

一种 Multi-Agent 的云联盟服务组合模型

尚广安, 蒋哲远

(合肥工业大学 计算机与信息学院, 安徽 合肥 230009)

摘 要: 提出了一种利用 Multi-Agent 的交互协作以实现云联盟平台中的云服务自动优化组合模型. 继而基于该模型设计了一种启发式云服务组合算法, 该算法首先根据用户的个性化请求对服务库中的云服务进行预处理, 筛选出合适的云服务并对其进行代价遍历. 最后, 以服务相似度和组合 QoS 对选取的云服务进行回溯优化组合, 进而获取最优的组合方案. 仿真结果表明该算法在组合效率和成功率方面较同类算法有较大提升.

关键词: 云联盟; Multi-Agent; 服务组合; 启发式算法

中图分类号: TP393.09

文献标识码: A

文章编号: 1000-7180(2014)09-0001-06

Cloud Federation Service Composition Model Based on Multi-Agent

SHANG Guang-an, JIANG Zhe-yuan

(School of Computer Science and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: A cloud federation service composition model based on Multi-Agent is proposed to compose optimal services in multi-Cloud platform. Then on the basis, a new heuristic algorithm is designed. According to the input parameters of request, the cloud service in the service library is preprocessed. And then the algorithm traverses the appropriate cloud service with heuristic function. Finally, the algorithm makes full use of service similarity and QoS to choose concrete service and program optimized in the process of regression. Compared with other similar algorithms, the simulation experiments show our algorithm can provide higher efficiency and shorter solution path for the requests.

Key words: cloud federation; Multi-Agent; service composition; heuristic algorithm

1 引言

云服务作为一种面向应用、资源共享、按需提供的网络化服务, 具有在线迁移、动态配置、灵活组合等特点. 当面对日益复杂的企业需求, 单一的云服务功能有限, 组合和集成已有的云服务成为必然选择. 然而由于平台的异构性, 限制了服务的可移植和集成. 因此, 业界和学术界提出了开放云计算联盟^[1] (open cloud computing federation), 即一种整合分布式不同服务提供商提供的云服务, 共同为用户提供服务的机制. 由于云联盟中的服务资源数量众多因而快速自动地实现服务组合还是存在不少的困难.

另一方面, 以 Web 服务为代表的云服务具有动态变化、安全难控、可定制和扩展等特征, 已有的服务组合方法由于基于固定流程难以直接应用在云服务组合中. Agent^[2] 是存在于某一环境中的具有一定智能的软件实体, 具有自治性、社会性、智能性、学习能力等特征. Multi-Agent 系统是由多个智能 Agent 所组成的系统, 将 Multi-Agent 技术引入到云服务组合^[3] 中, 利用 Agent 作为云服务主体的代表, 一方面能够实现服务组合过程中最合适的服务选取, 另一方面可加强组合系统的交互能力.

目前已有许多学者针对服务组合问题进行了研究^[4-5]. 其中文献^[5]将 Agent 引入组合云服务, 建立了基于 Agent 的组合云服务模型. 虽然上述研究

收稿日期: 2013-11-27; 修回日期: 2014-01-18

基金项目: 国家自然科学基金重点资助项目(60633060)

在一定程度上完成云服务组合问题的求解,但还存在改进的空间。

针对云联盟中如何实现云服务的自动组合问题,本文采用 Multi-Agent 主动获取用户的个性化需求,并选择最优的服务资源,生成组合方案以实现自动组合。

2 一种 Multi-Agent 的云联盟服务组合模型

2.1 基于 Multi-Agent 的云联盟服务组合模型结构

云联盟服务组合模型主要是借助 Multi-Agent 进行构建,即在每个云服务节点上部署 Agent. 一个 Agent 管理着特定的云服务,拥有该云服务所具有的详细信息,按照不同的分工担负不同的任务. 本文借鉴了文献[6]的 Web 服务组合框架,提出了基于 Multi-Agent 的云联盟服务组合模型结构. 该结构针对服务组合需求,利用 agent 的智能性和交互特点,实现云服务的自动组合. 如图 1 所示,主要包含 4 种 Agent 即用户 Agent(Consumer Agent, CA)、服务提供 Agent(Service Provider Agent, SPA)、组合管理 Agent(Composition Management Agent, CMA)和资源 Agent(Resource Agent, RA)和 3 种基本库即服务本体库、服务组合库和服务代理库。

