ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.5, May 2006, pp.1068−1077
DOI: 10.1360/jos171068

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

服务组合中一种自适应的负载均衡算法*

李文中[1,2+], 郭 胜[1,2], 许 平[1,2], 陆桑璐[1,2], 陈道蓄[1,2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

An Adaptive Load Balancing Algorithm for Service Composition

LI Wen-Zhong[1,2+], GUO Sheng[1,2], XU Ping[1,2], LU Sang-Lu[1,2], CHEN Dao-Xu[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: Phn: +86-25-83597284, E-mail: lwz@dislab.nju.edu.cn, http://www.nju.edu.cn

Li WZ, Guo S, Xu P, Lu SL, Chen DX. An adaptive load balancing algorithm for service composition. *Journal of Software*, 2006,17(5):1068−1077. http://www.jos.org.cn/1000-9825/17/1068.htm

Abstract: Service Composition enables a new way to create new services by assembling independent service components. One of the important goals in service composition is load balancing across service replicas. In this paper, a novel adaptive distributed load capacity based (LCB) algorithm is presented to solve the load balancing problem in service composition. LCB algorithm uses service routing mechanism to search services and transfer the application data. By self-adaptively calculating the load capacity (LC) metric, LCB algorithm can choose a reasonable service path. LC metric reflects the workload of the service node, and is adaptively adjusted according to the variations of the server's load. Compared with the previous researches, LCB algorithm needn't the global topology information and load information. It is much suitable for dynamic service composition in distributed systems. Simulations show that LCB algorithm performs well in terms of load balancing across the service replicas.

Key words: service composition; load balancing; service overlay network; service routing; quality of service

摘 要: 服务组合可以整合网络上现有的多种异构服务,形成新的服务.针对服务组合中服务路径的选择和负载均衡问题,提出了一种自适应的分布式负载均衡算法——LCB(load capacity based algorithm)算法.LCB 算法使用服务路由来查找服务和转发数据,使用负载容率(load capacity,简称 LC)测度来进行服务副本的选择,从而建立一条适当的组合服务路径.LC 测度是对服务器负载的估算,它根据服务器的负载波动信息不断地进行自适应的调整,从而实现多个服务副本之间的负载均衡.与现有的服务组合负载均衡算法相比,LCB 算法不需要知道服务器的最大负载量和当前负载信息,而且具有更好的可扩展性,更适用于分布式环境下动态服务副本的组合.模拟实验表明,LCB 算法具有良好的负载均衡效果.

* Supported by the National Natural Science Foundation of China under Grant No.60402027 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2004AA112090 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312002 (国家重点基础研究发展规划(973)); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2005411 (江苏省自然科学基金)

Received 2005-09-04; Accepted 2005-10-11

关键词: 服务组合;负载均衡;服务覆盖网;服务路由;服务质量
中图法分类号: TP393 文献标识码: A

Web 服务是一种新兴的分布式 Web 应用模式,是 Web 上数据和信息集成的有效机制.当前,Internet 上正出现越来越多的规模巨大且内容丰富的服务,需要有一种新的机制来整合和集成这些服务.在 Web 服务的研究中,服务组合(service composition)是一个极具挑战性的问题[1].

服务组合可以集成现有的简单服务,组合形成复杂服务,快速而又灵活地构建功能强大的新应用,对于网络资源的重用和协同具有重要意义[2].首先,现实中的应用一般很复杂,为了分散和简化应用逻辑,提高服务可用性,单个 Web 服务不可能做得很复杂.因此,复杂服务需要组合多个简单的 Web 服务.其次,目前已经存在丰富多样的异构 Web 服务,为了将这些分散的服务有机地组织成新的系统,需要进行服务组合.再次,各种异构的硬件设备(如 PC,PDA 等)和网络服务访问方式(如有线、无线等)不断涌现,为了实现随时随地的普适计算(pervasive computing),同一种服务要求以多种版本投递给用户[3],传统的信息存储站点、商业站点、网络服务和搜索引擎需要以一种新的方式结合起来,提供新的服务,也要由服务组合来完成.

1 问题描述

我们研究在服务覆盖网(service overlay network)中的服务组合问题.服务覆盖网是由服务节点构成的覆盖网络,服务提供者可以在服务节点上部署相应的服务.在服务覆盖网中,每类服务都可以有多个实例或称为服务副本,它们提供相同或不同级别的服务质量,可以实现覆盖网络中负载均衡并提供有弹性的容错[4].服务组合通过选择一组所需的实例,在服务覆盖网中形成路径,构成复杂的新应用.

如何选择服务路径是服务组合中一个具有挑战性的问题[5].出于负载均衡以及保证服务可用性的需求,服务提供者会在不同的网络节点上部署和管理多个服务副本.因此,一个组合服务可以由多条服务路径来实现.服务路径由服务组件和组件之间的网络构成.服务组件实现了组合服务的功能,因此,组合服务的性能严重依赖于组合过程中所选择的各服务组件的副本.为了提高服务质量,在建立服务路径时,要避免某些服务副本负载过重而另一些服务副本负载很轻的情况.在服务副本之间实现负载均衡是服务组合的一个重要目标.

