ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.12, December 2009, pp.3150−3162
doi: 10.3724/SP.J.1001.2009.03419

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

具有QoS保证的服务资源联合分配与管理*

伍之昂+, 罗军舟, 宋爱波, 曹玖新

(东南大学 计算机科学与工程学院,江苏 南京 210096)

QoS Guaranteed Service Resource Co-Allocation and Management

WU Zhi-Ang+, LUO Jun-Zhou, SONG Ai-Bo, CAO Jiu-Xin

(School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

+ Corresponding author: E-mail: zawu@seu.edu.cn

Wu ZA, Luo JZ, Song AB, Cao JX. QoS guaranteed service resource co-allocation and management. *Journal of Software*, 2009,20(12):3150−3162. http://www.jos.org.cn/1000-9825/3419.htm

Abstract: A QoS guaranteed resource management system architecture is proposed. The theoretical analysis demonstrates that this system architecture can well adapt to dynamic behavior of resources; Secondly, service resource co-allocation problem is formulated as MMKP (multiple-choice multiple-dimension knapsack problem) and an exact algorithm RA_BBLP and a heuristic algorithm RA_MHEU are both presented. Experimental results show that RA_BBLP which ensures the optimal solution can be used as reference of other algorithms. However, RA_MHEU can obtain near-optimal solution at high speed, which is an ideal service resource co-allocation algorithm.

Key words: service-oriented computing; resource management; quality of service (QoS); MMKP; RA_BBLP; RA_MHEU

摘 要: 提出一种具有 QoS 保证的资源管理系统架构,理论分析表明,该系统架构能够较好地适应资源的动态性;其次将服务资源联合分配问题归化为 MMKP(multiple-choice multiple-dimension knapsack problem)问题,并提出两种资源联合分配算法:最优解算法 RA_BBLP 和启发式算法 RA_MHEU,实验结果表明,RA_BBLP 保证求得最优解,可作为其他算法的参照系,而 RA_MHEU 收敛速度极快且所求出的解集接近最优,是一种理想的资源联合分配算法.

关键词: 服务计算;资源管理;服务质量; MMKP;RA_BBLP; RA_MHEU

中图法分类号: TP393 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant Nos.90604004, 60773103 (国家自然科学基金); the National Basic Research Program of China under Grant No.2010CB328104 (国家重点基础研究发展计划(973)); the Research Foundation for the Doctoral Program of the Ministry of Education of China under Grant No.200802860031 (国家教育部高等学校博士点学科专项科研基金); the Jiangsu Provincial Natural Science Foundation of China under Grant Nos.BK2007708, BK2008030 (江苏省自然科学基金); the Key Laboratory of Network and Information Security of Jiangsu Province of China under Grant No.BM2003201 (江苏省网络与信息安全重点实验室资助项目); the Key Laboratory of Computer Network and Information Integration (Southeast University), the Ministry of Education of China under Grant No.93K-9 (计算机网络和信息集成教育部重点实验室(东南大学))

Received 2008-05-05; Accepted 2008-07-09

服务计算作为一种新型的分布式计算模式,越来越受到国内外学术界和工业界的关注.服务计算基于面向服务架构(service oriented architecture,简称SOA),旨在通过服务资源的抽象描述、组织、管理以及服务的动态组合,支持广域开放环境下的应用协同汇聚以实现灵活、高效的资源共享[1].服务是服务计算中最基本的元素,它独立于系统平台,面向高层应用提供一组功能性操作,而使用户不必过多地关注其具体实现技术.

从功能模块来说,服务计算可分为应用层、服务实例层和系统资源层,主要解决 3 个方面的问题[2]:业务流程聚合和管理、服务选择以及服务资源管理.业务流程聚合和管理旨在根据业务流程控制逻辑和应用逻辑在用户层面上提供业务处理流程的建模技术及其服务工作流的设计方法,它产生由多个服务结点组成的工作流,各服务结点仅包含功能需求的描述,而无须指定服务实例.服务选择在众多具有相同功能的大量服务实例中选择一组服务实例形成组合服务,并使得组合服务在满足业务流程功能的基础上,尽量具有良好的服务质量(quality of service,简称QoS).服务资源管理处于服务计算模型的最底层,作为服务实例运行的支撑系统,它协同分配服务实例运行时所需的资源,确保服务实例的成功运行,并保证服务实例向组合服务所承诺的QoS水平.

国内外许多学者在业务流程聚合和管理及服务选择两方面做了大量研究工作[3−5],本文的研究集中于服务资源管理方面.服务选择方面的研究通常假设候选服务在运行过程中,系统资源层能够保证服务实例所承诺的QoS.然而,服务计算面向的是大规模开放环境,资源呈现出异构性、分布性、动态性和隶属于多管理域等特点,这些特点使得在系统资源层如何保证对上层承诺的QoS这一问题面临着巨大的挑战.本文通过对系统资源层的资源联合分配和管理问题的研究实现对服务实例承诺的QoS作出保证.首先,提出一种具有QoS保证的资源管理系统架构,该架构能够较好地适应服务资源的动态性.其次,将服务资源联合分配问题归化为MMKP问题(multiple-choice multiple-dimension knapsack problem),然后提出两种资源联合分配算法,第 1 种算法RA_BBLP在保证服务请求QoS的前提下,使用户满意度达到最优,但是由于其时间复杂度为指数级,不适用于问题规模较大的情况.因此,本文又提出第 2 种算法RA_MHEU,旨在多项式时间复杂度内得出用户满意度的近最优解.两种算法的性能比较说明,RA_MHEU算法是一种理想的资源联合分配算法,而RA_BBLP算法能够保证求得最优解,可以作为其他算法的参照系.

本文第 1 节介绍相关工作的研究现状.第 2 节提出具有 QoS 保证的资源管理系统架构(guaranteed service resource management system,简称 QG-SRMS QoS).第 3 节研究 QG-SRMS 中的资源联合分配算法,首先将问题归化为 MMKP 问题,然后再给出两种算法.第 4 节结合虚拟资源容器的理论分析和对两种算法的性能分析验证 QG-SRMS 的总体性能.第 5 节总结全文并展望下一步工作.

1 研究现状

服务计算起源于分布式计算,多年来国内外在分布式计算资源管理架构、协议、算法及理论模型方面的研究启发了本文的研究工作,另外,许多学者在基于 QoS 的 Web 服务组合方面的工作也为本文提供了有益的借鉴.本节综述与本文工作密切相关的分布式计算资源管理和基于 QoS 的 Web 服务组合方面已有的工作.

Park认为分布式资源可以因其功能的差别而分为不同的资源类,资源的可获得性和性能都表现出动态性,许多Internet应用需要同时从不同地点、不同管理域获得多种资源才能满足其需求,完成一个应用等同于能够同时获得所规定数量和种类的资源[6].Aron等人在分布式环境下引入资源容器对分散的资源进行集中式管理[7],所有分布式结点的操作系统部署了资源容器,将资源按功能划分,提供给上层的资源容器,从而以嵌套的方式形成管理大范围内资源的总的资源容器.资源容器具有良好的自治性、灵活性和协同性,本文将其引入到服务计算的资源管理系统以适应服务资源的动态性.