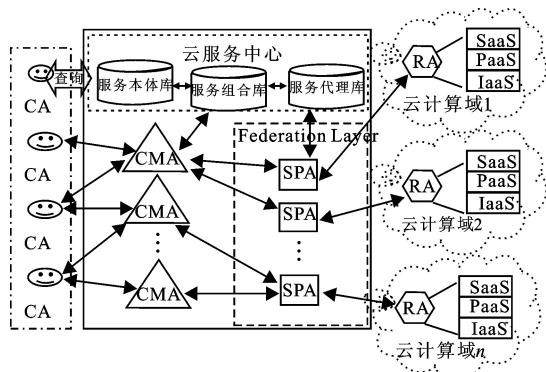


图 1 基于 Agent 的云联盟服务组合模型结构

(1) 4 种 Agent: CA 传递和过滤用户的服务请求,生成服务组合任务,和 CMA 进行交互,并根据用户的服务请求查询和更新服务本体库. CMA 接受和分解 CA 分配的组合任务,按照一定的规则生成组合方案. SPA 负责管理云联盟中服务提供者的资源即云平台及云应用,与联盟成员进行通信,并按照组合方案筛选最优的服务. RA 管理某个云控制域的云资源,拥有软件即服务(Software as a Service, SaaS)、平台即服务(Platform as a Service,

PaaS)和基础设施即服务(Infrastructure as a Service, IaaS)中的一种或几种服务。

(2) 3 种基本库:服务本体库存储已注册的云服务信息和相关领域知识、原子服务等. 服务组合库存储成功解决用户需求的服务组合方案的集合,包括用户服务需求和所需服务集合. 服务代理库记录模型中 Agent 的状态信息包括正常、维护和可用性等,还有服务信息. 各个基本库采用 OWL-S 语言进行描述。

2.2 组合管理 Agent

Agent 具有智能性和自主性,可作为服务的实体独立地在云联盟的网络环境中提供各种服务. 多个 Agent 则以一种松散耦合的网状结构出现,相互之间通过结构化的 ACL (Agent Communication Language) 进行交互、协作,共同完成某一任务. CMA 是组合模型中实现服务组合的关键部分,负责任务分解和生成组合方案等任务. 图 2 是 CMA 的内部构造图. CMA 通过通信模块与外界环境、其他 Agent 进行通信,任务分解模块负责划分组合任务和生成组合方案,知识库中存放有专家提供和历史的组合方案,若方案存在则直接取出;如果不存在则进行生成. 将组合方案的原子请求封装成原子服务,利用服务匹配模块对候选云服务进行服务参数和能力的匹配;如果匹配则由组合执行方案模块按照组合方案调用相应的 Agent. 能力管理模块和状态监控模块负责 CMA 服务能力的管理和工作状态的监控,并将能力和状态反馈至对应的部分。

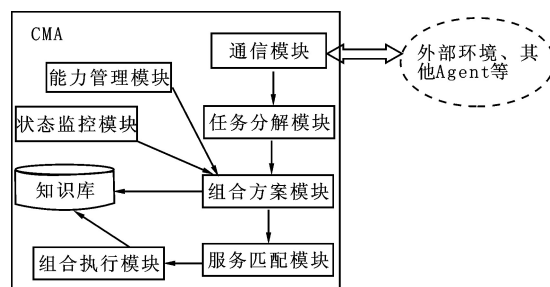


图 2 CMA 的内部构造图

2.3 云联盟服务组合流程

组合过程中基于模型实现的业务逻辑、各个基本库与 Agent 之间的交互活动如图 3 所示。

(1) 当用户输入服务请求后,CA 根据服务请求生成输入集合、输出集合和 QoS 集合,在服务本体库中查找与输出集合相匹配的抽象服务. 若查询成功则生成组合任务交给 CMA 执行,否则返回服务未注册。

