</i>	3		

此外本文规定, 在同一个服务器中的传输时间为 0ms。计算可得 *Cinfo* 的产生时间为 27ms。此外经过上述计算可得 *Dsite* 的产生时间为 29ms。这里 P_1 中产生 Concept 的时间就计算完毕了, 接着继续进行下面的规划图拓展过程, P_2 层中参数的产生时间的计算与前一层大致相同, 这里直接给出结果, 得到 *Csite*, *Cphone* 和 *Cdestination* 的时间为 47ms, 38ms 和 39ms。

表 3 案例服务描述

服务	w_{in}	w_{out}	位置	响应时间	描述
<i>extDinfo</i>	<i>Dinfo</i>	<i>Dcredit</i> , <i>DID</i>	Cloud	12ms	获取司机信息
<i>Locating1</i>	<i>Dinfo</i>	<i>Dsite</i>	Edge-1	28ms	获取司机位置
<i>FaceVer</i>	<i>Dinfo</i> , <i>Facepic</i>	<i>Cinfo</i>	Edge-2	27ms	人脸验证
<i>Locating2</i>	<i>Cinfo</i>	<i>Csite</i>	Edge-2	20ms	获取客户位置
<i>extPhone</i>	<i>Cinfo</i>	<i>Cphone</i>	Edge-3	10ms	获取客户手机信息
<i>extDest</i>	<i>Cinfo</i>	<i>Cdestination</i>	Edge-2	12ms	获取客户目的地
<i>getGPS</i>	<i>Cphone</i>	<i>CGPSinfo</i>	Edge-3	24ms	获得 GPS 信息
<i>findPathtoC</i>	<i>Dsite</i> , <i>Csite</i>	<i>RoutetoC</i>	Edge-3	15ms	获取到达用户路径
<i>findPathtodes1</i>	<i>Cdestination</i> , <i>Csite</i>	<i>Routetodes</i>	Edge-2	21ms	获取用户所在位置到用户目的地路径
<i>findPathtodes2</i>	<i>Cloact</i> , <i>Cdest</i>	<i>Routetodes</i>	Edge-2	19ms	获取用户所在位置到用户目的地路径
<i>findPathtodes3</i>	<i>Cloact</i> , <i>Cdest</i>	<i>Routetodes</i>	Cloud	24ms	获取用户所在位置到用户目的地路径
<i>extLocat</i>	<i>GPSinfo</i>	<i>Cloact</i>	Edge-2	16ms	获取客户位置

最后计算最后一层中得到 *RoutetoC* 的时间, 其需要的输入为 *Dsite* 以及 *Csite*, 因此在调用服务之前需要它们全部传输到服务器 Edge-3 中, 此时还需要计算两者到达时间 $29ms + 1.67ms < 47ms + 0.33ms$, 因此得到 *RoutetoC* 的时间为 $47ms + 0.33ms + 15ms = 63.3ms$ 。

计算 *RoutetoC* 同理结果为 68ms。最晚得到的结果是 *RoutetoC*, 因此服务组合的响应时间总共为 68ms。最后用户需要从服务器 Edge-3 和 Edge-2 中下载结果集合。

假设用户速度为 10m/s, 此时根据公式 (1) 计算, 用户距离基站的距离 d 为 92.1m。因此, 需要将

接下来,按照式(8)修复从 P_{n-1} 到 P_1 即 P_m 的规划图,修复和更新 BP 中的元素(第 10-28 行)。修复过程与前一部分大体相同,但是在这一部分本文不会因为某些不可被修复的元素而被迫停止修复过程,而是将其暂时存储到 BPQ 中,并在规划图中删除被其影响的服务以减少后续后向搜索空间以减少返回结果的时间(第 22-27 行)。当第一个 P 层被修复,但 BP 仍然不为空时,算法继续创建一个空的 P 层和一个空的 A 层,并按照式(9)修复这些层,修复过程也与前面类似,直到 BP 为空。其中 BPH 用来记录每层中的未被满足的参数,当 BP 与 BPH 有重合时则规划图修复失败,若成功则返回一个服务组合结果(第 29-45 行)。