图 1 是一个服务覆盖网.图中显示了 4 种服务及其副本的部署情况.用户可以向服务覆盖网中任意一个节点提出服务组合请求,该节点叫做出口节点(exit node).在图 1 中,用户请求组合服务 0 和服务 2,图中建立了一条服务路径,该路径经过了服务 0 和服务 2 两种服务组件,为用户提供新的服务.服务路径中允许存在非操作服务(no-op service).一个节点如果作为非操作服务节点,它只作数据的转发,不进行任何处理.数据沿着服务路径传输,最后到达出口节点,传递给用户.服务组合要解决的问题是根据用户的服务请求,寻找一条服务路径满足用户请求,同时,要使每种服务的各服务副本上的负载尽可能地分布均衡.

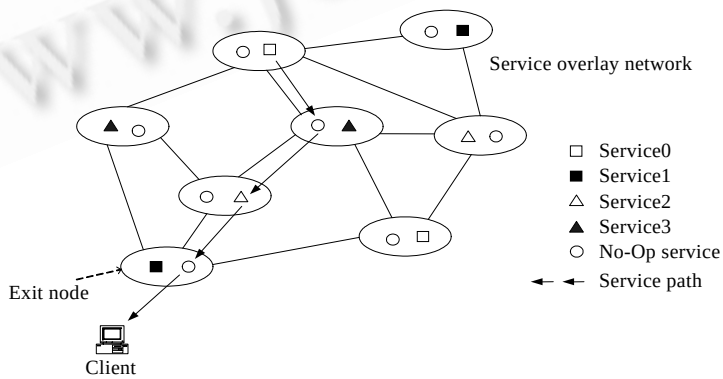

Fig.1 Service composition: An example

图 1 服务组合的例子

2 相关工作

近年来,国内外对 Web 服务器选择的负载均衡问题进行了广泛研究[6−8].但是,服务组合的负载均衡问题具有新的特点:首先,服务器负载均衡只需选择一台服务器提供服务,而服务组合需要选择一组服务副本,服务之间存在某种次序和依赖关系,形成服务路径;其次,服务实例的选择必须是动态的,因为组合服务往往是持续一段时间的软实时应用(比如多媒体流),这一点与接受短时间会话请求(如 Web 页面访问)的 Web 服务器不同.目前,关于服务组合的研究主要有 SAHARA 项目[9],CANS[10],QUEST[11],SpiderNet[12]等.

要在服务覆盖网中找出一条合适的服务路径,就是要找出一条通过特定节点,并以某种测度衡量的、代价最小的路径.目前,人们对 Web 副本站点的选择和机群的负载均衡问题[7,8]进行了深入的研究.但是,服务组合问题要对一组服务进行动态选择,原有的负载均衡算法并不能完全适用于服务组合的情况.QUEST 框架提出了一种在服务覆盖网中寻找满足多种 QoS 约束的服务路径的方法[11],但它主要考虑的是不同服务路径上的 QoS 约束和路径之间的负载均衡问题,并不一定能保证各服务副本的负载均衡.在可编程网络的研究中,文献[13]提出了一种构建通过特定中间节点的路径的算法.该算法从初始覆盖网络拓扑图派生出一个多层转换图,然后在这个转换图上使用 Dijkstra 最短路径算法,求出经过某些特定节点的最短路径.文献[14]提出了利用状态转换图来选择符合网络带宽约束的服务路径的方法.在此基础上,Raman 等人提出了使用 LIAC(least-inverse-available-capacity)测度来构建服务路径的方法[5](简称为 LIAC 算法).LIAC 算法的基本思想是:对于用户请求组合的 k 个服务,将覆盖网络图复制 k+1 次,并在用户所请求的服务之间添加垂直边,形成多层转换图;然后,根据服务器的最大负载和当前负载信息,在垂直边上设置权重.这样,只需使用 Dijkstra 算法在这个带权多层转换图上寻找一条从顶层到底层的最短路径,就是服务负载最轻的一条路径.

但是,LIAC 算法具有以下缺点:(1) 多层转换图需要将覆盖网络图复制 k+1 次,存储空间开销和计算时间开销比较大;(2) LIAC 算法需要知道服务覆盖网的全局拓扑结构、每台服务器的当前负载和最大负载值,在实际网络环境中,这些信息是难以准确测定的;(3) 每个服务器需要维护全局的网络拓扑和负载信息,一个服务器加入或退出系统,需要重新进行计算,开销较大.总之,LIAC 实际上是一种集中式的负载均衡算法,并不适用于大规模网络中服务节点动态加入和退出的情况.

3 LCB:一种分布式自适应服务组合负载均衡算法

针对上述 LIAC 算法的缺点,我们提出一种分布式的自适应服务组合负载均衡算法——基于负载容率的负载均衡(load capacity based algorithm,简称 LCB)算法.它借鉴 IP 路由的思想,使用分布式的服务路由表来进行服务的查找和数据的转发,同时根据各服务节点的负载波动情况,自适应地选择服务副本,达到良好的负载均衡效果.

LCB 算法需要使用两张表:服务路由表和负载容率(load capacity,简称 LC)表.服务路由表用于服务的查找和数据转发,LC 表则用于为服务副本的选择提供决策.下面,首先介绍如何在服务覆盖网中构建分布式的服务路由,然后详细介绍 LCB 负载均衡算法.