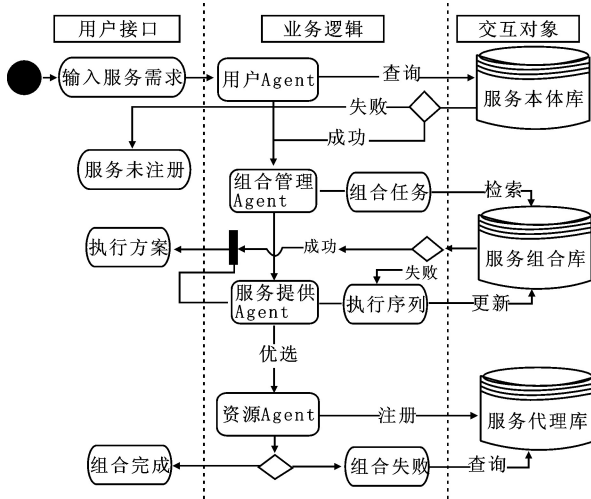


图3 云联盟服务组合活动图

(2) CMA 接受和分解 CA 的组合任务. 先从服务组合库中检索匹配的服务组合方案, 若匹配成功则返回组合执行方案, 若没有则生成组合执行序列并更新服务组合库.

(3) SPA 根据 CMA 生成的组合方案, 在联盟环境下每个云平台中优选符合参数要求的候选服务, 执行调用命令.

(4) RA 是组合服务的组成单元, 对应封装了云联盟中的三种类型服务, 按照 SPA 的选择组成组合服务, 将结果返回给用户.

3 云服务组合启发式算法

3.1 问题描述及定义

云联盟中存在海量数目的服务, 由于单一的云服务难以完成复杂的企业需求. 只有通过服务集合中筛选云服务, 将多种异构云服务组合成具有新功能的云服务才能满足需求. 如图4所示的一个旅行计划的组合服务实例, 游客只需登录目的城市的门户网站, 在门户网站上即可完成旅游计划. 由于单一的云服务功能有限, 实现此项业务必须使用多种云服务. 网站在购买或租赁相应的云服务后, 通过组合已有的云服务可以实现这项业务. 图中显示了实例及对应的 Agent 关系, 利用每个 Agent 抽象地封装成具有某种功能属性的云服务, 通过不同类型的 Agent 协同机制实现服务组合.

通常服务组合问题是根据用户请求不断调用服务实现目标的过程. 图4根据请求的输入参数, 调用满足条件的服务组合方案实现最终输出目标. 由于 Agent 作为云服务的载体, 可以代表服务主体参与服务组合. 因此服务组合问题可以看作是寻找代价

最小和质量最优的 Agent 集合序列.

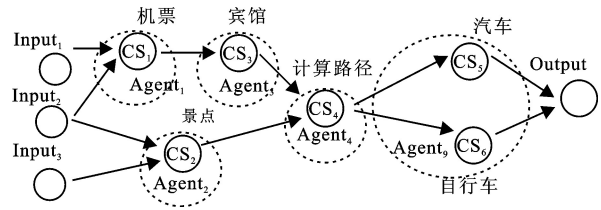


图4 云服务组合实例及对应的 Agent 关系

定义1 (Agent) 云联盟环境中的 Agent 定义为六元组模型: $agent = (ID, In, Out, Relation, state, capability)$, 式中 ID 标识云联盟中 Agent 的唯一性, 由所在域、所在节点和本地序号共同组成. In 代表 Agent 所需的输入参数的集合, Out 代表 Agent 的输出参数的集合. $Relation$ 标识节点 Agent 之间的关系, $state$ 标识 Agent 当前的工作状态 (如“忙”或“闲”还是“维护中”), $capability$ 描述 Agent 的服务能力, 主要指节点所提供的硬件和软件云服务.