类似于上述资源容器的思想,清华大学的Wu和Liu等人提出了网格计算池(grid computing pool,简称GCP)的概念[8,9],GCP将分布于各地的高性能计算机通过高速网络连接起来形成资源池,联合为用户提供计算能力.他们还利用排队论证明了GCP能够在保证QoS的前提下提高服务能力,其概念和理论证明的方法值得我们借鉴.但是,Web服务所集成的资源趋于多元化,计算能力仅成为了众多资源种类中的一种,因此资源池的概念需要进行扩展,而且对资源池的理论分析也有待进一步深化以适合更普遍的环境.

本文将服务资源联合分配问题归化为MMKP问题求解.MMKP是 1998 年Khan研究并发会话多媒体系统QoS自适应问题而首次提出的[10].他将QoS自适应问题建模为MMKP,并给出了解决MMKP的两种算法,第 1 种是基于分枝限界法(branch and bound)的BBLP算法,它可以得出最优解,但其时间复杂度是指数级的.第 2 种是启发式算法HEU,它以多项式级的时间复杂度得出近最优解.随后,Akbar等人改进了HEU算法[11],提出了M-HEU算法,具有更快的速度.目前,为了进一步提高M-HEU算法的速度,Akbar等人提出了运行在多个处理器上的并行M-HEU算法[12].本文在求解MMKP问题时借鉴了Khan和Akbar等人的研究工作.

服务组合领域具有代表性的是Lin等人所做工作:他们提出QBroker体系架构(Broker-based QoS architecture)[13],QBroker作为一种外部的独立代理结构帮助用户收集QoS信息,为服务发现、服务组合和服务选择等提供依据.基于QBroker架构,Lin[4]针对服务序列流模型以及服务一般流模型提出了多种服务选择算法,这些算法旨在QoS需求的限制下,最大限度地提高用户的效益值.尽管Lin等人的工作与本文的研究位于不同的层次,但是在QoS模型方面与我们具有共同的见解[14]:Internet应用具有多维QoS参数的约束,QoS的优劣通过用户满意度来体现,用户满意度成为了反映QoS优劣的统一标准.

2 具有 QoS 保证的服务资源管理

2.1 QG-SRMS系统架构

服务实例层为组合服务生成的 DAG 图中的每个子服务提供具体的服务实例,在组合服务工作流开始执行时触发激活服务实例请求,底层用于支撑 Web 服务运行的服务资源管理系统接受激活服务实例请求(以下简称服务请求),为其分配资源完成该服务实例.本文提出的服务资源管理系统能够保证服务实例向上层所承诺的QoS,因此称其为具有 QoS 保证的服务资源管理系统(QG-SRMS).图 1 给出 QG-SRMS 的系统架构,该架构处理服务请求的流程描述如下:

(1) QG-SRMS 对到达系统的服务请求的资源需求进行解析,确定该服务请求所需要的资源种类.

(2) QG-SRMS中的虚拟资源容器(virtual resource container,简称VRC)根据物理资源的功能进行物理资源的聚合,图 2 中的VRC_1,VRC_2,…,VRC_N这N个虚拟资源容器聚合N种不同功能的物理资源.QG-SRMS根据解析出的服务请求所需要的资源种类,与相应的虚拟资源容器建立绑定关系.

(3) 当服务请求到达运行时间,QG-SRMS 的物理资源分配器根据服务资源联合分配算法,为该服务请求所绑定的所有虚拟资源容器选择特定的物理资源,并分配给该服务请求.

(4) 服务请求获得所需的多种物理资源,运行完毕后释放这些资源,QG-SRMS 对该服务请求处理结束.

QG-SRMS系统架构的特点就是引入多个VRC来管理地理上分散的物理站点上的各种资源,VRC对物理资源进行功能聚类,一个虚拟资源容器聚合了相同功能的一类物理资源,比如,VRC_1聚合计算资源(CPU),VRC_2聚合存储资源,VRC_3聚合实验仪器设备资源.某个服务请求可能需要利用实验仪器设备获取实验数据,然后进行计算,最后将中间数据和计算结果存放到存储器中,因此,该服务请求就需要同时获得以上 3 个VRC中的资源才能完成.图 2 描述了VRC的实现机制,在分散的每个物理站点上部署资源采集Agent(resource collection Agent,简称RCA),RCA与物理站点上的LRM(local resource manager)协商,确定该物理站点贡献给服务资源管理器的资源种类及其比例,然后RCA根据其种类和比例在对应的虚拟资源容器内建立可用资源列表索引,而且RCA负责对其动态更新.利用虚拟资源容器管理服务资源至少具备以下两大优势:

(1) 适应服务资源的动态性:服务请求对应于组合服务 DAG 图中的一个结点,服务实例运行的时间依赖于拓扑排序在其前面的所有结点都完成,因此,服务实例占用资源是在未来的一段时间内.由于服务资源的可获得性和性能呈现出很强的动态性,如果服务请求直接与具体物理资源绑定,将可能由于所绑定的资源在未来时间段内不可用或性能不能满足服务实例要求而导致该服务实例无法成功运行.虚拟资源容器维持服务请求队列,为每个服务请求进行运行时绑定,从而屏蔽了服务资源的动态性,提高了服务请求的成功率.

(2) 高度的自治性和灵活性:服务计算面向大规模的开放环境,不可能对所有物理站点进行集中式的管理,各物理站点需要有良好的自治性,VRC 使得各物理站点可以控制贡献给服务层的资源种类和比例,具有高度的

自治性.另外,虚拟资源容器可以通过的层层嵌套的方式构成更大范围内的虚拟资源容器,具有很强的灵活性.

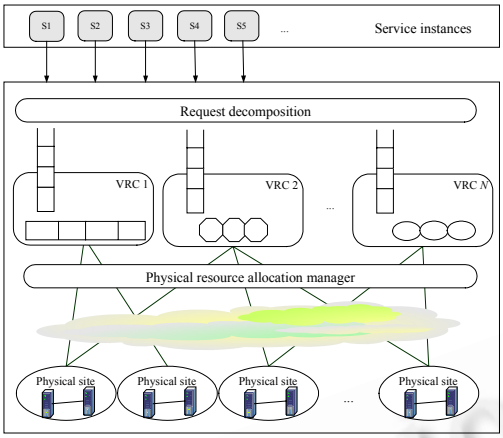

Fig.1 QoS guaranteed service resource management system architecture

图 1 具有 QoS 保证的资源管理系统架构

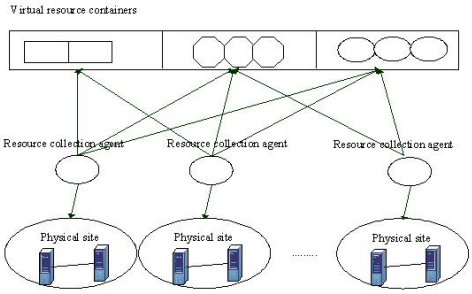

Fig.2 Implementation mechanism of VRC

图 2 虚拟资源容器的实现机制

QG-SRMS 系统架构的物理资源分配器通过为 VRC 选择物理资源使得服务请求能够同时获得多种物理资源,物理资源分配器部署具有 QoS 保证的资源联合分配算法,使所分配的多种物理资源不仅能够保证该服务请求所承诺的 QoS,而且能够优化用户满意度.具有 QoS 保证的资源联合分配算法将在本文的第 3 节中加以论述.