3.1 分布式服务路由的建立

在服务覆盖网中,为了实现服务的组合和服务之间的数据转发,每个服务节点要维护一张服务路由表.服务路由表的作用是记录服务之间的最短路径以及该路径上进行数据转发的服务节点.服务路由类似于 Internet 上的 IP 路由.通过服务路由,一个服务节点可以选择一个邻居服务节点将数据转发到目标节点.

服务覆盖网可以用一个无向连通图来表示,图中每一个顶点表示一个服务.服务路由问题本质上是要求出图中任意两个节点之间的最短路径问题.在经典的图论中,可以使用 Dijkstra 算法和 Floyd 算法来计算图中任意两顶点的最短路径,但它们是集中式的算法,不能用于建立分布式路由.Toueg 基于 Floyd 算法提出了一种简单的分布式路由算法[15],可用于建立网络中各服务器的路由.此外,还有许多改进的分布式路由算法,如 Merlin-Segall 算法[15]、Chandy-Misra 算法[16].

我们介绍如何用 Chandy-Misra 分布式路由算法来建立服务路由.假设服务覆盖网中部署有 N 个服务节点,每个服务节点要维护一张长度为 N 的服务路由表.服务路由表的每一项包含该服务节点到其他节点的最短路径的长度和转发邻居的信息.对于节点 u,用 R_u 表示它的服务路由表.$R_u[i].hops$ 表示服务节点 u 到节点 i 的最短路径的长度(用转发的跳数来表示),$R_u[i].next$ 表示服务节点 u 到节点 i 的转发邻居,u 到 i 的数据都经过该邻居节点转发.设 V 为服务覆盖网中所有服务节点的集合,$Neigh_u$ 为节点 u 的所有邻居节点的集合.使用 Chandy-Misra 算法计算到达目的节点 v_0 的服务路由的过程如下:

Initialization:
begin
  forall $v \in V$ **do**
    If $v=u$ **then begin** $R_u[v].hops:=0$; $R_u[v].next:=$null **end**
      else begin $R_u[v].hops:=\infty$; $R_u[v].next:=$null **end**
end

For node v_0 only:
begin
  forall $\omega \in Neigh_u$ **do** send $message(v_0,0)$ to ω;
end

Processing a $message(v_0,d)$ from neighbor ω by u:
begin
  receive $message(v_0,d)$ from ω;
  if $d+1<R_u[v_0].hops$ **then**
    begin
      $R_u[v_0].hops:=d+1$;
      $R_u[v_0].next:=\omega$;
      forall $x \in Neigh_u$ **do if** $x \neq \omega$ **then** send $message(v_0,R_u[v_0].hops)$ to x;
    end
end

Chandy-Misra 算法利用了扩散计算(diffusing computation),由一个节点初始化,其他节点一接到消息后就加入.对于服务覆盖网中的每个节点,都要作为目标节点,使用上述算法计算一次,才能形成完整的服务路由表.Chandy-Misra 算法中,每一信道交换消息复杂度为 $O(N^2)$,算法消息复杂度为 $O(N^2 \cdot |E|)$,其中,$|E|$表示覆盖网络中边的总数[15].

这里,Chandy-Misra 算法是根据最小跳数原则来建立服务路由的.LCB 负载均衡算法首先根据各服务节点的负载状况选择合适的服务副本,然后使用服务路由表构建一条较短的服务路径,从而减少用户访问延迟.但在实际的网络环境中,最短的服务路径并不一定是最佳的服务路径,这是因为我们还未考虑到各信道的带宽使用状况、消息的数量、信道的延迟等因素.如何构造更高效的 QoS 路由机制,是我们将来要进一步研究的问题.

3.2 LCB负载均衡算法

服务路由只是提供了服务的查找和数据转发的机制,服务组合负载均衡的一个更核心的问题是如何选择服务副本,以保证各副本上的负载分布比较均匀.这里,我们提出一种自适应的负载均衡算法——LCB 算法.LCB 算法的基本思想是:对每个服务节点,使用一个 LC 测度来表征该节点的当前负载状况,根据 LC 的值来选择一条合适的服务路径.LC 测度的值根据服务器的负载波动情况不断进行自适应的调整,为服务副本的选择提供决策.

为了介绍 LCB 算法,我们首先定义服务器的负载波动率Δ.一台服务器的负载波动率定义为两次接入用户请求时,服务器的负载变化对时间变化的比值.假设服务器 u 接入一个用户请求时,时间为 T_1,负载为 $L_u(T_1)$,而该服务器上一次接入用户请求时,时间为 T_2,负载为 $L_u(T_2)$,则负载波动率定义为

$$\Delta=\frac{L_u(T_1)-L_u(T_2)}{T_1-T_2} \tag{1}$$

事实上,如果在 T_2 到 T_1 这段时间内没有服务结束,则服务器在本次接入请求和上一次接入请求之间负载增量必然为 1.考虑到在 T_2 到 T_1 这段时间内会有已接入的服务运行结束,因此,这段时间内负载的变化量等于 1 减去这段时间内结束的服务数量,即 $L_u(T_1)-L_u(T_2)=1-expired(T_2,T_1)$.其中,$expired(T_2,T_1)$表示在 T_2 到 T_1 时间段内结束的服务数量(负载增量可为负值,表明这段时间内结束的服务数量大于接入的服务数量).式(1)可以改写如下:

$$\Delta=\frac{1-expired(T_2,T_1)}{T_1-T_2} \tag{2}$$

从式(2)可见,负载波动率Δ的计算并不需要知道服务器的最大负载信息和当前负载信息,只需要知道在两次用户接入的时间间隔内结束的服务数量.对于服务器来说,它的最大负载量一般很难预先估算,当前负载量也很难准确测量,但一段时间内结束的服务数量却是很容易计算的.