算法 1 修复损坏后的规划图并返回一个解决方案

输入: 待修复的规划图 G , 规划图 G 中一组不满足的服务的输入集合 BP , 一组无法修复的参数集合 BPQ , 一组候选服务 W , 一组未满足输入的 $goal$ 集合 g 输出: 生成一个修复完成的规划图或者返回失败

```

step 1: while  $g \neq \emptyset$  do
step 2: 根据公式(10)在  $W$  中选择  $w$ 
step 3: if  $w \notin W$  then break
step 4:  $A_n \leftarrow A_n \cup w$ 
step 5:  $P_n \leftarrow P_n \cup w_{out}$ 
step 6:  $P_{n-1} \leftarrow P_{n-1} \cup w_{in}$ 
step 7:  $g \leftarrow g/w_{out}$ 
step 8:  $BP \leftarrow BP \cup (w_{in}/P_{n-1})$ 
step 9: end while
step 10: If  $g \neq \emptyset$  then return fail
step 11: for each  $P_m$  in  $G$ 
step 12: while  $BP \cap P_m \neq \emptyset$  do
step 13: 根据公式(9)在  $W$  中选择  $w$ 
step 14: if  $w \notin W$  then break;
step 15:  $A_m \leftarrow A_m \cup w$ ;
step 16:  $P_m \leftarrow P_m \cup w_{out}$ ;
step 17:  $BP \leftarrow BP/(P_m \cap BP)$ 
step 18:  $BPM \leftarrow BP$ 
step 19:  $BP \leftarrow BP \cup (w_{in}/P_{m-1})$ 
step 20:  $P_{m-1} \leftarrow P_{m-1} \cup w_{in}$ 
step 21: end while
step 22:  $BPQ \leftarrow BPM \cup BPQ$ 
step 23: If  $BPQ \neq \emptyset$  then
step 24: 删除  $A_m$  层受  $BPQ$  影响的服务

```

```

step 25: end if
step 26:  $BP \leftarrow BP/BPQ$ 
step 27:  $BPH = BP$ 
step 28: end for
step 29: while  $BP \neq \emptyset$  do
step 30: 在规划图前面插入一个新的空  $P_{new}$ 
和  $A_{new}$  层
step 31:  $P_{new} = P_0/BP$ 
step 32: while  $BP \neq \emptyset$  do
step 33: 根据公式(9)在  $W$  中选择  $w$ 
step 34: if  $w \notin W$  then break;
step 35:  $A_{new} \leftarrow A_{new} \cup w$ 
step 36:  $P_0 \leftarrow P_0 \cup w_{out}$ 
step 37:  $BP \leftarrow BP/(P_0 \cap BP)$ 
step 38:  $BP \leftarrow BP \cup (w_{in}/P_{new})$ 
step 39: end while
step 40:  $A_0 \leftarrow A_{new}$ 
step 41:  $P_0 \leftarrow P_{new}$ 
step 42: if  $BP \neq \emptyset$  then break
step 43:  $BP \cap BPH \neq \emptyset$  then break else 添加
 $BP$  到  $BPH$ 
step 44: end while
step 45: if  $BP \neq \emptyset$  then 按照规划图的反向搜索
方法返回组合结果或者失败

```

4 实验结果及其分析

为了验证本文提出的修复方法的性能进行了下面的实验来模拟云边环境下规划图的修复和重新规划问题。仿真是在以下配置的计算机上进行的。(1)CPU: Intel Core i5-10400F at 2.90 GHz,(2)内存:16GB DDR4 2666MHZ,(3)硬盘:KINGSTON SNVS 500GB,(4)操作系统:Windows 10 Professional 64 位。

在本文的实验中,本文采用了两个数据集^[18]进行实验。这些数据集的格式是基于本体、服务和编排的 OWL 和 WSDL 表示。这些数据采用了语义 Web 服务的概念,其中使用接口和服务质量特征来描述 Web 服务。数据集包含了大量的服务和输入(w_{in})/输出(w_{out})参数。其中第一个数据集有 572 个服务和 1578 个输入/输出参数。第二个数据集有 4129 个服务,超过 12388 个输入/输出参数。输入/输出参数对应于 OWL 中定义的类。

在模拟的云边环境中,本文设定了 5 个云服务器以及 10 个边缘服务器,之后将数据集中的服务按照一定顺序布置在这些服务器中。此外设置了三个

评估标准进行比较: 用户从上传到得到结果的时间、解决方案中的服务数量和重新规划与修复方法对比原始计划的差距, 即与原始方案中不同服务的个数。用户从上传数据到获得结果所需的时间包括了重新获得结果的时间开销, 其中包括修复和重新规划过程的开销。

实验过程中, 本文按照一定比例从服务集合中移除候选服务, 在同一个服务损坏的环境中比较修复方法和重新规划方法得到结果所需的时间。通过图 5 可以发现重新规划通常需要更多的时间来获得结果, 而本文的修复方法 (RDPGSC) 以及原始的修复算法可以更快地得到服务组合结果。并且本文中