2.2 虚拟资源容器的理论模型及分析

QG-SRMS 系统架构中,VRC 起着关键的作用,它对某个特定服务请求响应时间长短直接影响 QoS 性能,而 VRC 的响应时间取决于它聚合资源的多少.但是,由于各物理站点需要执行本地任务,不可能将所有资源都贡献出来,另外,如果 VRC 聚合资源过量又会造成资源浪费.这就需要 VRC 对聚合资源数量和响应时间进行优化.

假设某服务资源管理系统有N个VRC,分别为$VRC_1,VRC_2,\ldots,VRC_i,\ldots,VRC_N$,服务实例对某类资源的请求,即分解到各虚拟资源容器的请求过程遵循Poisson过程,平均到达速率分别为$\lambda_1,\lambda_2,\ldots,\lambda_i,\ldots,\lambda_N$.假设服务实例请求占用资源的时间(以下称为服务时间)可以遵循任何分布,其期望和方差为有限值.于是,每个聚集c个资源的虚拟资源容器可以建模为$M/G/c$系统,$M/G/c$系统的平均等待时间的近似公式为

$$W_i=\frac{\overline{h}_i^2}{2\overline{h}_i(c-\lambda_i\overline{h}_i)}\left[1+\sum_{n=0}^{c-1}\frac{(\lambda_i\overline{h}_i)^n}{n!}\frac{(c-1)!(c-\lambda_i\overline{h}_i)}{(\lambda_i\overline{h}_i)^c}\right]^{-1} \tag{1}$$

其中,$\overline{h}_i$ 为服务时间的一阶矩,$\overline{h}_i^2$ 为服务时间的二阶矩,λ_i为VRC_i的请求到达速率.

如果服务时间遵循指数分布,VRC的排队模型就可以归化为$M/M/c$系统,文献[8,9]对其作过分析,本文不再研究虚拟资源容器为$M/M/c$模型的情况.以往的研究认为,Web服务器上的文件长度和网站服务时间都遵循重尾分布(heavy-tailed distribution)[15],一种典型的重尾分布叫做BP分布(bounded pareto),它具备两大特性:大方差和重拖尾(即小概率事件会产生较大影响),因而能够较好地刻画实际Internet服务器的服务时间,本文研究服务时间遵循BP分布时,VRC的平均等待时间和聚集资源量之间的关系,BP分布的概率密度函数见式(2),BP分布的一阶矩和二阶矩可以分别利用式(3)和式(4)进行计算.

$$f(x)=\frac{\alpha k^\alpha}{1-(k/p)^\alpha}x^{-\alpha-1},k\le x\le p \tag{2}$$

$$\bar{h}_i = \frac{\alpha}{\alpha-1}\frac{k^\alpha}{1-(k/p)^\alpha}\left(\frac{1}{k^{\alpha-1}}-\frac{1}{p^{\alpha-1}}\right) \tag{3}$$

$$\bar{h}_i^2 = \frac{\alpha}{\alpha-2}\frac{k^\alpha}{1-(k/p)^\alpha}\left(\frac{1}{k^{\alpha-2}}-\frac{1}{p^{\alpha-2}}\right) \tag{4}$$

任何一个排队系统能够正常运行的条件为$\lambda<\mu$,因此,到达流强度与服务能力之间的比值决定了该排队系统的工作负载,于是可以将工作负载定义为

$$\sigma = \frac{\lambda_i}{c\bar{h}_i} \tag{5}$$

我们研究各种k,p取值的 5 种情形下的VRC的平均等待时间和聚集资源量之间的关系,各种k,p取值及其BP分布的期望和方差见表 1.其他各参数取值为α =1.1,σ=0.5,p=10000×k.将虚拟资源容器所聚合的资源数量c在5~30 内变化,由式(1)得出c与平均等待时间W_i的关系,结果如图 3 所示.

Table 1 Experiment parameters (k,p) and the mean and variance of BP distribution

表 1 实验参数(k,p)及 BP 分布的期望和方差

	Case 1	Case 2	Case 3	Case 4	Case 5
k	0.1	1	10	30	50
p	1 000	10 000	100 000	300 000	500 000
Mean	0.662 11	6.621 1	66.211	198.63	331.05
Variance	48.209	4 820.9	20 900 000	388 000 000	520 000 000

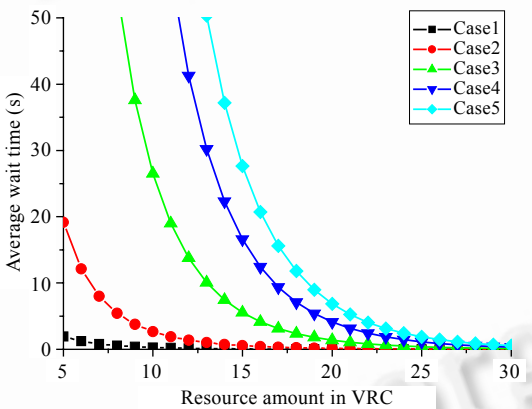

Fig.3 Resource amount in VRC vs. average wait time

图 3 资源聚集量和平均等待时间

VRC 可以根据最新的资源状态将服务请求与资源进行运行时绑定,屏蔽了服务资源的动态性,提高服务请求的成功率.由图 3 可知,如果 VRC 需要维持相同的服务性能(平均等待时间),服务时间的方差越大,VRC 需要聚合越多的资源,即 VRC 可以通过聚合更多的资源消除由资源动态波动幅度增大所带来的服务性能的下降.因此,VRC 可以在不影响 QG-SRMS 性能的前提下适应服务资源的动态性.

从图 3 可看出,k,p不变的情况下,当c较小时,平均等待时间W_i的减小速度极快,然而随着c的逐渐增大,平均等待时间W_i减小速度逐渐变慢.当c=23 时,图中 5 种不同k,p值情形下的平均等待时间W_i都远远小于平均服务时间(约为平均服务时间的 1%),对于用户来讲是完全可以接受的,这也就意味着QG-SRMS能够向用户提供较优的QoS.VRC无须聚合过量的资源就可以将平均等待时间下降到用户可以忍受的范围内,如果虚拟资源容器聚合过量的资源,不仅使QoS(平均等待时间)提高的幅度不大,还造成资源浪费,因此,本文认为VRC只需聚合适中数量的资源就足以使平均等待时间维持在用户可以忍受的范围之内,而不造成资源的浪费.

3 具有 QoS 保证的资源联合分配算法

如第 2 节所描述的,当有服务实例请求到达资源管理系统时,系统首先对服务请求的资源需求进行解析,确定该服务请求所需要的资源种类,然后选取相应的 VRC 所聚合的物理资源完成该请求,系统需要在满足该服务实例所承诺的 QoS 需求的情况下,优化用户满意度.这个问题可以通过归化为 MMKP 问题来解决.