每个服务节点用一个负载容率测度来估算它的相对负载状况.负载容率 LC 是表示当前服务器还能容纳多少负载量的一个测度,取值范围在 0~1 之间.LC 越大,表示服务器的负载容纳能力越大,说明服务器当前负载还比较轻,可以接纳更多的客户;反之,说明服务器负载较重,应该减少接纳的客户.LC 的值是根据服务器的负载波动情况不断进行自适应调整的.我们设 LC 的初始值为 1,表示开始时服务节点负载为空,每接入一个客户,需要对 LC 的值进行调整,LC 的更新公式为

$$LC=f((1-\alpha\Delta)LC) \tag{3}$$

其中,f 是一个门限函数,用于将 LC 的值限定在 0~1 之间.$f(x)$定义为

$$f(x)=\begin{cases}0, & x\le 0\\ x, & 0<x<1\\ 1, & x\ge 1\end{cases} \tag{4}$$

α是一个取值在 0~1 之间的数,称为波动因子,它反映了随负载波动对 LC 进行调整的幅度.由式(3)可见,当服务节点的负载以恒定的速率增加时,Δ恒定,LC 是一个指数衰减函数,它的值由 1 逐渐衰减,趋于 0.事实上,负载波动率Δ不是恒定的.从式(2)可见,两次接入客户的时间间隔越短,Δ就越大,LC 也就衰减得越快.但是,如果在一段时间内结束的服务数量大于接入的客户数量,Δ的值就是负值,表明服务器有了空余的服务能力,这样,LC 的值就会由小而大地递增.α是控制衰减和递增速率的参数,α取不同的值有不同的负载均衡的效果.我们将在后面的实验中详细讨论α取值对于负载均衡的影响.

LCB 负载均衡算法要在每个服务节点上维护两张表:服务路由表和 LC 表.服务路由表提供从本服务器到达其他服务副本的路由信息.LC 表则记录各服务副本的 LC 值,服务路径的选择就是根据 LC 表来进行决策的.假设服务器 u 接收到用户的服务组合请求 $Req=(S_1,S_2,\ldots,S_k)$,令 $Replica(S)$表示服务 S 的所有副本的集合,$LC_u(S)$表示节点 u 中服务 S 的 LC 测度值,LCB 算法可以如下来描述(其中,*request_forward*(*u*,*S*,*Req*)的功能是查找服务路由表,将请求 *Req* 由源节点 *u* 转发给目标节点 *S*):

1. 接入一个客户请求
   $Req:=(S_1,S_2,\ldots,S_k)$;
2. 对于用户请求的第 1 个服务,选择服务副本集合中 LC 值最大的服务副本 S 来提供服务
   max_*lc*:=−1; *S*:=null;
   forall $x\in Replica(S_1)$ **do**
   begin
   if $LC_u(x)>$max_*lc* **then begin** max_*lc*:=$LC_u(x)$; *S*:=*x* **end**
   end
   if *S*=null **exit**
3. 使用式(2)~式(4)更新节点 u 的 LC 表:
   $LC_u(S):=MAX(0,MIN(1,(1-\alpha\times\Delta_s)\times LC_u(S)))$; /*$\Delta_s$为节点 S 的负载波动率*/
4. 查询服务路由表,将客户的其他服务请求转发给 S 所在的服务器,由 S 选择下一个服务副本,最终形成服务路径
   $Req:=(S_2,\ldots,S_k)$;
   request_forward(*u*,*S*,*Req*);

LCB 算法克服了上面提到的 LIAC 算法的一些缺点:(1) 在 LIAC 算法中,构建多层转换图需要将覆盖网络图复制 k+1 次,再用 Dijkstra 算法寻找最短路径.LCB 算法只需对覆盖网络图应用服务路由算法,存储空间开销和计算时间开销都比 LIAC 算法小得多.(2) 在 LIAC 算法中,构造多层转换图需要知道服务覆盖网的全局拓扑结构,LIAC 测度的计算则需要知道各服务节点的当前负载和最大负载值,但是,服务器的最大负载量往往难以预先估算,而且服务器的当前负载也是难以测定的.LCB 算法只需要知道服务节点的负载波动情况,也就是服务器在一段时间内结束的服务数量,这对于服务器来说是可以准确测量的.(3) 在 LIAC 算法中,一个服务器加入或退出系统,会导致整个多层转换图的拓扑结构发生改变,需要重新计算,开销较大.LCB 算法使用分布式的服务路由,一个节点的状态发生改变,只需对分布式路由表的部分项进行修改,并重新计算 LC 表中该节点的测度值即可.(4) LIAC 实质上是一种集中式的负载均衡算法,而 LBC 是一种分布式的负载均衡算法,更适用于大规模网络环境下服务节点动态加入和退出的情况.