定义2 (Agent 关系, Relation) $Relation$ 是一个标识节点 Agent 之间关系的元组. 若 Agent A_i 和 A_j 之间存在 $A_i, Out \cap A_j, In = \emptyset$ 则 A_i 和 A_j 独立; 若 A_i 和 A_j 之间存在 $A_i, Out \cap A_j, In \neq \emptyset$ 则 A_i 和 A_j 依赖记为 $Dep(A_i, A_j, X)$, 其中 $X = A_i, Out \cap A_j, In$.

定义3 (云服务) 云服务可以表示为一个四元组 $cs = (id, cs^{in}, cs^{out}, qos)$. 其中 id 代表云服务的唯一标识, cs^{in} 代表服务的输入参数的集合, cs^{out} 代表服务的输出参数的集合. qos 代表云服务的非功能属性, 包括服务成本 $C(s)$ 、服务时间 $T(s)$ 、可信度 $Rep(s)$ 和可靠性 $Rel(s)$ 等^[7], 用四元组 $qos = \langle C(s), T(s), Rep(s), Rel(s) \rangle$ 表示.

定义4 (云服务组合) 云服务组合可以表示为一个四元组 $CSC = \langle S, C^{in}, C^{out}, SSet \rangle$, 式中 S 表示服务本体库, C^{in} 表示用户需求的输入参数集合, C^{out} 表示用户目标的输出参数集合, $SSet$ 表示所有注册可用的云服务集合.

通常组合执行路径优选的重要依据是组合云服务的 QoS, 它是通过计算每个节点的服务得到组合路径的整体 QoS. 组合云服务 QoS 可作为组合方案的约束条件对基本服务进行筛选以确保其质量最优. 组合服务的 QoS 与组合结构关系相关, 表1列出了组合结构及对应的 QoS 计算公式. 其中 p_i 为服务选中的概率, k 为循环次数.

表 1 组合结构及对应的 QoS 计算公式

	顺序结构	并行结构	选择结构	循环结构
服务成本 $C(s)$	$\sum_{i=1}^N C(s_i)$	$\sum_{i=1}^N C(s_i)$	$\sum_{i=1}^N (C(s_i) \times P_i)$	$k \sum_{i=1}^N C(s_i)$
服务时间 $T(s)$	$\sum_{i=1}^N T(s_i)$	$\max(T(s_i))$	$\sum_{i=1}^N (T(s_i) \times P_i)$	$k \sum_{i=1}^N T(s_i)$
可信度 $Rep(s)$	$\sum_{i=1}^N Rep(s_i)/N$	$\sum_{i=1}^N Rep(s_i)/N$	$\prod_{i=1}^N Rep(s_i) \times P_i$	$\sum_{i=1}^N Rep(s_i)/N$
可靠性 $Rel(s)$	$\prod_{i=1}^N Rel(s_i)$	$\prod_{i=1}^N Rel(s_i)$	$\sum_{i=1}^N Rep(s_i) \times P_i$	$\prod_{i=1}^N Rel(s_i)$

基于上述的定义,云服务组合方案(CCS)可看作由一系列云服务调用即服务链构成,可表示为 $CCS = (cs_1, \dots, cs_i)$. 而 Agent 是对应的服务载体, CCS 亦可以分别对应被映射成 Agent 集合 $A = (agent_1, \dots, agent_j)$, 可通过得到 Agent 集合的调用和一系列动作执行即可实现服务组合. 如图 4 中的云组合方案应为 $CCS = (cs_1, cs_2, cs_3, cs_4, cs_5, cs_6)$, 可以得到映射的 Agent 集合 $A = (agent_1, agent_2, agent_3, agent_4, agent_5)$.

3.2 HACSC 算法

文献[8-9]中 Web 服务组合方法是主要通过遍历 Web 服务库给出组合方案,这种 Web 服务组合方案效率很低. 若直接在云服务搜索中应用,算法的状态空间会随着集合 SSet 基数的增长呈指数增长. 另外,通过该算法得到的组合方案有较多冗余项,不适用于复杂多变的云联盟环境. 为此,我们采用了启发式算法来解决上述问题.

