计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100152

一种基于微服务的检察业务服务封装方法

陆懿帆　曹芮浩　王俊丽　闫春钢
同济大学电子与信息工程学院嵌入式系统与服务计算教育部重点实验室　上海 201804
(374329974@qq.com)

摘　要　微服务架构是一种新兴的服务架构风格,在处理复杂服务系统时表现出运行高效、部署灵活等特性,相较于单体式服务架构,能够提供更好的业务管理和服务支持。针对检察院复杂的办案业务,需要对服务进行组合封装,形成新的增值服务以满足用户需求。但是,单独进行服务质量驱动的服务封装不能满足检察业务的需求,因此,结合服务功能和服务质量,提出了微服务架构下图规划算法的改进方法(Improved Graphplan Under MicroService Architecture,IGMA)。该方法首先对服务、用户请求建立数学模型,其次综合服务的功能需求和非功能需求,在不同案件类型下为用户提供多种组合方案,最后建立服务工作流,完成案件服务封装。该方法能够智能判断服务组合结构中的分支结构,并对不同的分支结构建立不同的组合方案。实验结果表明,该方法在服务封装的时效性和准确性上有了较大的提升。

关键词:微服务;服务质量;服务组合;服务封装;图规划
中图法分类号　TP393

Method of Encapsulating Procuratorate Affair Services Based on Microservices

LU Yi-fan,CAO Rui-hao,WANG Jun-li and YAN Chun-gang
Key Laboratory of Embedded System and Service Computing,Ministry of Education,College of Electronics and Information Engineering,Tongji University,Shanghai 201804,China

Abstract　Microservice architecture is an emerging style of service architecture,which is characterized by efficient operation and flexible deployment when dealing with complex service systems. Compared with monolithic architecture,it can provide better business management and service support. In view of the complex case of the procuratorate affair,it is necessary to combine and encapsulate the services to form new value-added services to meet the needs of users. However,quality-of-service driven service encapsulation alone cannot meet the needs of procuratorate affair. Therefore,combining service functions and quality of service,an improved graphplan under microservice architecture (IGMA) is proposed. Firstly,the method establishes a mathematical model for the service and user request,then integrates the functional and non-functional requirements of the service,and provides users with a variety of combination schemes under different case types. Finally,the service workflow is established to complete the case service encapsulation. This method can intelligently judge the branch structures in the service composition structure and establish different composition schemes for different branch structures. Experimental results show that the proposed method improves the timeliness and accuracy of service encapsulation.

Keywords　Microservice,Quality of service(QoS),Service composition,Service encapsulation,Graphplan

1　引言

检察业务系统错综复杂,存在很多子服务。以案件公诉流程为例,它包括受案模块、办理模块、查询模块等,每个模块由多个子服务构成。使用单体式服务架构会导致服务爆炸、系统开发和维护困难等问题[1]。而且,办案人员在案件审查过程中会进行重复工作,造成工作效率低,无法达到智能化办案的目标。

微服务架构作为一种新兴的服务架构风格,相较于单体式服务,为高效管理和使用复杂服务系统提供了技术支持。它将单一的应用程序分解为一组小型的服务,所有小型的服务独立运行、独立部署、互相协作完成请求。同时,微服务架构中的服务可被当作进程外的组件,独立部署、独立调用,并且,微服务架构还带有网关、负载均衡、配置管理、熔断器等基础设施,能够对服务集群进行有效的管理[2]。而且在检务系统中,单一的服务难以满足检察官的需求,需要对多个服务进行组合封装,进而为检察官提供更多个性化的服务组合方案。

本文的主要工作包括:首先对服务库中的服务和用户请

到稿日期:2019-11-20　返修日期:2020-03-27　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研发计划(2018YFC0831403)
This work was supported by the National Key Research and Development Project (2018YFC0831403).
通信作者:闫春钢(cgyan2@163.com)

求进行解析，从而筛选出符合案件个性化需求的功能各异的服务；其次，综合分析服务之间的输入和输出以及参数格式之类的信息，得到服务之间的逻辑关系；最后，使用服务组合算法将服务封装为工作流，完成服务封装。

在服务组合算法中，图规划算法是一种常见的服务组合封装的算法，其通过遍历搜索建立服务规划图，根据归一化后的服务质量，借鉴 Dijkstra 的单源最短路径的思想，搜索最佳的服务组合路径，从而完成服务的组合封装。但是，图规划算法存在以下问题：1）通过图规划算法进行服务组合需要遍历整个服务库，在遍历过程中，存在很多冗余服务，降低了算法的运行效率。2）在图规划的算法中需要先建立服务规划图，服务规划图通常规模较大，算法的空间性能不佳。3）图规划算法通过调整归一化函数来实现不同的服务偏好下的服务组合，此方法在处理并行结构的服务组合问题时存在一些问题。因为并行结构对于整体服务结构综合服务质量的运算往往取某一个分支的最大值或者最小值，对于其他服务分支的优化可能不仅无法优化整体服务组合，还会降低整体服务组合的其他服务指标的服务质量。

针对图规划算法的 3 个问题，本文做出了以下改进：1）通过建立不同偏好下的任务库来减小每一次搜索所需要的搜索空间，从而提高服务组合算法的时间性能；2）通过建立任务与任务之间的逻辑关联来代替建立服务规划图，提高服务组合算法的灵活性和算法的空间性能；3）在处理并行结构的服务组合时，对于不同的服务分支采取不同的处理方式，在对整体服务组合进行优化的同时，提高其他服务分支的服务性能。

本文第 2 节介绍微服务架构以及现有的服务组合方法；第 3 节介绍服务封装问题的形式化描述，包括服务、请求、工作流、任务和任务关联的基本定义，并将服务封装问题转化为将服务组合为工作流的问题；第 4 节介绍本文提出的改进图规划算法；第 5 节给出在 WS-Dream 数据集[3-7]和 QWS 数据集[8-9]上进行的实验，以验证本文方法的可行性和有效性，并给出实验细节和结果分析；最后总结全文并展望。