定义 1. 假设服务实例承诺的QoS向量表示为m维的向量$\boldsymbol{Q}=[Q_1,Q_2,\ldots,Q_m]$,需要使用到的虚拟资源容器为$VRC_1,VRC_2,\ldots,VRC_n$,是总的$N$个虚拟资源容器的子集,所聚合的资源数量分别为:$c_1,c_2,\ldots,c_n$.资源$\boldsymbol{r_{ij}}$表示$VRC_i$中的第$j$个资源,其资源QoS属性可以表示为$m$维向量$\boldsymbol{r_{ij}}=[r_{ij1},r_{ij2},\ldots,r_{ijm}]$,资源$\boldsymbol{r_{ij}}$的用户满意度为$U_{ij}$.QG-SRMS为服务实例请求在每个虚拟资源容器中选择一个资源,使得选出的n个物理资源总的QoS参数属性满足服务实例所承诺的QoS向量$\boldsymbol{Q}$,且使得被选出的资源的用户满意度最大,即

$$\mathrm{Max}\sum_{i=1}^{n}\sum_{j=1}^{c_i}U_{ij}x_{ij}\ .$$

满足

$$\sum_{i=1}^{n}\sum_{j=1}^{c_i}r_{ijk}x_{ij}\le Q_k\ \ (k=1,2,...,m),\sum_{j=1}^{c_i}x_{ij}=1\ x_{ij}\in\{0,1\}\ \ i=1,2...,n;j=1,2,...,c_i\ .$$

其中:x_{ij}表示物理资源$\boldsymbol{r_{ij}}$是否被选出,x_{ij}=1 表示VRC_i中的第j个原子资源被选出,$\sum_{j=1}^{c_i}x_{ij}=1$ 表示在VRC_i中仅且仅有选择一个物理资源.本文将满足 QoS 需求向量的约束统一用符号‘≤’表示,其具体含义却视 QoS 参数的不同而异.若 QoS 参数为响应时间,要求为 5s,则满足此 QoS 参数指小于 5s 的时间;若 QoS 参数为可靠性,要求为 80%,则满足此 QoS 参数指大于 80%的可靠程度.

3.1 QoS模型

尽管 QoS 参数的表现形式千差万别,但是其优劣最终由用户满意度来反映.用户满意函数描述了某个特定的 QoS 参数与其用户满意度之间的映射关系.本节提出我们所使用的 QoS 模型,最终由此导出用户满意度函数.

定义 2. 用户满意度.区间[0,1]内的实数,值越大表示用户对服务越满意,取值 1 表示用户需求得到完全满足;取值 0 表示用户的最低要求未被满足.

定义 3. 用户满意函数.用来描述某个特定的QoS参数与用户满意度之间的映射关系.f_{usk}表示第k维QoS参数与用户满意度之间的映射关系,用户满意函数的定义域因不同的QoS参数而异,值域为[0,1]内的实数.U_{ij}表示资源$\boldsymbol{r_{ij}}$的m维QoS向量的总体用户满意度.

定义 4. 方向特征值:QoS参数可以分为积极度量和消极度量两类,消极度量指其值越高,质量越低的QoS参数,如响应时间.积极度量指其值越高,质量越高的QoS参数,如CPU份额、声誉等.定义方向特征值$dir_k\in\{-1,1\}$表示QoS参数的方向,dir_k=−1 表示该QoS参数为消极度量,dir_k=1 表示该QoS参数为积极度量.

定义 5. 协商灵活水平参数.从 QoS 保证机制的角度来看,QoS 参数包括硬 QoS 和软 QoS,硬 QoS 参数是网格服务的必备属性,其取值不能改变.软 QoS 参数是网格服务的辅助属性,其取值可以在一定的范围内变化.定义协商灵活水平参数来描述QoS参数的硬软程度,$c_{flex}\subset[0,1]$,c_{flex}=0 表示该QoS参数为硬QoS参数,取值无法变动;c_{flex}越接近于 1 表示QoS参数可变动的范围越大.

用户满意度由资源的QoS属性和应用QoS需求两者共同决定,两者之间的距离越小用户满意度就越大,因此,用户满意函数就是以资源的QoS属性和应用QoS需求之间的距离为自变量、用户满意度为函数值的函数.对于单个QoS参数,其差值由供求的QoS值、协商灵活水平参数决定,Siddiqui提出利用式(6)计算单个QoS参数的差值[16].

$$d_k=\frac{Q_k-r_{ijk}}{c_{flex}}\tag{6}$$

结合供求QoS参数之间差值和方向参数,得出供求QoS参数距离的计算公式,见式(7).当c_{flex}=0,即QoS参数

为硬QoS参数时,如果$r_{ijk}>Q_k$,即资源提供的QoS参数没有满足需求QoS参数时,距离为无穷大.当$(dir_k\cdot d_k)<0$,即资源所提供的QoS参数超出网格应用的QoS需求时,距离为 0.其他情况即该QoS参数为软QoS参数,且资源所提供的QoS参数值达不到应有需求的QoS参数值,此时的QoS距离为差值的绝对值.

$$D(Q_k,r_{ijk})=\begin{cases}\infty, & c_{flex}=0 \text{ and } r_{rjk}>Q_k\\ 0, & (dir_k\cdot d_k)<0\\ |d_k|, & \text{other}\end{cases} \tag{7}$$

第k个QoS参数所带来的用户满意度由QoS距离来决定,提出如式(8)所示的一种计算方法.当QoS距离为无穷大时,用户满意度为 0;当QoS距离为 0 时,用户满意度为 1;QoS距离为有限值时的用户满意度是QoS距离与该QoS参数的权重的乘积.由于资源$\boldsymbol{r_{ij}}$的QoS参数是一个m维向量,因此将$\boldsymbol{r_{ij}}$的整体用户满意度U_{ij}定义为m维QoS参数所带来用户满意度的总和,计算公式见式(9).

$$f_{usk}(D(Q_k,r_{ijk}))=\begin{cases}0, & D(Q_k,r_{ijk})=\infty\\ 1, & D(Q_k,r_{ijk})=0\\ \omega_k D(Q_k,r_{ijk}), & \text{other}\end{cases} \tag{8}$$

$$U_{ij}=\sum_{k=1}^{m} f_{usk}(D(Q_k,r_{ijk})) \tag{9}$$

通过以上一系列的定义可以求解出资源为某个特定的 Web 服务所带来的用户满意度,用户满意度成为资源联合分配算法的目标函数,第 3.2 节和第 3.3 节分别介绍两种资源联合分配算法,为了清晰地描述算法,我们将算法中变量的记法和描述列于表 2 之中.