启发式算法 HACSC 从深度搜索开始,首先在预处理阶段由 CA 根据用户的服务需求生成请求服务 RS 的输入参数和输出参数集合,通过匹配算法进行海量服务筛选. 然后在遍历阶段得到 Agent 集合,实现组合路径中 Agent 的数量尽可能最少,以减少 Agent 代价的目标. 最后,通过回溯组合过程得到云服务组合方案,这是因为存在功能相似的服务导致服务能力相同的 Agent 的存在,从而以服务相似度和组合 QoS 进行回溯能使组合方案质量达到最佳.

3.2.1 预处理阶段

当用户输入服务需求后,由 CA 自动获取请求服务 RS 的输入参数集合 C^m 和输出参数集合 C^{out} . 由于本体库中存在云服务海量,不可能直接遍历服务本体库,所以通过匹配算法即基于服务能力的服务功能匹配进行海量服务筛选. 具体算法描述如下:

算法 1 预处理阶段

输入:初始集合 C^m ,服务本体库 S,服务能力 *capability*
输出:候选服务集合 CSet

CSet $\leftarrow \emptyset$; SSet $\leftarrow \emptyset$; //初始化候选服务集合和服务集合

For each cs_i in S do

SSet $\leftarrow S$; //将可用服务加入注册集合

While(SSet $\neq \emptyset$) do

For each $input_i$ in C^m do

/* 将请求服务 RS 的 C^m 中的输入参数和资源 Agent 的 *capability* 匹配 */

If(match(RS, C^m , RA, *capability*))

CSet = CSet $\cup \{cs_i\}$; //匹配成功则加入到候选服务集合

SSet = SSet $-\{cs_i\}$; //将服务从服务集合中删去

If(CSet = $\emptyset$) {

Return false; //候选集合为空,则匹配失败

}

Return CSet;

3.2.2 代价遍历

由于 Agent 及代表的服务状态多变,当组合服务执行失效时会增加额外的组合代价和其他开销. 因此需要对 Agent 集合 A 进行多次遍历得到最优 Agent 集合 An . 计算 Agent 的收益函数 $Eff_{agent} = agent_i, capability / cost_{agent}$ 作为 Agent 的评估标准,其中 $cost_{agent}$ 代表 Agent 的通信开销和提供服务的成本. 服务成本包括硬件云服务和软件云服务的成本.

算法 2 遍历搜索阶段

输入:Agent 集合 A,代价阈值 *threCost*

输出:最优 Agent 集合 An

$An \leftarrow \emptyset$; $Eff_{agent} \leftarrow 0$; //初始化最优 Agent 集合和收益函数

For each $agent_i$ in A do

//计算 Agent 的收益函数

Compute the Eff_{agent} for the $agent_i$;

While($Eff_{agent} < threCost$) do

$An = An \cup \{agent_i\}$; /* 满足阈值则加入到最优 Agent 集合 */

$A = A - \{agent_i\}$; //将 $agent_i$ 从原集合中删去

If($agent_i, relation = Dep(A_i, A_j, X)$)

If($agent_j, state = work$) //判断 Agent 的工作状态

$An = An \cup \{agent_j\}$; /* 将关联的 $agent_j$ 加入到最优 Agent 集合 */

$A = A - \{agent_j\}$; //将 $agent_j$ 从原集合中删去

if($An = \emptyset$) {

return false; //最优集合为空则遍历失败

}

Return An ;

3.2.3 回溯组合

根据上面的执行结果,以回溯组合的方式得到

服务组合方案.在海量云服务中,一个服务可能对应多个候选服务.所以存在功能相似的服务,导致服务能力相同的 Agent 的存在.当服务失效时,首先根据后继服务参数相似度的大小和服务质量来对候选服务进行排序,缩小候选服务数量以选择最优的候选服务替代失效的服务,实现自动组合服务的目标.计算两个服务相似度函数的公式为

$$\text{Sim}(cs_i, cs_j) = \alpha \text{Sim}(cs_i^{\text{in}}, cs_j^{\text{in}}) + \beta \text{Sim}(cs_i^{\text{out}}, cs_j^{\text{out}}) \quad (1)$$

式中 α, β 为相似参数,根据不同的设定满足不同用户的需求.