2 相关工作

本节介绍微服务架构下服务组合封装的相关工作，主要包括服务组合方法的研究现状和微服务架构。

2.1 服务组合方法

在互联网上，大量的服务具有相同的功能但却具有不同的 QoS 属性[10-11]，将服务进行组合来构造出增值的服务从而满足用户的需求，一直是服务计算领域的研究热点，下面就服务组合的多个方面的相关工作进行论述。

服务组合主要分为基于服务功能的服务组合方法[12-14]和基于非服务功能的服务组合方法[15-20]。在基于服务功能的服务组合问题的研究上，文献[21]提出在服务匹配中除了基于输入/输出类型的匹配，还需要对实际的数据类型进行匹配。文献[22]为了进一步提高服务匹配的精确性，引入了语义匹配度的概念，以实现具有多种匹配值的服务类别语义匹配，并给出了综合语义匹配值的计算方法。文献[23]通过语义相似度量化服务属性和参数概念之间的相符程度，并在此基础上引入服务二部图的概念，将两个服务关联程度的计算转化成二部图的匹配程度度量。文献[24]提出了一种基于语义匹配的 Web 服务模型，该模型能够描述前驱服务和后继服务，并能够对服务进行顺序组合和并发组合。基于服务语义匹配的服务组合方法通常被用来完成服务逻辑功能上的组合，其基本能够满足用户的需求，但是不能保证服务组合的服务质量。

在基于非服务功能（即基于服务质量）的组合问题的研究上，服务质量优化问题是一个 NP 问题，文献[25-30]提出通过人工智能规划，在 Markov 模型中对服务进行建模。文献[31-34]提出了多 agent Q-learning 模型来解决服务组合问题，从而加快各自的学习速度，改善服务组合质量。强化学习算法的准确率较高，但是需要对服务数据集进行长时间的训练，算法的时间性能较差。文献[35]将服务组合问题转化为混合整数线性规划问题，并通过线性规划算法进行服务组合。文献[36]解决了服务间的匹配问题，认为一个服务的输出参数可以用作另一个服务的输入参数，通过向后的搜索算法解决服务组合问题。文献[37]提出了双向的搜索算法，从输出接口和输入接口更快地解决服务组合问题。文献[38]基于用户偏好，使用模糊分析层生成服务的综合服务质量，并使用动态阈值对竞争力较差的服务进行剪枝。

2.2 微服务架构

传统的应用系统具有规模较小、结构简单、用户群体小、实时通信要求低等特点，因此通常采用单体式架构。单体式架构是指将整个软件系统的功能模块以及运行数据等作为整体看待，统一进行设计、开发、打包以及部署运行。但是，单体式架构业务逻辑复杂、代码庞大、灵活性差；单体式架构受其架构限制，所有模块采用相同的语言和框架，难以实现协同服务，整体的创建和部署时间极长，成本高，可伸缩性差。

在检察院的业务场景下，系统的每次调整都需要对应用程序中的某个小部分进行变更，如果使用常规的单体式架构，每次的局部变更都需要对整体架构进行重新构建并且重新部署。但是使用微服务架构时，无须编译、部署整个应用，只需变更局部模块即可。因此使用微服务架构能够使发布更加高效，同时降低对系统环境造成的风险，易于开发人员对系统进行可持续的维护。

微服务架构通常由多个微服务构成，其中，每一个微服务都对应于一个独立的业务功能，并且微服务会针对主要功能定义一些基本的操作。每一个微服务具有一个单一业务功能，微服务架构就是将庞大的整体应用拆分为一组服务，在不改变整体功能的同时，将应用程序分解为可管理的模块和服务，通过定义良好的接口表示服务边界，使用微服务架构将服务模块化，同时保持其高可维护性和高开发效率。

如今，微服务架构下的组合方案主要分为 4 种。传统的方法是借鉴 Web 服务的组合调用方法，采用 Choreography Description Language（CDL）和 Business Process Execution Language（BPEL）来描述服务之间的交互、协作以及通信过程，但是这种基于句法的描述语言会随着服务数量的增加而越来越复杂，难以应用到大规模的服务组合中。Medley 是一种基于事件驱动的轻量级服务组合平台，通过编译器生成一种专门的服务描述语言，并部署在 Medley 平台上，从而实现

大量的服务之间的组合，但是该服务描述语言特殊、实用性不强。Petri 网模型是一种用于对微服务组合进行定义的方法，该方法偏重于服务建模，不太适用于实际中的微服务组合。Netflix Conductor 是当前比较流行的借鉴 Web 服务组合策略进行微服务组合的解决方案，其通过对服务和工作流的定义来实现微服务的编排，基于 JSON DSL 的蓝图定义执行流程。同时，Conductor 遵循基于 RPC 的通信模型，其中服务在不同的机器上运行，服务之间可以通过 http 调用另一个服务，并使用轮询模型来管理工作队列。基于此，本文使用 Netflix Conductor 作为工具，将服务组合转化为 Conductor 中的形式语言以完成服务封装。本文将采用 Netflix Conductor 进行微服务架构下服务的封装研究。

另外，在检察院的业务场景下，不同的业务需求需要选择不同的开发语言进行实现。但是，在单体式架构中，无法使用不同的语言开发同一个架构，因此，研究者提出了容器化的方法。容器方法使用了内核接口，允许不同的容器共享相同的内核，同时，容器方法能够完全屏蔽程序的语言和运行环境，实现细粒度的服务封装。Docker 是一种容器化的解决方案，其具体做法是将服务作为一个镜像封装在容器中，每个容器都能够实现其特定的功能。通过使用 Docker 引擎对每个服务进行容器化，可以实现细粒度的封装，从而实现跨语言和高效部署。