Table 2 Notations and descriptions in algorithms

表 2 算法中变量的记法和描述

Notations	Description
Q	m dimension vector representing QoS vector which is promised by service instance
$Qfix$	m dimension vector representing QoS vector which is provided by selected resources
Uf	Users' degree of satisfaction provided by selected resources
QLP	m dimension vector representing QoS vector which is provided by selected resources, namely $Q=Qfix+QLP$
r	$n*Max(c_i)$ dimension matrix, $r[i][j]$ represents QoS property vector of resource $\boldsymbol{rij}$
U	$n*Max(c_i)$ dimension matrix, $U[i][j]$ represents users' degree of satisfaction of resource $\boldsymbol{rij}$
g	n dimension vector, $g[i]=1$ represents the resource of VRC_i has been fixed
b	the serial number of extended VRC
$U\max$	Estimated upperbound of users' degree of satisfaction of all live nodes in search tree
$nfree$	The amount of VRC whose resources have not been selected
fk	$fk=Qfix[k]/Q[k]$, the infeasibility factor of the k-th QoS parameter (see definition 7)
x	n dimension solution vector, $x[i]$ represents the serial number of resource in VRC_i, $x[i]\in\{1,2,\ldots,c_i\}$

3.2 RA_BBLP算法

MMKP 是 NP-complete 问题,本节提出基于分枝限界法的服务资源联合分配算法:RA_BBLP.其本质是穷举算法,通过迭代生成搜索树直到搜索出最优资源联合分配方案为止,最优资源联合分配方案是指选择出的多 VRC 中的物理资源能够保证服务实例所承诺的 QoS,且用户满意度最大,首先定义 RA_BBLP 中的搜索树.

定义 6. 搜索树是一棵高度为n,度为$\mathrm{Max}(c_1,c_2,\ldots,c_n)$的树.树的每个结点表示一个搜索状态,第$l$层的树结点表示$l$个VRC的物理资源已经确定,剩余$n-l$个VRC的物理资源未曾确定的状态,其孩子结点由未确定的$n-l$个VRC中的一个VRC的物理资源扩展而得.孩子结点尚未产生的结点称为活结点.所有活结点中用户满意度最大

的结点称为扩展结点.可以选择扩展结点中任意未确定物理资源的VRC进行扩展,被选中的VRC称为扩展VRC.若某结点l个已经确定物理资源的VRC提供的QoS参数违反了服务实例所承诺的QoS,称该结点为不可行结点,否则称为可行结点.

RA_BBLP 通过以下步骤构造搜索树求解最优资源联合分配方案:

(1) 初始化搜索树,搜索树的根结点是 n 个 VRC 的物理资源都未曾确定的状态.

(2) 为所有活结点计算用户满意度的上界,将用户满意度上界最大的活结点作为扩展结点.

(3) 若扩展结点的所有 VRC 中的物理资源都已经确定,这个扩展结点的资源分配方案即为最优解,算法结束.

(4) 若扩展结点还存在未确定物理资源的 VRC,选择未确定物理资源的某个 VRC 作为扩展 VRC,遍历该扩展 VRC 所聚合的所有物理资源:判断将扩展 VRC 中某物理资源作为选择方案生成的新结点是否为可行结点,如果是,则添加该新结点到搜索树,反之则不添加.回到第(2)步.

RA_BBLP 算法如何计算用户满意度上界和确定扩展 VRC 的方法解释如下:

(1) 计算用户满意度上界:搜索树的结点已经确定了一部分VRC的物理资源,假设为l个.因此,已选资源产生的用户满意度总和U_f为

$$U_f=\sum_{\substack{i=1,2...l\\ j=1,2...c_i\\ g[i]=1}} x_{ij}U_{ij} \tag{10}$$

计算出剩余的$n-l$个VRC所选出的物理资源所要保证的QoS承诺:$Q_{LP}=Q-Q_{fix}$.因此,计算用户满意度上界就是计算在Q_{LP}的约束下$n-l$个VRC中所选择物理资源带来的用户满意度的上界,我们如果将x_{ij}放宽到[0,1]区间,该问题就转化为线性规划问题(linear programming),MATLAB提供的linprog函数可以方便地求解线性规划问题,将线性规划所求得$n-l$个VRC中用户满意度最大值记为U_{LP},则该结点的用户满意度上界为

$$U_{\max}=U_f+U_{LP} \tag{11}$$

(2) 确定扩展VRC:在搜索树结点的$n-l$个未曾确定物理资源的VRC中选择其中之一作为扩展VRC的方法依然根据线性规划的计算结果,本文选择由线性规划计算出的x_{ij}最大的那个VRC为扩展VRC.

RA_BBLP算法的程序伪代码见算法 1,理论上,RA_BBLP是时间复杂度为指数级的算法,时间复杂度为$O(c^n)$,此处假设n个VRC聚合的物理资源数量相等,都为c.

算法 1. RA_BBLP 算法.

输入:服务实例承诺的 QoS 向量 $\boldsymbol{Q}$;n 个 VRC 的所有物理资源的 QoS 属性及其用户满意度.

输出:n 维解空间向量 $\boldsymbol{x}$.

描述:

1: $U_f\leftarrow 0, n_{free}\leftarrow n, Q_{LP}\leftarrow Q, g\leftarrow 0$; /*初始化算法变量*/

2: $U_{\max}\leftarrow$Mat_LP(x,g,U,r,Q_{LP},U_f);
/*利用线性规划法计算初始状态活结点的用户满意度最大值*/

3: find $x[b][j]\leftarrow$Max $i=1,2,\ldots,N;j=1,2,\ldots,c_i; g[i]=0(x[i][j])$; /*确定初始状态的扩展VRC b */

4: TreeInsert ($x,g,U_f,b,U_{\max},Q_{LP},n_{free}$); /*插入搜索树中的第 1 个结点*/

5: **while** (TRUE) **do**

6: $t\leftarrow$TreeExtractMax(); /*选择搜索树中用户满意度最大的活结点*/

7: $x\leftarrow t.x$, $g\leftarrow t.g, b\leftarrow t.b$;

8: **if** ($t.n_{free}$=0) **then**

9: return $t.x$; /*所有 VRC 的资源都已确定,算法结束,返回 t */

10: **end if**

11: $g[b]\leftarrow 1, x[b]\leftarrow 0$;

12: **for** $j\leftarrow 1$ **to** c_b /*遍历扩展VRC中所有的物理资源*/

13: **if** ($Qfix+r[b][j]<=\boldsymbol{Q}$) **then** /*判断加入扩展 VRC 的第 j 个资源时是否为可行结点*/
14: $x[b]\leftarrow j, Q_{LP}\leftarrow t.\boldsymbol{Q_{LP}}-r[b][j]$, $U_f\leftarrow t.U_f+U[b][j]$;
15: $U_{\max}\leftarrow$Mat_LP ($\boldsymbol{x},\boldsymbol{g},U,\boldsymbol{r},\boldsymbol{Q_{LP}},U_f$); /*计算新结点的用户满意度上界*/
16: find $x[b][j]\leftarrow$Max i=1,2,…,N;j=1,2,…,c_i;$g[i]$=0($x[i][j]$); /*确定新结点的扩展VRC*/
17: TreeInsert ($\boldsymbol{x},\boldsymbol{g},U_f,b,U_{\max},\boldsymbol{Q_{LP}},n_{free}$); /*插入新结点*/
18: **else**
19: continue; /* 加入第 j 个资源时为不可行结点,继续下一个物理资源*/
20: **end for**
21: **end while**