4 模拟实验

4.1 实验的建立

我们通过模拟实验来验证 LCB 算法的负载均衡性能.我们使用网络仿真工具 GT-ITM[17]生成物理网络拓扑,并在生成的物理网络上随机选择 100 个节点作为服务覆盖网的节点,在这些节点之间定义了 117 条链接.我们设置了 10 个不同的服务 S0~S9,每个服务有 4 个副本数.我们在服务覆盖网的 100 个节点上随机选择 40 个节点来部署这些服务副本.每个节点的最大负载量设为 1 500.假设每使用一次服务副本消耗 1 个单位的负载.我们设置每个会话的持续时间为 70s~90s,会话请求的到达速率设为 20 次/秒.每个会话随机包含 2 个服务.一次实验处理 10 000 个会话.每个会话到达后,可以使用不同的方式选择出口节点.根据不同的实验目的,我们将使用随机和非随机两种方式选择出口节点.

4.2 LCB算法的负载均衡效果

假设用户的请求是随机地向服务覆盖网的 100 个节点发送的,我们令波动因子α=0.1,在上述实验环境下,运行 LCB 算法.表 1 显示了当 10 000 个会话全部建立后,每种服务的 4 个服务副本所接入的会话总数.从表中可以看出,每种服务的 4 个副本所接入的负载总量相差不太大,表明使用 LCB 算法,各服务副本在一个时间段内所接入的负载总量是均衡的.

Table 1 Load distribution across service replicas

表 **1** 服务副本的负载分布

Replica number	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
0	452	493	503	476	474	473	502	496	491	473
1	527	536	540	510	488	513	539	516	493	510
2	496	510	501	485	477	499	493	504	545	506
3	487	501	513	533	470	448	492	511	505	519

为了研究各服务副本的负载随时间变化的情况,我们的实验每隔 15 秒测量一次各服务副本的负载量.我们通过实验比较 LCB 算法和 LIAC 算法的性能.图 2 和图 3 显示了分别使用这两种算法时,一个服务 S0 的 4 个副本的负载量随时间变化的曲线(其他服务的副本负载变化情况也类似).图中 4 条曲线相互越靠近,表明分布在 4 个服务副本上的负载越均衡.从图 3 可见,在很多情况下,4 条曲线的点几乎重合在一起,这表明 LIAC 算法具有良好的负载均衡效果.图 2 的 4 条曲线则相互分得开一些,表明 LCB 算法的负载均衡效果不如 LIAC 算法.造成这种差异的原因是,LIAC 算法使用了全局的准实时负载信息,进行集中式的计算,因此大部分时间服务副本的负载分布很均衡,但 LIAC 算法是集中式的,计算开销比较大,而且算法所需的信息在真实网络环境中也很难获取.LCB 算法是一种分布式的算法,它使用的 LC 测度是一个根据服务器负载波动估算的测度,并不是真正的服务器负载信息.同时,LC 测度是自适应调整的,当几个服务副本之间的负载差异比较大时,LC 值会自动调整,使各副本的负载趋于均匀.从图 2 可见,虽然在有些点,4 条曲线的距离比较大,但通过 LC 测度的调整会逐渐趋于

靠近.在实际的网络应用环境中,并不要求各服务副本的负载在任何时刻都严格保持均衡,各服务副本之间的负载只要相差在一定的范围内,都是可以接受的.从图 2 来看,大部分情况下 4 个服务副本之间的负载相差不超过 20.因此,LCB 算法可以适用于真实网络环境中的服务组合应用.

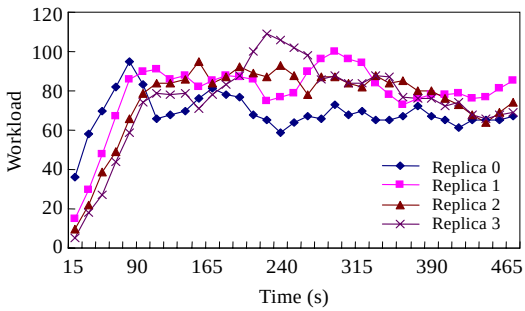

Fig.2 Load variation with LCB algorithm (α=0.1)

图 2 LCB 算法负载变化(α=0.1)

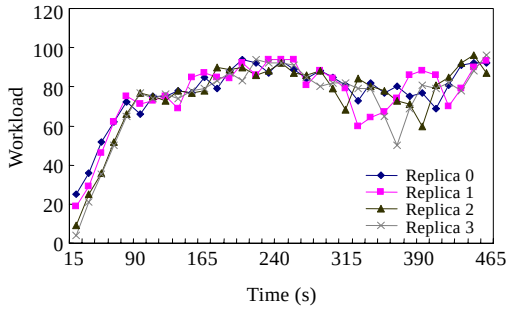

Fig.3 Load variation with LIAC algorithm

图 3 LIAC 算法负载变化

4.3 波动因子α对负载均衡的影响

图 2 显示了α=0.1 时 LCB 算法的负载均衡效果.事实上,波动因子α的取值对负载均衡有着重要的影响,适当地选取α的值可以获得更好的负载均衡效果.因此,我们需要研究波动因子α的变化与负载均衡的关系.