3 服务封装问题的形式化描述

一个服务库中会存在大量服务，当没有一种单一服务可以满足收到的请求时，就需要智能组合多个单一服务来满足请求。本文提出的服务封装方法的实质是智能匹配组合服务以完成用户请求。这样，服务封装问题就可以转化为从服务库中寻找一个有限的服务集合，使这些服务能够按照一定的流程结构组织起来，从而形成一个工作流。这个工作流能够对用户所提供的输入进行处理并得到符合用户需求的输出，在满足用户需求的基础上提升工作流的整体 QoS。首先，给出服务、用户请求和工作流的定义。

定义 1(服务) 一个服务可以用一个四元组表示 $S=(SN,SI,SO,QoS)$。其中，SN 表示服务的名称，服务的名称与其唯一功能对应；SI 表示服务预期输入的值，为服务的输入参数有限集；SO 表示服务预期输出的值，用于记录服务的输出，为服务的输出参数有限集；QoS 表示服务的服务质量集合，用一个向量表示，包括服务的响应时间、吞吐量、可靠性等。

定义 2(请求) 一个用户请求可以用一个四元组表示 $R=(RN,RP,RI,RO)$。其中，RN 表示用户请求的名称；RP 表示用户以及检察官对于案件服务封装的偏好；RI 表示用户请求的输入参数所对应的语义概念；RO 表示用户期望的输出所对应的语义概念。

定义 3(工作流) 一个工作流可以用一个五元组表示 $W=(WN,Ts,Edges,WI,WO)$。其中，WN 表示工作流的名称；Ts 为工作流所包含的一系列的任务集合；$Edges$ 表示工作流的边的集合；WI 为工作流的输入参数列表，用于记录工作流所需要的输入；WO 为工作流的输出参数列表，用于表示工作流的输出。

定义 4(任务) 任务 T 表示工作流中一个工作流程的节点，通常表示在该工作流节点待选服务的集合。$T_i=\langle S_{i1},S_{i2},S_{i3},\cdots,S_{im}\rangle$ 表示对于任务节点 T_i 的待选服务的集合，在集合中的每一个服务都能够提供同样的功能和不同的 QoS。

定义 5(任务关联) 任务关联 $Edge$ 表示工作流中两个任务节点之间的连接关系，$Edge=\langle T_i,T_j\rangle$ 表示任务节点 T_i 为任务节点 T_j 的前置任务，同时 T_j 也是 T_i 的后置任务。

服务的结构主要分为 3 种：顺序结构、分支结构和并行结构。由于检察业务系统的特殊性，本文对于服务质量的考虑限于响应时间(ResponseTime，RT)、吞吐量(Throughput，TP)、可靠性(Reliability，RE)和综合服务质量。

在顺序结构中，整体服务结构的服务指标的计算公式如下：

$$RTS=\sum_{i=1}^{n}RT(S_i) \tag{1}$$

$$RES=\prod_{i=1}^{n}RE(S_i) \tag{2}$$

$$TPS=\sum_{i=1}^{n}TP(S_i) \tag{3}$$

在分支结构中，整体服务结构的服务指标的计算公式如下：

$$RTB=\sum_{i=1}^{n}p_i*RT(S_i) \tag{4}$$

$$REB=\prod_{i=1}^{n}p_i*RE(S_i) \tag{5}$$

$$TPB=\sum_{i=1}^{n}p_i*TP(S_i) \tag{6}$$

在并行结构中，整体服务结构的服务指标的计算公式如下：

$$RTP=\max\{RT(S_i)\} \tag{7}$$

$$REP=\prod_{i=1}^{n}RE(S_i) \tag{8}$$

$$TPP=\min\{TP(S_i)\} \tag{9}$$

其中，S_i 表示每一个任务节点最终选择的服务，p_i 表示在分支结构中每一个分支选取的概率系数。大部分服务组合算法仅可以组合顺序执行的服务，而不能组合包含分支结构和并行结构的服务。但是，分支结构和并行结构的服务组合普遍存在于检察院的业务中，比如，在案件移交过程中会同时移交至刑事检察部和民事行政监察部。在类似的情况下就会产生并行服务。本文提出的组合方法能够很好地解决服务中的并行结构问题，并且时间效率等也有了很大的提升。

另外，在构造工作流时，工作流的组成部分并不一定完全是一个个细粒度子服务，还有可能是粗粒度的子工作流。也就是说，在检察院的业务场景下，常常会出现工作流互相嵌套的场景，因此工作流同样可以被视为一个服务进行处理。

4 服务组合封装方法——改进的图规划算法

服务封装可以帮助用户选择、组合、协调单个服务，从而形成满足用户需求的工作流。假设用户需要安排一次紧急案件的案件审查，其中可能包括证据搜索、财产冻结、批准逮捕等服务。服务封装模块能够自动匹配组合这些对应的工作流来满足用户的需求。

封装过程分为 3 个阶段：第一阶段，根据服务、请求、工作

流的具体功能，对服务、请求、工作流建立数学模型[4]，数学模型包括输入输出等信息；第二阶段为服务组合封装算法阶段，由于服务的各项指标的不同，需要对服务进行指标量化，根据量化后的服务质量对服务进行筛选，根据功能逻辑对服务进行匹配，依据用户的请求选择初始服务，依据案件类型、用户的语义描述等属性动态调整组合策略，构建不同偏好下的不同组合方案，以满足用户的个性化需求；第三阶段基于服务间的通信机制将服务组合封装为 Conductor 中的语义形式，完成服务封装。

在服务封装模块中，将会部署本文的服务组合封装算法，即微服务架构下改进的图规划算法 IGMA。首先，本文针对微服务架构根据用户需求对每一个服务的服务质量进行归一化计算；其次，根据每一个服务归一化后的服务质量进行筛选，从而得到最优的服务候选集，再构建服务依赖关系；最后，根据用户需求对服务进行封装。图 1 给出了微服务架构下服务封装方法的框架，微服务封装模块中的 IGMA 算法分为以下 4 个步骤：1）指标量化。根据不同的案件类型，对每一个服务的服务质量在不同的案件类型下进行归一化计算。2）服务筛选。根据当前用户的服务请求和处理的案件类型，针对检查业务系统的特殊性，对服务进行筛选，在每个状态节点下获取最优的服务。3）任务匹配。根据每个服务节点的输入输出参数集合的语义描述，构建任务关联。4）服务组合。以用户请求的输入为起点，以用户请求的输出为终点，构建服务组合并完成封装。