3.3 RA_MHEU算法

由于RA_BBLP是时间复杂度为指数级的算法,不适用于VRC较多的情况,借鉴Akbar等人提出的M-HEU算法[11],本节提出复杂度为多项式级的算法:RA_MHEU.它是启发式算法,借鉴 0-1KP问题启发式算法的思路(试图选择效益值与重量比最大的背包),试图选择用户满意度较高、节省QoS较多的资源,这些资源留给剩余资源更大的QoS范围.比如,VRC_i聚合物理资源$\boldsymbol{r_{ij}}$和$\boldsymbol{r_{ik}}$,用户满意度$U_{ik}>U_{ij}$,假设某服务实例承诺的响应时间为t,$\boldsymbol{r_{ij}}$的响应时间为t_1,$\boldsymbol{r_{ik}}$的响应时间为t_2,$t_1>t_2$,此时RA_MHEU算法的启发式规则就认为$\boldsymbol{r_{ik}}$比$\boldsymbol{r_{ij}}$具有更大的优势,因为$\boldsymbol{r_{ik}}$相比于$\boldsymbol{r_{ij}}$提高了用户满意度,还缩短了响应时间(响应时间这个QoS参数得到了节省),使得剩余VRC中的物理资源有更宽裕的响应时间.但是,服务实例的QoS承诺和资源的QoS属性是m维向量,因此需要对m维QoS综合的大小进行量化,RA_MHEU算法将已选出物理资源已经提供的QoS属性向量$\boldsymbol{Q_{fix}}$作为权重进行m维QoS的量化,即令

$$\Delta r_{ij}=(r[i][x[i]]-r[i][j])\cdot Q_{fix}/|Q_{fix}| \tag{12}$$

若$\Delta r_{ij}>0$,则说明物理资源的总体QoS属性得到了节省.当Δr_{ij}无法再得到提高时,RA_MHEU算法的规则就试图提高单位QoS属性上的用户满意度Δp,令

$$\Delta p=(U[i][x[i]]-U[i][j])/\Delta r \tag{13}$$

若$\Delta p>0$,则RA_MHEU算法认为物理资源的单位QoS属性的用户满意度得到了提高.若$\Delta r_{ij}>0$,RA_MHEU算法就是在一组可行的初始解的基础上不断按照以上两条规则对解向量进行升级,从而得出近最优解,本节将RA_MHEU算法分成寻找初始解和升级解向量两步进行论述.

RA_MHEU 寻找初始解步骤:

定义 7. 违背因子:定义$f_k=\boldsymbol{Q_{fix}[k]/Q[k]}$为第$k$个QoS参数的违背因子,$f_k$越大表示选出物理资源所提供的第$k$个QoS参数越接近服务实例的QoS承诺,未选物理资源在第k个QoS参数上的变动范围就越小,该QoS参数就越容易导致解集不可行.$f_k>1$ 表示第k个QoS参数已经违背服务实例的QoS承诺,解集不可行.

RA_MHEU 算法寻找初始解的步骤如下:

(1) 将所有 VRC 中的物理资源按用户满意度非递减排列,将每个 VRC 的第 1 个物理资源作为初始解,即以用户满意度最小的物理资源作为初始解.

(2) 如果当前的解向量为可行解,则进入 RA_MHEU 的升级解向量步骤.

(3) 遍历 n 个 VRC,选择 QoS 节省量Δr 最大的物理资源作为替换方案,令 $f'k$ 为替换后的第 k 个 QoS 参数的违背因子.如果对于任意 k,$k\in$ {1,2,3,…,m}同时满足以下 3 条原则就将该物理资源替换解向量中的值,否则不作替换.回到步骤(2),3 条原则是:

(i) 令$\delta=k$,s.t.Max $\{f_k\}$,即f_δ为违背因子最大值,有:$f'_\delta<f_\delta$;

(ii) 对任意ε=1,2,…,m,如果$f_\varepsilon>1$ 且$\varepsilon\neq\delta$,有:$f'_\varepsilon\leq f_\varepsilon$;

(iii) 对任意ε=1,2,…,m,如果$f_\varepsilon\leq 1$ 且$\varepsilon\neq\delta$,有:$f'_\varepsilon\leq 1$;

(4) 可行解向量不存在,RA_MHEU 算法结束.

RA_MHEU算法的在寻找初始解时,一旦判定解向量可行就进入升级解向量步骤.否则,通过替换某个VRC中的物理资源重新判定解向量是否可行,但是替换后的违背因子需要满足 3 条原则,即替换后的最大违背因子

需要减小,对于已经违背服务实例承诺的QoS参数(f_ε>1)在替换后不增加,对于未违背服务实例承诺的QoS参数(f_ε≤1)在替换后仍然不违背.如果所有VRC的违背因子在不断的进步后,仍然找不到可行的解向量则可以判定保证服务实例的QoS资源分配方案不存在,RA_MHEU算法失败.

RA_MHEU 升级解向量步骤:

(1) 遍历 n 个 VRC,选择比初始可行解具有更高用户满意度的物理资源,如果替换后解向量仍然可行,且Δr大于当前的最大值,则替换当前 VRC 中的物理资源.

(2) 对于此物理资源如果Δp 大于当前的最大值,则替换当前 VRC 中的物理资源,回到步骤(1).

RA_MHEU 算法寻找初始解步骤和升级解向量步骤的伪代码见算法 2.1 和算法 2.2.RA_MHEU 算法的时间复杂度为$O(n^2(c-1)^2m)$,n 是多项式函数.

算法 2.1. RA_MHEU 寻找初始解.

输入:服务实例承诺的 QoS 向量 $\boldsymbol{Q}$;n 个 VRC 的所有物理资源的 QoS 属性及其用户满意度.

输出:可行初始解 $\boldsymbol{x}$.

描述:

1: **for** i←1 **to** n

2: Order $\boldsymbol{r_{ij}}$ in VRC_i according to U_{ij} with non-decreasing order;

3: $\boldsymbol{x}[i]$=1; /*取每个 VRC 中的第 1 个物理资源为初始化解*/

4: **end for**

5: **for** i←1 **to** n

6: $Q_{fix}=\sum_{i=1}^{n} r[i][x[i]]$; /*计算当前解向量所提供的 QoS */

7: **if** (Q_{fix}<=Q) **then** /*判断当前解向量是否为可行解*/

8: Go to Step 2**;** /*如果当前解向量是可行解,则进入步骤 2 */

9: **else**

10: $\boldsymbol{x}'[i]=j$, s.t. Max$\{\Delta r_{ij}\}$=Max$\{(r[i][x[i]]-r[i][j])Q_{fix}/|Q_{fix}|\}$

/*将 QoS 节省量Δr 最大的资源作为替换资源*/

11: $Q'_{fix}=Q_{fix}-r[i][x[i]]+r[i][x'[i]]$ /*计算替换后解向量所提供的QoS */

12: $\delta=k$,s.t.Max $\{f_k\}$; /*选择违背因子的最大值*/

13: **if** ($f'_\delta<f_\delta$ and $f'_\varepsilon<=f_\varepsilon$ s.t.f_ε>1 & $\varepsilon\neq\delta$ and $f'_\eta<=f_\eta$ s.t.f_η<=1 & $\eta\neq\delta$) **then**

/*判断替换资源是否的 QoS 属性的违背因子是否满足 3 条原则*/

14: $\boldsymbol{x}[i]=\boldsymbol{x}'[i]$,$Q_{fix}=Q'_{fix}$;

15: **end if**

16: **end for**

17: return 0; /*可行解向量不存在,RA_MHEU 算法结束*/

算法 2.2. RA_MHEU 升级解向量.