为了衡量负载均衡的效果,我们引入负载距离 LD 来表示 4 条负载曲线之间相互靠近的程度.在任一时刻,LD 定义为 4 条负载曲线上各点两两距离的总和.4 条负载曲线相互越靠近,LD 越小,表明负载越均衡.

假设σ为负载观测点的集合,sum 为观测点的总数.设 P 为一个观测点,则在 P 点处观测到 4 个服务副本的负载值 L_0,L_1,L_2,L_3.在 P 点处的负载距离 LD_P 定义为

$$LD_P=\sum_{i=0}^{2}\sum_{j=i+1}^{3}\left|L_i-L_j\right| \tag{5}$$

我们用平均负载距离来衡量整体的负载均衡效果.平均负载距离定义为在所有观测点上的负载距离的平均值.平均负载距离的计算公式为

$$\overline{LD}=\frac{\sum_{P\in\sigma}LD_P}{sum} \tag{6}$$

为了考察波动因子α对平均负载距离 $\overline{LD}$ 的影响,我们让α取值 0.1,0.2,…,0.9,对每个α,计算相应的平均负载距离,可以画成如图 4 所示的关系曲线.可见,曲线在α=0.3 和α=0.6 附近各有一个极小值.这表明α取值在 0.3 或 0.6 附近时,整体的负载均衡效果最好.在后面的实验中,我们令α的取值为 0.3.

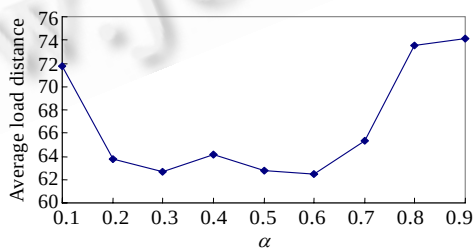

Fig.4 Average load distance variation with fluctuating factor α

图 4 平均负载距离随波动因子α的变化

4.4 LCB算法对于非均匀用户访问的负载均衡效果

前面讨论了用户访问均匀分布在各服务器的负载均衡情况.但在实际中,用户对各服务器的访问请求往往不是均匀的.研究表明,用户对 Web 页面的访问服从 Zipf-like 分布[18],即大部分用户请求集中在小部分的“热点”

数据上.因此,有必要研究当用户请求符合 Zipf-like 分布时,使用 LCB 算法的负载均衡效果.

为了产生符合 Zipf-like 分布的用户访问序列,我们假设μ为用户请求的对象总数,i表示第i个热门的对象的 ID,$P_\mu(i)$表示该对象被请求的概率,则 $P_\mu(i)=\frac{\Omega}{i}$,其中 $\Omega=\left(\sum_{i=1}^{\mu}\frac{1}{i}\right)^{-1}$.

在实验中,我们令μ=100,产生 10 000 个符合 Zipf-like 分布的用户访问序列.取α=0.3,重做第 4.2 节的实验,可以得到如图 5 所示的负载曲线图.由图可见,4 条负载曲线比较接近,各服务副本的负载在大部分时间相差不超过 20.这说明,对于非均匀的用户访问请求,虽然大部分访问请求集中在小部分节点上,但是使用 LCB 算法,通过对 LC 测度的自适应调整,仍然可以使负载均衡地分布在服务的各副本上.

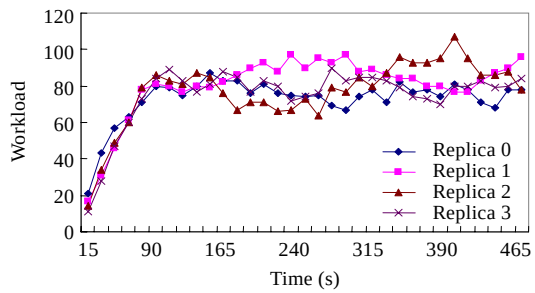

Fig.5 Load variation with uneven user visits (α=0.3)

图 5 非均匀用户访问的负载变化(α=0.3)

4.5 服务路径长度的影响

在以上讨论中,LC 测度完全是根据负载波动信息进行负载均衡的,并没有考虑服务路径的长度.因此,有可能根据 LC 测度选择出来的服务路径虽然可以保证各服务副本的负载相对均匀,但很长,这样,服务组合要经过服务覆盖网的多跳,造成客户访问的延时较长,同时会浪费网络的带宽资源,使网络的整体服务质量下降.

为了研究 LCB 算法中服务路径长度对性能的影响,我们分别使用 LC 测度和最短路径测度来进行服务路径选择,图 6 给出了使用这两种测度选择出来的服务路径长度的累积分布(cumulative distribution function,简称 CDF)图.从图中两条曲线的比较可见,当使用最短路径测度时,80%的服务路径长度在 10 跳之内;而当使用 LC 测度时,只有不到 40%的服务路径在 10 跳之内.由于 LC 测度没有考虑服务路径长度因素,故选择出来的服务路径长度并不理想.