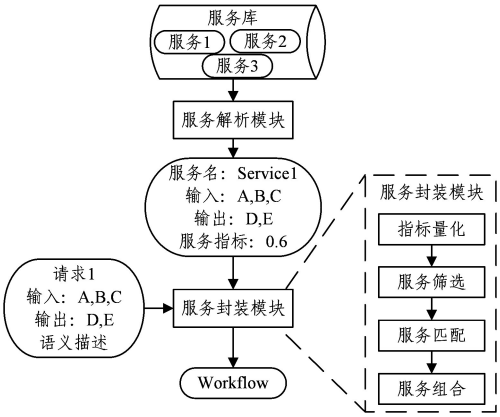

图 1 微服务架构下服务封装方法框架

Fig. 1 Service encapsulation method under microservice architecture

4.1 服务质量归一化

在对服务进行组合优化时，一项服务存在多个服务质量指标，如响应时间、可靠性、吞吐量、加工性能、可靠性、合规性、最优行为、延迟、自描述程度等。这些属性有不同的取值范围，且由于不同属性的取值范围可能相差较大，因此无法直接通过属性值的大小来评价服务的优劣。另外，考虑到有些服务质量的值越大越好（如吞吐量），而有些服务质量的值则是越小越好（如响应时间），因此，我们需要将所有服务的服务质量归一化。对于类似于吞吐量的服务指标，我们采取的归一化策略是用当前服务的吞吐量除以所有服务中吞吐量的最大值，从而得到一个归一化后的相对吞吐量；对于类似于响应时间的服务指标，我们采取的归一化策略是用所有服务中响应时间的最小值除以当前服务的响应时间，从而得到一个归一化后的相对响应时间。

在检察业务系统中，服务较为复杂，不同的案件需要不同的处理方式，比如民事案件更加注重案件的吞吐量，重大案件注重于案件的可靠性等。基于上述情形，本文根据案件类型计算每个服务在该案件类型下的服务质量。

算法的输入为案件类型和服务质量信息，其中服务质量信息为一个向量。综合调研检务系统的实际情况后，本文选择了检务系统中最重要的 3 个服务质量因素：响应时间、吞吐量和可靠性。在此基础上，本文定义了 4 种服务质量偏好，分别为响应时间、吞吐量、可靠性和综合服务质量。其中综合服务质量偏好为响应时间、吞吐量和可靠性三者的加权平均和。其中，紧急案件更加偏好于服务质量中的响应时间，民事案件更加偏好于服务质量中的吞吐量，重大案件更加偏好于服务质量中的可靠性，其他案件则会偏好于综合服务质量。在式(10)－式(13)中，服务 i 的服务质量由 QoS_i 表示。QoS_i 为一个三维行向量，其中的值分别代表响应时间、吞吐量和可靠性；R 是一个三维列向量，处理不同的案件类型时 R 也会相应地改变，R 中所有元素之和为 1。QoS_{i1} 为在处理民事案件时，服务 i 归一化后的服务质量；QoS_{i2} 为在处理紧急案件时，服务 i 归一化后的服务质量；QoS_{i3} 为在处理重大案件时，服务 i 归一化后的服务质量；QoS_{i4} 为在处理其他案件时，服务 i 归一化后的服务质量。依据案件类型，对每一个服务计算其在该案件类型下归一化后的 QoS 值，最终得到在 4 种偏好下服务的 QoS 值，计算公式如下：

$$QoS_{i1}=QoS_i * R_{CivilCase} \tag{10}$$

$$QoS_{i2}=QoS_i * R_{EmergencyCase} \tag{11}$$

$$QoS_{i3}=QoS_i * R_{importantCase} \tag{12}$$

$$QoS_{i4}=QoS_i * R_{aggregative} \tag{13}$$

4.2 服务筛选与匹配

图规划算法需要搜索遍历整个服务集合空间，在服务集合规模小的情况下，图规划算法的时间性能消耗较小。但是，微服务架构通常为大规模的服务集合，图规划算法往往不能在较短时间内返回用户所需要的服务组合，因此需要一些特定的策略来减少遍历所需要的服务空间。经研究发现，对于一次访问请求，最终生成的服务组合中的服务数量与整体的服务集合的服务数量相比是很少的。也就是说，对于某一次的访问请求，服务集合中存在很多无关的服务，这些服务对于最终服务组合的结果没有任何贡献，却大大延长了服务组合算法的计算时间。因此，本文提出了服务筛选算法对这些无关服务进行筛选。该算法在服务组合之前，将部分提供相同功能但服务质量不满足用户需求的服务进行筛选并剔除，从而减少图规划算法在组合过程中所需要的搜索空间和搜索时间。服务筛选算法以服务库中的服务作为输入，输出各偏好下的任务集合，遍历一遍服务库，对于其中的每个服务，如果有其他的服务能够提供相同的业务功能，并且具有更高的服务质量，则用它替换该服务进入任务集合。最终，算法结束时，任务集合中的服务都能够提供不同的功能，而且全都具有较高的服务质量。服务筛选算法的详细步骤如算法 1 所示。

算法 1 服务筛选

输入：服务库 W={S_1，S_2，S_3…}

输出：各个偏好下的任务集合 Ts[p]

```
1. for each Si in W
2.   if W==empty
3.     结束当前算法
4.   将当前的服务 Si 置为待选服务
5.   W=W-Si
6.   for each Sj in W
7.     if Sj==Si
8.       W=W-Sj
9.     if Sj.qos[p]>Si.qos[p]
10.    将当前的服务 Sj 置为待选服务
11.      将待选服务加入该偏好下的任务集合中
```