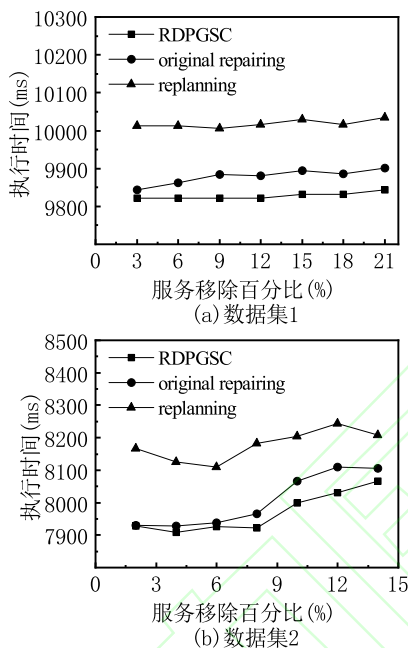


图 5 获取解决方案的时间对比

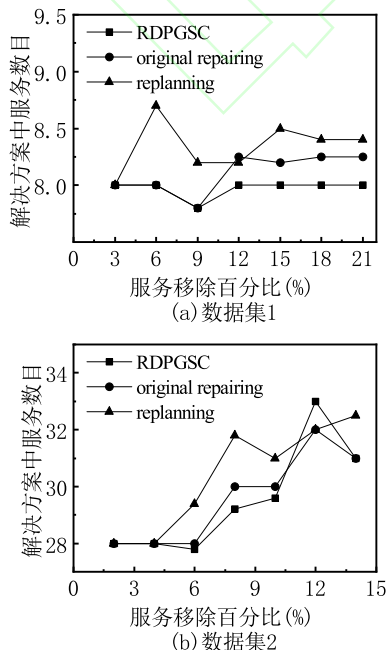


图 6 获取解决方案中的任务数量的对比

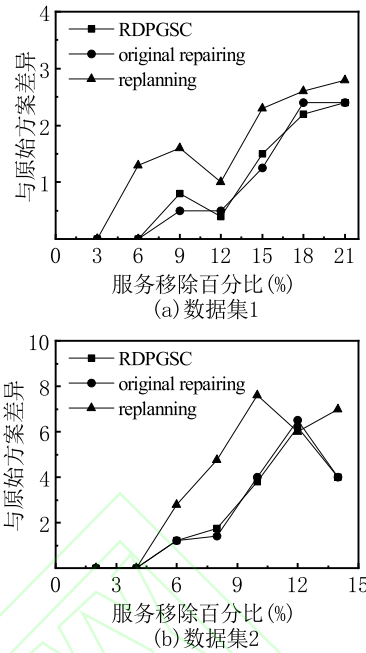


图 7 获取解决方案与原始方案的差异

RDPGSC 只需要更少的时间, 这是由于本文方法在修复过程中会优先选择响应时间更短的服务。然而, 随着服务被破坏的比例增加, 两种方法所需的时间也会增加。在较高的比例情况下, 修复方法可能会失效, 无法返回一个组合方案。例如在数据集 2 中, 如果删除超过 12% 的服务后的十次运行结果中, 只有 4 次修复成功。但通常情况下, 修复方法可以更快地得到结果。在数据集 2 中可以观察到当删除更多服务时, 重新规划的时间略有减少, 主要是因为随着可用候选服务的较少, 问题规模也随之减小。

图 6 显示了在修复规划图和重新生成规划图后解决方案中的服务数量。从图中可以看出, 修复后产生的解决方案需要更少的服务。并且本文的方法与原始方法比需要的服务也较少。这里的解释是在修复过程中, 本文在原始规划图的基础上进一步完善规划图, 因此修复的解决方案中的服务数量更接近原始方案。这一点可以从图 7 中观察到, 两种修复方法产生的解决方案比重新规划更接近原始规划图。当然, 随着移除服务比例的增加, 无论是修复还是重新规划, 与原始规划图都存在差距, 这是可以理解的。但是, 修复也不会受限于原始规划图方案, 并且所需的执行时间也会更少。

5 结 论

本文提出了一种基于规划图的云边服务组合修复方法, 以应对在云边环境中面临的服务组合和规划挑战。通过使用规划图来建模服务间的依赖关系和约束条件, 来满足用户需求。修复算法通过搜索启发式函数快速修复部分规划图, 直到满足所有输

出条件为止。相比重新规划,修复方法能更快地找到服务组合解决方案,并最大限度地重用部分规划图内容。

实验证实了修复方法的优势,因此该方法能够满足用户需求并应对云边服务环境中的动态变化和不确定性。未来研究可以进一步改进修复算法,提高修复效率和性能,并将其应用于更广泛的实际场景。

6 参考文献

参考文献 (References)