输入:可行初始解 $\boldsymbol{x}$.

输出:升级后的近最优解向量 $\boldsymbol{x}$.

描述:

1: **while** (TRUE) **do**

2: $\Delta r_{max}=0,\Delta p_{max}=0$;

3: **for** i←1 **to** n

4: **for** j←$\boldsymbol{x}[i]$ **to** c_i /*不断选择用户满意度较高的物理资源*/

5: **if** ($\boldsymbol{Q_{fix}}-r[i][x[i]]+r[i][j]$<=$Q$) **then** /*判断替换后的解向量是否仍然可行*/

6: $\Delta r_{ij}=(r[i][x[i]]-r[i][j])Q_{fix}/|Q_{fix}|$;
7: **if** ($\Delta r>\Delta r_{\max}$) **then** /*判断替换后解向量的QoS节省量是否提高*/
8: $\Delta r_{\max}=\Delta r, i'=i, j'=j$; /*如果QoS节省量提高,升级解向量,更新$\Delta r_{\max}$ */
9: **if** ($\Delta r_{\max}=0$) **then** /*若QoS节省量无法再提高*/
10: $\Delta p=(U[i][x[i]]-U[i][j])/\Delta r$; /*计算单位 QoS 属性上的用户满意度*/
11: **if** ($\Delta p>\Delta p_{\max}$) **then** /*判断替换后解向量单位QoS属性上的用户满意度是否提高*/
12: $\Delta p_{\max}=\Delta p, i'=i, j'=j$; /*如果单位QoS属性上的用户满意度提高,升级解向量,更新$\Delta p_{\max}$ */
13: **if** ($\Delta r_{\max}=0$ and $\Delta p_{\max}=0$) return x; /*若Δr和Δp都无法再提高,返回当前解向量*/
14: $Q_{fix}=Q_{fix}-r[i'][\rho[i']]+r[i'][j']$;
15: $\boldsymbol{x}[i']=j'$; /*更新解向量*/
16: **else**
17: continue; /*如果替换后的解向量不可行,则继续查看下一个物理资源*/
18: **end for**
19: **end for**
20: **end while**

4 仿真实验

QG-SRMS 系统架构的性能由 VRC 的平均等待时间和资源联合分配的时间共同决定,本文第 2.2 节通过对 VRC 的理论分析说明只要 VRC 聚合适量的物理资源就可以将平均等待时间降低到用户可以忍受的范围之内,因此,VRC 不会成为 QG-SRMS 系统架构的性能瓶颈.物理资源分配器可以部署 RA_BBLP 或 RA_MHEU 算法,这两种算法的性能直接影响 QG-SRMS 的性能,本节通过仿真实验分析 RA_BBLP 和 RA_MHEU 算法的性能,由于这两种算法的性能指标包括算法执行时间和解的用户满意度,因此,我们定义两个指标:

执行时间比,即 RA_BBLP 算法的执行时间与 RA_MHEU 算法执行时间之比.

$$\sigma_T=\frac{T_{RA\_BBLP}}{T_{RA\_MHEU}}.$$

用户满意度比,即RA_MHEU算法所求得解集的用户满意度与RA_BBLP算法求得最优解的用户满意度之比,显然有$\sigma_U\le 1$.

$$\sigma_U=\frac{U_{RA\_MHEU}}{U_{RA\_BBLP}}.$$

我们使用C++和MATLAB混合编程技术实现两种算法,通过C++程序调用MATLAB接口实现线性规划和产生随机数.为了简化实验的实现,假设n个VRC都聚集相同量的资源c.每个测试用例由m维的服务实例承诺的QoS向量$\boldsymbol{Q}$、$c\times n$个m维资源属性向量$\boldsymbol{r_{ij}}$及$c\times n$个资源的用户满意度U_{ij}所组成,实验按照如下步骤产生测试用例:

(1) 随机生成m个随机数组成向量$\boldsymbol{Q}$,假设Qc_k为第k个QoS分量的最大值,满足$Q_k=n\times Qc_k\times 0.5$.

(2) 将Qc_k作为第k个QoS分量的最大值,随机生成的$c\times n$个资源属性向量,满足$r_{ijk}=r$ and (Qc_k).

(3) 由供求QoS的距离所决定,假设Q_k/n为平均的第k个需求QoS分量,将式(9)稍作变化得出式(14):

$$U_{ij}=\sum_{k=1}^{m}\omega|Q_k/n-r_{ijk}|+rand(\ ) \tag{14}$$

通过式(14)可以计算出$c\times n$个资源的用户满意度U_{ij},它反映了r_{ijk}与U_{ij}的线性约束关系.

实验在 3 种情况下进行比较,每种情况的 m 值不相同,分别取 3、5 和 7.在这 3 种情况下都取 c=5,将 n 在区间[5,50]之内变化.对于每个不同的 m,n 都产生 10 个测试用例,最后分别将两种算法 10 组输出结果的平均值作为最后实验结果,得到图 4 和图 5.

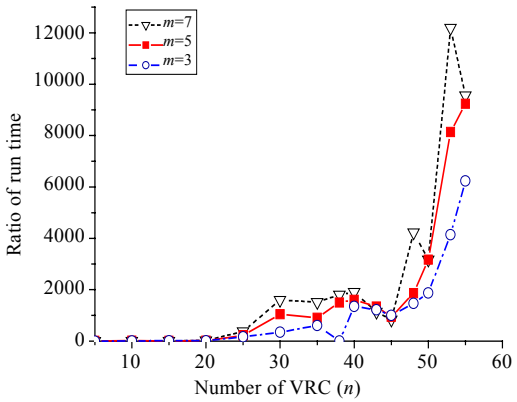

Fig.4 RA_BBLP vs. RA_MHEU (σ_T)

图 4 RA_BBLP vs.RA_MHEU(执行时间比)

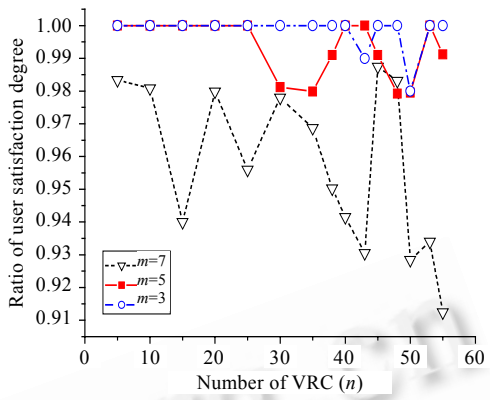

Fig.5 RA_BBLP vs. RA_MHEU (σ_U)

图 5 RA_BBLP vs.RA_MHEU(用户满意度比)

图 4 给出执行时间比(σ_T)随着VRC数量增大的变化曲线,当规模较小(即n较小时),RA_BBLP算法执行时间大约在RA_MHEU算法执行时间的 10 倍左右,随着n的增大,执行时间比(σ_T)随之增大,最大值超过了 10000,这说明随着VRC数量的增加,RA_BBLP算法的执行时间明显增加,最大值时RA_MHEU算法的执行时间仅为RA_BBLP算法的 0.01%.但是,执行时间比(σ_T)的增长趋势并非是指数级增长,而且有时甚至会随着n的增大或m的增大而减小,这说明RA_BBLP算法并非一定需要指数级的时间才能完成,某些情况下可能收敛得极快,并且RA_BBLP算法的收敛速度因测试实例而定,并无特定的规律可依.其原因在于好的情况下,线性规划能够准确地估算出用户满意度上界,能够较快地引导RA_BBLP算法寻找到最优解,而在坏的情况下,RA_BBLP算法需要生成搜索树的几乎全部结点方能找到最优解.