算法3 回溯组合阶段

输入:候选服务集合 CSet,最优 Agent 集合 An

输出:服务解决方案 CCS: $cs_1 \rightarrow cs_2 \rightarrow \dots \rightarrow cs_i$

CCS $\leftarrow \emptyset$; //初始化组合方案

For each $agent_i$ in An do

//获得 $agent_i$ 的服务信息

Get the $agent_i$.capability;

If the $agent_i$.capability has the cs_i

//计算两个服务的相似度

For the cs_j in CSet

Then Match the cs_i and cs_j with $\text{Sim}(cs_i, cs_j)$;

//对相似度进行排序,得到最大值

For eachSim do{

Sort the sim;

bestSim = argmaxSim;

}

CCS = CCS $\cup \{cs_i\}$; //满足条件则加入到组合方案*

*/

CSet = CSet - $\{cs_i\}$; //将服务从原集合中删去

For each CSS do{

//计算组合方案的 QoS 并选择最高的 QoS

Get theQoS for CSS;

bestQoS = Optimal(CSS);

}

//返回并输出方案

Return CSS and output " $\rightarrow$ " to cs_i ;

3.3 算法性能分析

阶段1的预处理产生了一个符合用户请求的候选服务集合,在生成集合的过程中,每个服务通过基于服务能力和服务功能匹配进行处理.每次处理在常数时间内完成,假设服务本体库的服务个数为 N,则该阶段的时间复杂度为 $O(N)$.

在阶段2通过代价遍历得到最优的组合云服务.假设初始集合中的参数个数为 N,则需要根据代价函数至多进行 N 次遍历.遍历过程通常需要进

行若干次迭代才能结束,假设进行了迭代次数为 M,则该阶段的时间复杂度为 $O(N * M)$.

最后在解决方案生成过程中需要处理候选服务集合和临最优 Agent 集合,假设两个集合的服务个数为 N,则最坏的情况下时间复杂度为 $O(N * N)$.然而,由于引入相似度函数将大大减少搜索空间,提高效率.

4 实验验证及分析

试验平台:硬件为 CPU Intel Core i3-2350M, 4 GB 内存和 640 GB 硬盘.开发语言为 Java 及其开发工具 Eclipse4.2,使用 Hadoop 平台构建云计算环境,采用 IKV++ Grasshopper^[10]构建系统的各个 Agent.由于缺乏相关的测试集,所以采用随机生成的 Web 服务进行仿真.实验分别选取 300, 600, 900, 1 200, 1 500, 1 800, 2 100, 2 400 个 Web 服务作为云服务库.单个请求的执行时间和路径长度选取本机上 50 个随机服务请求的平均值,并根据效率公式计算其效率值.

4.1 成功概率分析

组合方案执行的成功概率反映了算法提供的组合方案的质量.成功概率越大说明组合方案的质量越好,算法的性能越优.本文对本文算法、A*^[8]算法和 BF*^[9]算法的组合效率进行了比较,如图5所示.

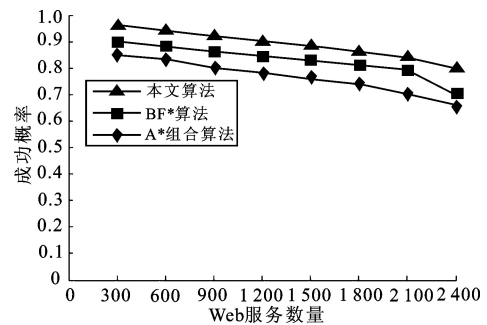


图5 成功概率分析

与 A* 算法和 BF* 算法相比,本文算法在遍历搜索阶段采用基于代价的遍历,能够为每个目标输出集合选出最优的云服务,从而增加了组合方案执行成功的概率.由图5知,在服务数量相同的前提下,根据成功概率公式 $PR_{suc} = \sum_{i=1}^n pro_{suc_i} / n$, 其中 pro_{suc_i} 为组合成功概率, n 为组合次数,计算得到本文算法成功概率最大,明显比 A* 算法和 BF* 算法好.