服务筛选算法中的 p 值代表在不同的案件偏好下筛选出的案件的集合。当 p 为 1 时，集合为偏好于处理民事案件的任务集合；当 p 为 2，集合为偏好于处理紧急案件的任务集合；当 p 为 3，集合为偏好于处理重大案件的任务集合；当 p 为 4，集合为偏好于处理其他案件即注重于服务的综合质量的任务集合。

在服务组合问题中，需要通过形式化方法得到服务与服务之间的逻辑关系。在图规划算法中，一般会先建立服务规划图，然后在服务规划图中进行遍历，并搜索满足用户需求的服务组合。但是，在微服务架构下，大规模的服务规划图会大大降低算法的空间性能。本文通过建立任务关联的集合来代替服务规划图，从而提高算法的空间效率，同时能够提高后续组合算法的灵活性。在任务匹配算法中，输入为上一步算法得到的服务候选集，其中任务集合包括了功能各异的不同服务模块，任务匹配算法对任务集合中的每个任务进行遍历，如果任务 i 的输出参数包含任务 j 的输入参数，则构建一个任务 i 到任务 j 的向量，该向量表示任务 i 与任务 j 之间包含着语义逻辑关系。

算法 2 任务匹配

输入：任务集合 Ts[p]

输出：任务关联集合 Edges[p]

```
1. for each Ti in Ts[p]
2.   for each Tj in Ts[p]
3.     if Ti.output⊇ Tkj.input
4.       任务关联集合中添加(Ti，Tj，TotallyMatch or PartlyMatch)
```

由于服务筛选算法中已经剔除了所有冗余的服务，在此时的任务集合中，每个任务节点只有一个服务，而且每个服务都能够提供不一样的功能和工作。在服务匹配算法中，当任务 i 的输出集合包含任务 j 的输入集合，即 $T_i.output$ 包含 $T_j.input$，则可以认为任务 i 与任务 j 之间存在逻辑关系，即任务 i 为任务 j 的前置服务。其中，当任务 i 的输出集合等于任务 j 的输入集合，则可以认为任务 i 与任务 j 为完全匹配，服务结构为链式或者分支结构；当任务 i 的输出集合真包含于任务 j 的输入集合，则可以认为任务 i 与任务 j 为部分匹配，即存在并行结构。传统的服务机制仅能够组合链式结构的服务，而任务匹配算法针对检察院的实际服务场景，可以支持包含分支结构的服务扩展，使得服务组合的方式和内容更加丰富。

4.3 服务组合

在服务组合算法中，输入是任务关联集合和用户请求，输出为封装后的工作流。首先，起始状态为用户请求的输入，对于每一个任务关联，如果其前置任务满足当前状态，则将该任务关联加入工作流，并将后继任务作为新的输入重新在任务关联集合中寻找下一个任务关联，直到达到用户请求的输出要求，则该工作流即为最终用户需求所要封装的服务组合。

并行结构在服务组合问题中存在一定程度的特殊性，在响应时间的计算上，并行结构仅计算响应时间最长的分支，并将其作为分支的服务质量。同样，在吞吐量的计算上，并行结构仅计算吞吐量最少的分支并将其作为该分支的服务质量。因此，优化服务组合其他分支的服务质量就会失去意义。传统的图规划算法仅能在每一层中选取最优的服务进行组合，忽视了对并行结构的处理，不仅无法优化整体服务组合，还可能因为服务组合优化而降低其他偏好下整体服务组合的服务质量。本文设计的案件服务组合封装能够判断出需要优化的服务质量的分支和其他分支，对于需要优化服务质量的分支，会对其中的所有服务进行优化，而对于其他分支则会进行综合服务质量的优化，在不影响服务组合的整体服务质量的情况下能够使得其他服务质量得到提高。具体的算法步骤如算法 3 所示。另外，本文算法在处理重大案件（即可靠性偏好）和其他案件（即综合偏好）时，会直接借鉴传统图规划的思想，这是因为整体服务结构的可靠性是通过累乘的方式获得，不存在对并行结构的处理；同样地，计算综合偏好时也是直接取综合偏好下最优的服务即可，也不存在对并行结构的处理。

算法 3 的案件服务组合封装首先根据案件类型动态选择不同的任务关联集合，其中，EdgeSet[1]表示偏好于处理民事案件的集合；EdgeSet[2]表示偏好于处理紧急案件的集合；EdgeSet[3]表示偏好于处理重大案件的集合；EdgeSet[4]表示偏好于处理其他案件的集合。算法 3 会判断服务生成的图中是否存在并行结构，如果存在并行结构，则对其中的每个分支分别进行该偏好下的优化，然后判断不需要进行优化的分支，并对这些分支进行服务综合服务质量上的优化。

算法 3 案件服务组合封装

输入：任务关联集合 EdgeSet，用户请求 Request，案件类型 Type

输出：工作流 W

```
1.  通过服务请求，在服务偏好下初始化工作流的初始任务节点 Start 与终止任务节点 End
2.  将初始节点置入空队列 Queue 中
3.  while Queue!=None
4.  从队列中取出一个任务节点作为待处理任务节点
5.    Case Type=CivilCase
6.  如果该任务节点与后继任务节点不完全匹配
7.  在吞吐量偏好下的任务关联集合中进行任务关联的选择
8.  对吞吐量高的分支，在全局偏好下的任务关联集合中进行重新选择
9.  如果该任务节点与后继任务节点完全匹配
10. 在吞吐量偏好下的任务关联集合中进行任务关联的选择
11.   Case Type=EmergencyCase
12. 如果该任务节点与后继任务节点不完全匹配
13. 在响应时间集合偏好下的任务关联集合中进行任务关联的选择
14. 对于响应时间短的分支，在全局偏好下的任务关联集合中进行重新选择
```

15. 如果该任务节点与后继任务节点完全匹配
16. 在响应时间偏好下的集合中进行任务关联的选择
17. Case Type=ImportantCase
18. 在可靠性偏好下的任务关联集合中进行任务关联的选择
19. Case Type=AggregativeIndicator
20. 在综合服务质量偏好下的任务关联集合中进行任务关联的选择
21. 将边集合中的后继任务节点置入队列
22. 记录 W
23. Return W