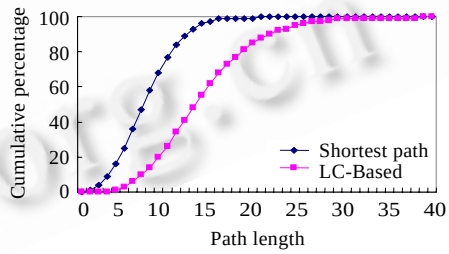

Fig.6 Path length CDFs, with α=0.3

图 6 路径长度累积分布函数图(α=0.3)

我们希望在保证负载均衡的同时选择较短的服务路径,以减少用户访问延迟.因此,我们考虑对 LCB 算法进行改进,在负载容率 LC 中加入路径长度的影响.假设用两个节点之间的跳数(hops)来表示两节点之间的路径长度,我们改进的副本选择原则是,当两个服务副本的负载比较接近时,选择服务路径较短的一个副本为用户提供服务.这样,LC 的更新公式可以改写如下:

$$LC=f\left((1-\alpha\Delta)LC+\beta\frac{1}{hops}\right) \tag{7}$$

其中,β是路径因子,它表示路径长度因素对于服务路径选择的影响.β越大,表明路径长度的影响越大,越趋近于最短路径.但β的值不能选得太大,β越大,负载波动的相对影响越小,这样,LC 测度也就越不能反映服务副本的负载状况,从而造成负载不均衡的程度增大.

图 7 显示了随着β的增大,平均负载距离的变化情况.可见,β越大,平均负载距离越大,造成各服务副本之间

的负载越不均匀.为了保证负载均衡,β的取值不能过大.例如,我们希望服务的任意两个副本之间的负载距离不超过 20,那么平均负载距离 $\overline{LD}$ 应控制在小于 120 的范围内.由图可见,β取值应在 0~0.2 之间.

为了研究β的取值对于所选择的服务路径长度的影响,我们让β分别取值 0,0.05,0.1,0.15,0.2,研究这几种情况下使用 LCB 算法建立的服务路径长度,并与最短路径进行比较.图 10 显示了这几种情况下服务路径的 CDF 图.可见,随着β的增大,CDF 曲线越来越接近于最短服务路径.但是,从负载均衡的角度,我们希望β尽可能小,这样负载会更均衡.两者要取得折衷.从图 8 可以看出,当β>0.1 时,β取值为 0.1,0.15,0.2 这 3 条 CDF 曲线已经比较接近,表明当β>0.1 时,β的增大对于缩短服务路径长度的影响已经很小.因此,我们认为β的值取为 0.1 比较适当,这时可以获得较短的服务路径,同时保证负载比较均衡.

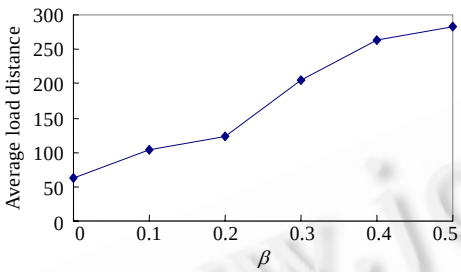

Fig.7 Average load distance variation with path factor β (α=0.3)

图 7 平均负载距离随路径因子β的变化(α=0.3)

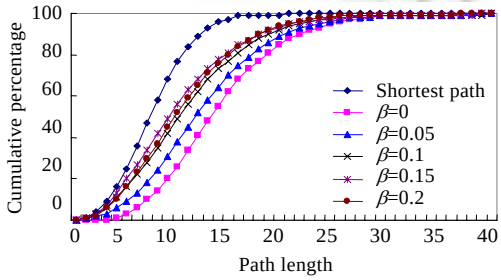

Fig.8 Comparison of path length CDFs under different β value (α=0.3)

图 8 路径长度累积分布函数对不同β取值的比较(α=0.3)

5 总 结

服务路径的选择和负载均衡是服务组合研究的一个重要问题.我们提出了一种自适应的服务组合负载均衡算法 LCB 算法,它使用负载容率(LC)测度来衡量各服务副本的负载状况,根据 LC 测度来选择一条合适的服务路径,可以获得较好的负载均衡效果.

与现有的服务组合负载均衡算法相比,LCB 算法的创新之处在于:(1) 基于多层转换图的算法时间和空间开销较大,LCB 算法提出使用服务路由来查找服务和转发数据,服务路由表可以由服务覆盖网的各节点进行分布式计算,计算的时间和空间开销都少得多.(2) 现有服务组合负载均衡算法大部分需要知道服务器的最大负载和当前负载量信息,而这些数值是很难准确测量的,LCB 算法需要的信息量更少.LCB 算法提出根据负载波动来进行服务副本的选择,负载波动的值是很容易准确计算的.(3) 基于多层转换图的算法需要服务覆盖网的全局信息,本质上是一种集中式算法.LCB 算法是一种分布式的负载均衡算法,当服务节点发生改变时,只需对服务路由表和 LC 表的部分数据进行更新,可扩展性更好.因此,LCB 算法更适用于分布式环境下动态服务副本的组合.