- [1] Zhu M, Yu F L, Yan X K, et al. Scaling up mobile service selection in edge computing environment with cuckoo optimization algorithm[C]. 2021 IEEE International Conference on Services Computing. Chicago: IEEE, 2021: 394-400.
- [2] 李燕君, 蒋华同, 高美惠. 基于强化学习的边缘计算网络资源在线分配方法 [J]. 控制与决策, 2022, 37(11): 2880-2886.
(Li Y J, Jiang H T, Gao M H, et al. Reinforcement learning-based online resource allocation for edge computing network[J]. Control and Decision, 2022, 37(11): 2880-2886.)
- [3] 吕玲玲, 杨志鹏, 张磊. 基于合约设计的移动边缘计算任务卸载策略研究 [J]. 控制与决策, 2019, 34(11): 2366-2374.
(Lyu L L, Yang Z P, Zhang L, et al. Contract theory based task offloading strategy of mobile edge computing[J]. Control and Decision, 2019, 34(11): 2366-2374.)
- [4] Guo S T, Xiao B, Yang Y Y, et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing[C]. IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. San Francisco: IEEE, 2016: 1-9.
- [5] 韩敏, 张丽君. 基于多样性检测的双子群多目标粒子群算法 [J]. 控制与决策, 2017, 32(12): 2268-2272
(Han M, Zhang L J. Bi-group multi-objective particle swarm optimization algorithm based on diversity metric[J]. Control and Decision, 2017, 32(12): 2268-2272.)
- [6] Li J, Yan Y H, Lemire D. Scaling up Web service composition with the skyline operator[C]. 2016 IEEE International Conference on Web Services. San Francisco: IEEE, 2016: 147-154.
- [7] Boussalia S R, Chaoui A. Optimizing QoS-based web services composition by using quantum inspired cuckoo search algorithm[C]. Mobile Web Information Systems: 11th International Conference. Barcelona: Springer International Publishing, 2014: 41-55.
- [8] 韩红桂, 徐子昂, 王晶晶. 基于 Q 学习的多任务多目标粒子群优化算法 [J]. 控制与决策, 2023, 38(11): 3039-3047.
(Han H G, Xu Z A, Wang J J. A Q-learning-based multi-task multi-objective particle swarm optimization algorithm[J]. Control and Decision, 2023, 38(11): 3039-3047.)
- [9] Li j, Yan Y H, Lemire D. Full solution indexing for top-k web service composition[J]. IEEE Transactions on Services Computing, 2016, 11(3): 521-533.
- [10] Zhu M, Fan G D, Li J, et al. A service composition approach based on overall QoS and modified graphplan[J]. International journal of web and grid services, 2019, 15(4): 319-339.
- [11] 韩敏, 段彦忠. 融合可信性评价的 Web 服务组合 QoS 优化 [J]. 控制与决策, 2020, 35(8): 1859-1865.
(Han M, Duan Y Z. QoS optimization of Web services composition incorporating with credibility evaluation[J]. Control and Decision, 2020, 35(8): 1859-1865.)
- [12] Friedrich G, Fugini M G, Mussi E, et al. Exception handling for repair in service-based processes[J]. IEEE transactions on software engineering, 2010, 36(2): 198-215.
- [13] Church J, Motro A. Recommending Service Repairs[C]. 2014 IEEE 8th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems. Victoria: IEEE, 2014: 11-18.
- [14] Eder J, Mangler J, Mussi E, et al. Using stateful activities to facilitate monitoring and repair in workflow choreographies[C]. 2009 Congress on Services-I. Los Angeles: IEEE, 2009: 219-226.
- [15] Yan Y H, Poizat P, Zhao L D, et al. Repairing service compositions in a changing world[M]. Berlin, Heidelberg, Springer Berlin Heidelberg, 2010, 17-36.
- [16] Yan Y H, Chen M. Anytime QoS-aware service composition over the GraphPlan[J]. Service oriented computing and applications, 2015, 9(1): 1-19.
- [17] Bettstetter C, Hartenstein H, Perez-Costa X. Stochastic properties of the random waypoint mobility model: epoch length, direction distribution, and cell change rate[C]. Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems. New York: Association for Computing Machinery, 2002: 7-14.
- [18] WS-Challenge, Testsetgenerator2009[DB/OL]. (2009-08-09)[2023-08-15]
<https://code.google.com/p/wsc-pku-tcs/downloads/list>.

作者简介

高志浩 (1998—), 男, 硕士生, 从事服务计算、服务选择的研究, E-mail: 21505020607@stumail.sdut.edu.cn;

李静 (1986—), 女, 副教授, 博士, 从事服务计算、计算机网络的研究, E-mail: li_jing@sdut.edu.cn;

祝铭 (1983—), 男, 讲师, 博士, 从事服务计算、边缘计算的研究, E-mail: zhu_ming@sdut.edu.cn.

刘彤阳 (1999—), 男, 硕士生, 从事服务计算的研究, E-mail: liutongyang0716@163.com.

鲁瑞 (1998—), 男, 硕士生, 从事计算机视觉的研究, E-mail: u7353434@anu.edu.au.