图 5 给出RA_MHEU算法所求解出的用户满意度与最优用户满意度之间的比值,可以看到,σ_U始终大于 0.9,而在QoS维数较小的情况下,σ_U几乎接近于 1,说明RA_MHEU算法总能求解出与最优解相差不远的近最优解,尤其是在QoS维数较小时.

通过以上两组的比较,RA_MHEU 算法是一种收敛速度极快所求出的解集接近于最优的启发式算法,是QG-SRMS 中理想的资源联合分配算法.而 RA_BBLP 算法尽管不适用于 VRC 种类较多时,但它能够保证求得最优解,可以作为其他算法的参照系.

5 结论与下一步工作

本文研究服务计算中底层服务资源管理系统如何对服务实例的 QoS 进行保证的问题,首先提出一种具有 QoS 保证的资源管理系统架构——QG-SRMS,引入虚拟资源容器聚合同类资源,使得各物理站点具有高度的自治性和灵活性,它通过聚集适量的资源就可以在不影响 QG-SRMS 性能的前提下适应服务资源的动态性.其次,将保证服务实例向组合服务所承诺 QoS 的前提下的资源联合分配问题归化为 MMKP 问题,并提出两种资源联合分配算法,实验结果表明,RA_BBLP 算法能够保证求得最优解,可以作为其他算法的参照系,而 RA_MHEU 算法收敛速度极快,所求出的解集接近于最优,可以作为 QG-SRMS 中理想的资源联合分配算法.

在下一步的工作中,我们计划将 QG-SRMS 系统架构实现到基于 Globus Toolkit 4 开发的校园网格平台之中,对海量数据处理和分析、数字教育资源集成和共享等典型网格应用的 QoS 做出可靠的保证.

References:

[1] Huhns MN, Singh MP. Service-Oriented computing: Key concepts and principles. Internet Computing, 2005,9(1):75−81.

[2] Papazoglou MP, Traverso P, Dustdar S, Leymann F, Kramer BJ. Service-Oriented computing: A research roadmap. In: Cubera F, ed. Service Oriented Computing (SOC), No. 05462 in Dagstuhl Seminar Proceedings, 2006. http://d.wanfangdata.com.cn/NSTLQK_NSTL_QK16770010.aspx

[3] Li XN, Hao KG, Zhao K. A business process management model. Chinese Journal of Computers, 2008,31(1):104−111 (in Chinese with English abstract).

[4] Tao Y, Zhang Y, Lin K J. Effective algorithms for Web services selection with end-to-end QoS constraints. ACM Trans. on the Web (TWEB), 2007,1(1):1−26.

[5] Wen JJ, Chen JL, Peng Y. A method of heuristic Web services composition based on goal distance estimate. Journal of Software, 2007,18(1):85−93 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/85.htm

[6] Park J. A deadlock and livelock free protocol for decentralized internet resource coallocation. IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2004,34(1):123−131.

[7] Aron M, Drushnel P, Zwaenepoel W. Cluster reserves: A mechanism for resource management in cluster-based network servers. In: Proc. of the Int'l Conf. on Measurement and Modelling of Computer Systems (SIGMETRICS 2000). Santa Clara: ACM, 2000. 90−101.

[8] Wu YW, Yang GW, Mao JY, Shi SM, Zheng WM. Grid computing pool and its framework. In: Proc. of the Int'l Conf. on Parallel Processing Workshops (ICPPW 2003). Kaohsiung: IEEE Computer Society, 2003. 271−277. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1240380

[9] Liu P, Shi Y, Li SL. Computing pool: A simplified and practical computational grid model. In: Proc. of the Int'l Conf. on Grid and Cooperative (GCC 2003). Shanghai: Springer-Verlag, 2003. 661−668. http://www.springerlink.com/content/qwcrl1c3c1ucg3vm/

[10] Khan S. Quality adaptation in a multisession multimedia system: Model, algorithms and architecture [Ph.D. Thesis]. Department of ECE, University of Victoria, 1998.

[11] Akbar M, Manning E, Shoja G, Khan S. Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In: Proc. of the Int'l Conf. on Computational Science (ICCS 2001). San Francisco: Springer-Verlag, 2001. 659−668.

[12] Shahriar M, Akbar MM, Rahman MS, Newton MAH. A multiprocessor based heuristic for multi-dimensional multiple-choice knapsack problem. The Journal of Supercomputing, 2008,43:257−280.

[13] Tao Y, Lin K J. A broker-based framework for QoS-aware Web service composition. In: Proc. of the Int'l Conf. on e-Technology, e-Commerce and e-Service (EEE 2005). Hong Kong: IEEE Computer Society, 2005. 22−29. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1402263

[14] Wu ZA, Luo JZ, Song AB. QoS-Based grid resource management. Journal of Software, 2006,17(1):2264−2276 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2264.htm

[15] Subrata R, Zomaya AY, Landfeldt B. Game-Theoretic approach for load balancing in computational grids. IEEE Trans. on Parallel and Distributed Systems, 2008,19(1):66−76.

[16] Siddiqui M, Villazon A, Fahringer T. Grid capacity planning with negotiation-based advance reservation for optimized QoS. In: Proc. of the ACM/IEEE Supercomputing (SC 2006) Conf. ACM. 2006. 103. http://portal.acm.org/citation.cfm?id=1188455.1188563

附中文参考文献:

[3] 李向宁,郝克刚,赵克.一种新的业务过程管理模型.计算机学报,2008,31(1):104−111.

[5] 温嘉佳,陈俊亮,彭泳.基于目标距离评估的启发式Web Services组合算法.软件学报,2007,18(1):85−93. http://www.jos.org.cn/1000-9825/18/85.htm

[14] 伍之昂,罗军舟,宋爱波.基于QoS的网格资源管理.软件学报,2006,17(1):2264−2276. http://www.jos.org.cn/1000-9825/17/2264.htm

伍之昂(1982－),男,江苏南京人,博士生,主要研究领域为网格计算,服务计算.

宋爱波(1970－),男,博士,副教授,CCF 会员,主要研究领域为网格计算,海量数据处理,Petri 网.

罗军舟(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为下一代网络体系结构,协议工程,网络安全和管理,网格与服务计算.

曹玖新(1967－),男,博士,副教授,CCF 高级会员,主要研究领域为网络安全,服务计算.