4.2 组合效率分析

算法从初始集合开始到找到正确组合方案所花费的时间,反映了算法的组合效率,实验结果如图 6 所示.

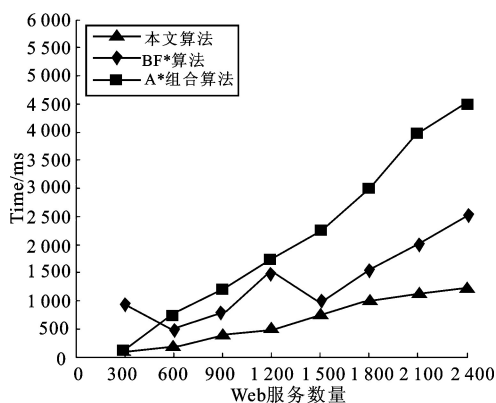


图 6 组合效率分析

由图 6 知,由于在回溯组合过程通过计算服务相似度和服务质量并进行排序,缩小了候选服务集,使得本文算法较 A * 算法和 BF * 算法在组合效率方面有较大提升. 尤其随着云服务数量增长较大时,根据组合次数和组合效率得到平均效率为 $T_{avg} =$

$\sum_{i=1}^n (time_i \times pro_{suc_i}) / PR_{suc}$, 其中 $time_i$ 为组合成功时花费的时间, PR_{suc} 为成功概率. 本文算法的执行时间要远短于 BF * 算法和 A * 算法.

5 结束语

本文针对云联盟下云服务自动组合问题,面向用户的个性化需求,引入了 Agent 来构建相关的服务节点,在此基础上提出了一种基于 Agent 的云联盟服务组合模型. 仿真实验结果表明了算法具有较好的性能.

本文的后续研究工作是对服务组合模型进行验证和改进,以便扩展应用到云联盟环境下其他云服务组合问题;进一步提升启发式云服务组合算法的性能,以提高服务组合质量.

参考文献:

- [1] Villegas D, Bobroff N. Cloud federation in a layered service model[J]. Journal of Computer and System Sciences, 2012, 78(5): 1330-1344.
- [2] Gutierrez-Garcia J O, Sim K M. Self-organizing agents for service composition in cloud computing[C]//Proc of Cloud Computing Technology and Science. New York: IEEE Computer Society, 2010: 56-66.
- [3] 陈昊, 曹健. 基于 Agent 自组织机制的服务组合[J]. 小型微型计算机系统, 2011, 10(32): 2048-2051.
- [4] 陶飞, 张霖, 郭华, 等. 云制造特征及云服务组合关键问题研究[J]. 计算机集成制造系统, 2011, 17(3): 477-486.
- [5] Gutierrez-Garcia J O, Sim K M. Agent-based cloud service composition [J]. Journal of Applied Intelligence, 2013, 38(3): 436-464.
- [6] 徐其兴, 余镇危. 一种基于移动 Agent 的服务组合模型[J]. 微电子学与计算机, 2006, 23(10): 134-137.
- [7] Cao Buqing, Li Bing, Xia Qiming. A service-oriented qos-assured and multi-agent cloud computing architecture [M]//Cloud Computing. Berlin: Springer Heidelberg, 2009: 644-649.
- [8] 蔡虹, 胡云, 李存华. 一种基于 A * 算法的 Web Services 动态组合策略[J]. 微电子学与计算机, 2011, 28(8): 197-201.
- [9] Oh S C, On B W, Larson E J, et al. BF * : web services discovery and composition as graph search problem[C]//Proc of the 2005 IEEE Int'l Conf. on e-Technology, e-Commerce and e-Service. New York: IEEE Press, 2005: 784-786.
- [10] IKV ++ Grasshopper [EB/OL]. (2010-01-16). [2014-04-10]. <http://www.Grasshopper.de/>.

作者简介:

尚广安 男, (1987-), 硕士研究生. 研究方向为云计算、服务组合.

蒋哲远 男, (1966-), 博士研究生, 硕士生导师, 副研究员. 研究方向为云计算为面向服务软件工程.