服务组合封装算法的目标是将一系列服务以及子工作流组合为一条符合用户需求的工作流;算法的输入是用户请求、任务关联集合和用户请求;算法的输出是一条封装后的服务工作流。该算法通过队列实现工作流的层次搜索,同时能够根据案件类型和用户请求智能地提供不同的组合方案。最终封装的服务组合方案将会是一个 Directed Acyclic Graph (DAG),而不是传统的封装为链的形式。在该算法产生的服务组合方案中,每个方案都由一个工作流表示,在算法产生的工作流中,每一个服务都代表服务注册库中的一个可用服务。其中,每个工作流都由 Conductor 的形式语言表达。最后,系统根据服务组合情形生成 Conductor 的流程定义原始文件,完成微服务架构下的服务封装。

5 实验结果与分析

5.1 实验 1:算法性能实验

本文实验首先使用的数据集是 WsDream 数据集,其统计了真实世界中 339 个用户调用 5825 个服务所得到的 QoS 结果,QoS 包括响应时间和吞吐量两个属性。本文算法使用 python3.5 实现,实验环境为 64 位 win10,CPU 为 i7,内存为 16GB。

对比实验算法为 WSC-MDP[32] (Adaptive Web Service Composition Based on Reinforcement Learning)算法。WSC-MDP 是一种典型的基于人工智能规划进行服务组合优化的算法,其借鉴强化学习中的马尔可夫决策过程进行组合优化,通过对奖励函数进行多次训练,最终得到不同状态下选择不同服务的奖励函数,通过奖励函数实现服务组合,提高表达和推广能力。

本节实验由多个任务节点构成,除了终止任务节点和起始任务节点以外,每个任务节点都有多个后继任务节点,任务节点到后继任务节点构成了一条转移路径,每条转移路径上至少有一个可能的待选服务,整个执行过程中需要选择多次候选服务,每个任务节点可以选择多个后继任务节点。在具体的实验中,可以手动设置服务的选择次数、候选服务个数以及后续的状态数,随机生成不同的工作流,用以对算法的性能进行充分验证。

图 2 给出了服务库规模和算法运行时间的实验结果,其中横轴是服务库的服务规模,纵轴是算法完成请求所消耗的时间(本文对运行时间取对数进行绘制)。红色曲线是本文提出的算法 IGMA,绿色曲线是典型的强化学习的 WSC-MDP 算法。从对比实验的结果可以看出,随着服务库规模的不断扩大,本文所提算法的运行时间基本保持稳定,即使服务库的规模到达了 5000 个,算法的运行时间依然保持在 10μs 以内。这是因为在实验 1 中,服务组合的方法成为服务选择的方法,即在每一次服务选择中,本文方法都会选择最佳的服务作为工作流的服务。另外,强化学习的方法所消耗的时间随着服务库规模的增大而显著增长,当服务库规模达到 5000 个时,运算时间甚至达到了 4.3s。这是因为强化学习需要对服务数据进行上千次的训练。因此,从时间效率角度考虑,本文所提出的服务组合封装算法完全适用于大规模的服务组合。

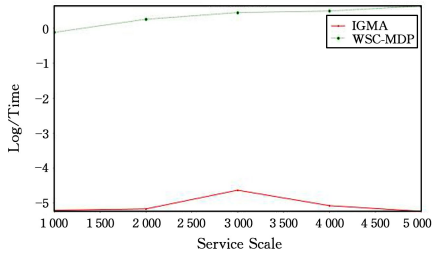

图 2 服务规模-运行时间比较图(电子版为彩色)

Fig. 2 Service scale-running time comparison diagram

图 3 给出了服务库规模和最终工作流的服务质量的实验结果,其中横轴是服务库的规模,纵轴是工作流整体结构的服务质量。工作流中每一个服务的服务质量为归一化后的响应时间和归一化后的吞吐量的加权平均。同样,红色曲线代表 IGMA,绿色曲线代表 WSC-MDP。对于同一个服务库规模下的工作流,工作流的整体质量越高,即可证明服务组合算法越好。从实验中可以看到,在所有服务场景下,对于不同的服务库规模,本文算法生产的服务工作流的服务质量都略高于传统强化学习方法组合的服务工作流。

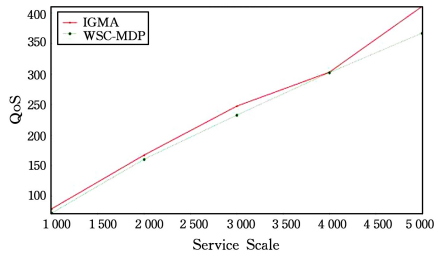

图 3 服务规模-服务质量比较图

Fig. 3 Service scale-QoS comparison diagram

5.2 实验 2:仿真数据实验

本节实验是基于 QoS 的组合方案优化,实验中除了使用 WS-Dream 数据集之外,还使用了 QWS 数据集中的部分数据。QWS 数据集收集了 5000 个 Web 服务,并对此数据集执行了各种测量,提供了目前 Web 上存在的 365 个真实 Web 服务实现的子集,使用 Web 服务爬虫引擎(WSCE)收集服务,持续监控特定的服务质量,包括响应时间、吞吐量和可靠性。由于强化学习算法通常只能够得到链式的服务组合,本节实验选取传统图规划算法作为对比算法。我们针对检察业务机制进行了实验,通过本文算法和 GP 算法[39]分别生成服务组合,并计算服务组合的服务质量。本文设计的实验中的服务组合结构主要分为 3 种,分别是顺序、分支和并行。