References:

[1] Yang J, Papazoglou M. Web component: A substrate for web service reuse and composition. In: Pidduck AB, Mylopoulos J, Woo CC, Ozsu MT, eds. Proc. of the 14th Conf. on Advanced Information Systems Engineering (CaiSE 2002). LNCS 2348, Toronto: Springer-Verlag, 2002. 21−36.

[2] Yue K, Wang XL, Zhou AY. Underlying techniques for Web services: A survey. Journal of Software, 2004,15(3):428−442 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/428.htm

[3] Satyanarayanan M. Pervasive computing: Vision and challenges. IEEE Personal Communications, 2001,6(8):10−17.

[4] Gu X, Nahrstedt K. Dynamic QoS-Aware multimedia service configuration in ubiquitous computing environments. In: Proc. of the 22nd Int'l Conf. on Distributed Computing Systems (ICDCS 2002). Vienna: IEEE Computer Society, 2002. 311−318.

[5] Raman B, Katz RH. Load balancing and stability issues in algorithms for service composition. In: Proc. of the 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2003). San Franciso: IEEE Communications Society, 2003. 1477−1487.

[6] Cardellini V, Colajanni M, Yu PS. Dynamic load balancing on Web-server systems. IEEE Internet Computing, 1999,3(3):28−39.
[7] Shan Z, Lin C, Marinescu DC, Yang Y, Modeling and performance analysis of QoS-aware load balancing of web-server clusters. Computer Networks, 2002,40(2):235−256.
[8] Guo CC, Yan PL. A dynamic load-balancing algorithm for heterogeneous Web server cluster. Chinese Journal of Computers, 2005, 28(2):179−184 (in Chinese with English abstract).
[9] Raman B, Agarwal S, Chen Y, Caesar M, Cui WD, Johansson P, Lai K, Kavian T, Machiraju S, Mao ZM, Porter G, Roscoe T, Seshadri M, Shih J, Sklower K, Subramanian L, Suzuki T, Zhuang S, Joseph AD, Katz RH, Stoica I. The SAHARA model for service composition across multiple providers. In: Mattern F, Naghshineh M, eds. Proc. of the Int'l Conf. on Pervasive Computing (Pervasive 2002). LNCS 2414, Zürich: Springer-Verlag, 2002. 1−14.
[10] Fu X, Shi W, Akkerman A, Karamcheti V. CANS: Compassable, adaptive network services infrastructure. In: Anderson T, ed. Proc. of the 3rd USENIX Symp. on Internet Technologies and Systems. San Francisco: USENIX, 2001. 135−146.
[11] Gu X, Nahrstedt K, Chang RN, Ward C. QoS-Assured service composition in managed service overlay networks. In: Proc. of the 23rd Int'l Conf. on Distributed Computing Systems (ICDCS 2003). Providence: IEEE Computer Society, 2003. 194−203.
[12] Gu X, Nahrstedt K, Yu B. SpiderNet: An integrated peer-to-peer service composition framework. In: Proc. of the Int'l Symp. on High-Performance Distributed Computing (HPDC-13). Honolulu: IEEE Computer Society, 2004. 110−119.
[13] Choi S, Turner J, Wolf T. Configuring sessions in programmable networks. Computer Networks, 2003,41(2):269−284.
[14] Ma Q, Steenkiste P. On path selection for traffic with bandwidth guarantees. In: Proc. of the Int'l Conf. on Network Protocols (ICNP'97). Atlanta: IEEE Computer Society, 1997. 191−202.
[15] Tel G. Introduction to Distributed Algorithms. 2nd ed., Beijing: China Machine Press, 2004. 61−92 (in Chinese).
[16] Chandy KM, Misra J. Distributed computation on graphs: Shortest path algorithms. Communications of the ACM, 1982,25(11): 833−837.
[17] Zegura EW, Calvert K, Bhattacharjee S. How to model an Internet work. In: Proc. of the 15th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'96). San Francisco: IEEE Communications Society, 1996. 594−602.
[18] Breslau L, Cao P, Fan L, Phillips G, Shenker S. Web caching and Zipf-Like distributions: Evidence and implications. In: Proc. of the 18th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'99). New York: IEEE Communications Society, 1999. 126−134.

附中文参考文献:

[2] 岳昆,王晓玲,周傲英.Web 服务核心支撑技术:研究综述.软件学报,2004,15(3):428−442. http://www.jos.org.cn/1000-9825/15/428.htm
[8] 郭成城,晏蒲柳.一种异构 Web 服务器集群动态负载均衡算法.计算机学报,2005,28(2):179−184.
[15] Tel G,著;霍红卫,译.分布式算法导论.第 2 版.北京:中国:机械工业出版社,2004.61−92.


李文中(1979－),男,广西平南人,博士生,主要研究领域为分布式计算,并行处理.


陆桑璐(1970－),女,博士,教授,博士生导师,主要研究领域为分布并行处理,高性能计算.


郭胜(1978－),男,硕士,主要研究领域为分布式计算,并行处理.


陈道蓄(1947－)男,教授,博士生导师,CCF 高级会员,主要研究领域为分布式计算,软件工程.


许平(1978－),女,讲师,主要研究领域为分布式计算,并行处理.