我们分别使用本文算法与 GP 算法针对 10 个随机生成的请求进行实验,实验的请求中包含用户的输入、输出以及服

务质量偏好。其中，请求 1—请求 5 为处理紧急案件；请求 6—请求 10 为处理民事案件。我们对算法生成的服务组合计算其服务质量，并将算法得到的服务质量与综合质量最优的服务组合的综合质量相除得到相对服务质量。实验结果如图 4 所示，可以看出，IGMA 生成的服务组合的相对服务质量都略高于 GP 算法。这是因为本文算法在处理分支的服务组合问题时会进行进一步的优化，在处理吞吐量偏好与响应时间偏好的请求时会获得更好的服务组合结果。

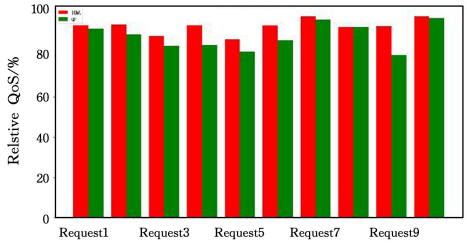

图 4 相对服务质量图

Fig. 4 Relative QoS diagram

表 1 为 IGMA 和 GP 算法的实验结果对比，表 1 中第 2 列和第 3 列的参数依次表示服务组合的综合质量、服务组合的可靠性、服务组合的响应时间、服务组合的吞吐量和相对服务质量；第 4 列的参数依次表示服务组合的综合质量、服务组合的可靠性、服务组合的响应时间、服务组合的吞吐量。

表 1 实验结果对比

Table 1 Comparison of experimental results

	IGMA	GP	综合最优质量
请求 1	2.2/39.56%/68.7/3.8/95.7%	2.16/14.24%/68.7/3.8/93.9%	2.3/83.42%/79.2/1.2
请求 2	3.23/25.07%/26.5/3.8/95.8%	3.07/6.87%/24.63/0.5/91.1%	3.37/52.85%/26.5/1.2
请求 3	2.43/15.92%/49.2/0.5/90%	2.3/7.55%/49.2/0.5/85.2%	2.7/44.09%/58.7/1.2
请求 4	3.04/19.73%/24.6/3.8/95.6%	2.73/8.29%/24.6/3.8/85.8%	3.18/41.62%/25.86/1.2
请求 5	2.01/7.65%/49.3/0.5/88.5%	1.87/3.63%/49.3/0.5/82.4%	2.27/49.16%/87.5/1.2
请求 6	2.47/34.15%/77.9/7.5/95.4%	2.28/29.45%/77.9/7.5/88%	2.59/56.72%/87.5/4.4
请求 7	2.7/43.37%/77.9/7.5/100%	2.66/15.61%/77.9/7.5/98.5%	2.7/72.02%/87.5/4.4
请求 8	3.48/52.59%/31.6/16.2/94.6%	3.48/52.59%/31.6/16.2/94.6%	3.68/60.98%/26.5/8.8
请求 9	2.16/29.6%/77.9/1.2/95.15%	1.83/28.18%/77.9/7.5/80.61%	2.27/49.16%/87.5/1.2
请求 10	3.2/55.28%/49.2/15.2/100%	3.17/19.9%/49.2/15.2/99.06%	3.2/64.6%/58.7/1.2

从表 1 可以看出，在偏好于所要求的服务质量的情况下，相比传统的组合方法，本文方法能够进一步加强服务组合在其他偏好下的服务质量。例如，在处理请求 1、请求 3、请求 4 和请求 5 时，在服务组合响应时间达到最短的同时，服务组合的可靠性和吞吐量也有了明显的提升。在处理请求 6、请求 7 和请求 10 时，在服务组合的吞吐量达到最大的同时，服务组合的可靠性和响应时间也有明显的提升。虽然，在请求 2 和请求 9 中，本文得到的服务组合在响应时间和吞吐量上并没有达到最优，但是总体的服务质量却有明显的提升。另外，在处理请求 8 时，由于服务结构中各个分支的服务指标具有特殊性，本文算法与传统图规划算法得到了相同的性能。因此，通过本实验能够验证本文算法能够显著提升服务质量。

结束语 本文提出了一种基于微服务的服务封装方法，以解决微服务架构下的检察业务服务封装问题，同时，在封装方法中，设计了一种改进的图规划方法（IGMA）进行服务组合。通过实验验证，IGMA 在用户偏好和算法运行时间上都有良好的表现。在下一步的研究工作中，将解决微服务架构下服务组合中的服务修复问题。

参 考 文 献

[1] ZHANG X Z, LYU T Y, ZHANG B. Modeling of complex service collaborative network based on service interaction behavior [J]. Journal of Software, 2016, 27(2): 231-246.

[2] XIN Y Y, NIU J, XIE Z J, et al. Overview of microservice architecture implementation framework [J]. Computer Engineering and Application, 2018, 54(19): 16-23.

[3] DU M, LI F F, ZHENG G N, et al. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning [C] // ACM Conference on Computer and Communications Security (CCS). 2017: 1285-1298.

[4] DELAC G, SILIC M, SRBLJIC S. A Reliability Improvement Method for SOA-Based Applications [J]. IEEE Transactions on Dependable and Secure Computing (TDSC), 2015, 12(2): 136-149.

[5] ZHANG W C, SUN H L, LIU X D, et al. Temporal QoS-Aware Web Service Recommendation via Non-negative Tensor Factorization [C] // International World Wide Web Conference (WWW). 2014: 585-595.

[6] SILIC M, DELAC G, SRBLJIC S. Prediction of Atomic Web Services Reliability for QoS-Aware Recommendation [J]. IEEE Transactions on Services Computing, 2015, 8(3): 425-438.

[7] LO W, YIN J, DENG S, et al. An Extended Matrix Factorization Approach for QoS Prediction in Service Selection [C] // IEEE Ninth International Conference on Services Computing. IEEE, 2012: 162-169.

[8] ZHENG Z, ZHANG Y, LYU M R. Investigating QoS of Real-World Web Services [J]. IEEE Transactions on Services Computing, 2014, 7(1): 32-39.

[9] ZHENG Z, ZHANG Y, LYU M R. Distributed QoS Evaluation for Real-World Web Services [C] // 2010 IEEE International Conference on Web Services (ICWS). IEEE, 2010: 83-90.

[10] YANG Y, TANG S, XU Y, et al. An Approach to QoS-aware Service Selection in Dynamic Web Service Composition [C] // Third International Conference on Networking and Services. ICNS, IEEE, 2007: 19-25.

[11] FANJIANG Y Y, SYU Y, MA S P, et al. An Overview and Classification of Service Description Approaches in Automated Service Composition Research [J]. IEEE Transactions on Services Computing, 2017, 10(2): 176-189.

[12] VARDHAN A V, BASHA M S S, DHAVACHELVAN P. An Overview of Web Services Composition Approaches [J]. International Journal of Computer Applications, 2011, 29(8): 10-15.

[13] CONSTANTINESCU I, FALTINGS B, BINDER W. Large

scale, type-compatible service composition[C]// IEEE International Conference on Web Services. IEEE Computer Society, 2004:5-6.

[14] THAKKER D, OSMAN T, AL-DABASS D. Knowledge-Intensive Semantic Web Services Composition[C]// Tenth International Conference on Computer Modeling & Simulation. IEEE, 2008:673-678.

[15] AIELLO M, KHOURY E E, LAZOVIK A, et al. Optimal QoS-Aware Web Service Composition[C]// Joint IEEE Conference on E-Commerce Technology and IEEE Conference on Enterprise Computing, E-Commerce and E-Services IEEE Computer. 2009: 491-494.

[16] ALRIFAI M, RISSE T, NEJDL W. A hybrid approach for efficient Web service composition with end-to-end QoS constraints [J]. ACM Transactions on the Web, 2012, 6(2):1-31.

[17] YAN Y, XU B, GU Z, et al. A QoS-Driven Approach for Semantic Service Composition[C]// IEEE Conference on Commerce & Enterprise Computing. IEEE, 2009:523-526.

[18] ZHENG Z, ZHANG Y, LYU M R. Investigating QoS of Real-World Web Services[J]. IEEE Transactions on Services Computing, 2014, 7(1):32-39.

[19] ZHENG Z, ZHANG Y, LYU M R. Distributed QoS Evaluation for Real-World Web Services[C]// 2010 IEEE International Conference on Web Services (ICWS). IEEE, 2010:83-90.

[20] CONSTANTINESCU I, FALTINGS B, BINDER W. Large scale, type-compatible service composition[C]// IEEE International Conference on Web Services. IEEE Computer Society, 2004:506-513.

[21] ZHANG P Y, HUANG B, SUN Y M. Service semantic matching mechanism based on service combination[J]. Journal of University of Electronic Science and Technology, 2008, 37(6): 917-921.

[22] LI S Z, YANG G, YANG S X. A semantic Web service matching algorithm based on ontology concept similarity[J]. Microcomputer and Applications, 2009, 28(15):57-60.

[23] ZHANG R L. Research on combination method based on Web service group model[J]. Science and Technology Information, 2012(30):285-285.

[24] WU B, WU J, DENG S, et al. Automatic Composition of Semantic Web Services An Enhanced State Space Search Approach [C]// International Conference on Service Sciences. IEEE Computer Society, 2010, 13/14:226-230.

[25] CORBIN H. Changing maternity service in a changing world [J]. Public Health Nursing, 1950, 42(8):427.

[26] KAELBLING L P, LITTMAN M L, MOORE A W. Reinforcement Learning: A Survey[J]. Artificial Intelligence Research, 1996, 4(1):237-285.

[27] GAO A, YANG D, TANG S, et al. Web service composition using markov decision processes[C]// Processing of the 6th International Conference on Advances in Web-Age Information Management. 2005:308-319.

[28] OH S C, LEE D, KUMARA S R. Effective Web Service Composition in Diverse and Large-Scale Service Networks[J]. IEEE Transactions on Services Computing, 2008, 1(1):15-32.

[29] REN L, WANG W, XU H. A Reinforcement Learning Method for Constraint-Satisfied Services Composition[J]. IEEE Transactions on Services Computing, 1939, 13(5):786-800.

[30] WANG H, ZHOU X, ZHOU X, et al. Adaptive Service Composition Based on Reinforcement Learning[C]// ICSOC. LNCS, 2010:92-107.

[31] WANG H, QIN W, XIN C, et al. Adaptive and Dynamic Service Composition via Multi-agent Reinforcement Learning[C]// IEEE International Conference on Web Services. 2014:447-454.

[32] LI J J. Adaptive service combination combining QoS prediction and multi-agent reinforcement learning[D]. Nanjiang: Southeast University, 2018.

[33] WANG H, CHEN X, WU Q, et al. Integrating On-policy Reinforcement Learning with Multi-agent Techniques for Adaptive Service Composition[M]// Service-Oriented Computing. 2014: 154-168.

[34] ABOUHEAF M, GUEAIEB W. Reinforcement Learning Solution with Costate Approximation for a Flexible Wing Aircraft [C]// 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA). 2018:1-6.

[35] ARDAGNA D, PERNICI B. Adaptive Service Composition in Flexible Processes[J]. IEEE Transactions on Software Engineering, 2007, 33(6):369-384.

[36] LIN S Y, LIN G T, CHAO K M, et al. A Cost-Effective Planning Graph Approach for Large-Scale Web Service Composition [J]. Mathematical Problems in Engineering, 2012:1-21.

[37] HUA Z, YAN F, HUI G. A Web Service Composition Algorithm Based on Dependency Graph[J]. Lecture Notes in Electrical Engineering, 2012, 113:1511-1518.

[38] FAN G D. Web service combination method based on FAHP and scheme map fusion[J]. Computer Science, 2020, 47(1): 270-275.

LU Yi-fan, born in 1995, postgraduate. His main research interests include service computing, service composition, service encapsulation and microservice.

YAN Chun-gang, born in 1963, Ph. D, professor, Ph. D supervisor. Her main research interests include collaboration and service computing, Petri net modeling and analysis.