ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.5, May 2008, pp.1212−1223
DOI: 10.3724/SP.J.1001.2008.01212

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

自适应组件副本选择模型及算法*

左 林[1,2,3+], 刘绍华[1], 魏 峻[1], 冯玉琳[1,2], 范国闯[1]

[1](中国科学院 软件研究所 软件工程技术中心,北京 100190)

[2](中国科学院 软件研究所 计算机科学重点实验室,北京 100190)

[3](中国科学院 研究生院,北京 100049)

Adaptive Component Replica Selection Model and Algorithms

ZUO Lin[1,2,3+], LIU Shao-Hua[1], WEI Jun[1], FENG Yu-Lin[1,2], FAN Guo-Chuang[1]

[1](Technology Center of Software Engineering, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: martin_zl@otcaix.iscas.ac.cn

Zuo L, Liu SH, Wei J, Feng YL, Fan GC. Adaptive component replica selection model and algorithms. *Journal of Software*, 2008,19(5):1212−1223. http://www.jos.org.cn/1000-9825/19/1212.htm

Abstract: This paper proposes a domain based adaptive replica selecting model named DARSM (domain based adaptive replica selection model), in which component replicas are organized into strong consistency domain and weak consistency domain, and a schema based on consistency window is utilized to synchronize states between theses domains. Accordingly, a partition-balanced based adaptive replica selection algorithm PWARS (partition-weighted based adaptive replica selection) could be built on dynamic performance metric information to select the appropriate replica set that satisfies specific QoS constraints. A consistency window adaptive reconfiguration algorithm CWAR (consistency window adaptive reconfiguration) is presented to adapt to the dynamic change of consistency constraints. In this algorithm, a probability model built on the base of current consistency constraints distribution is used to dynamically adjust the configuration of consistency window. As a result, the inconsistency of each replica is controlled adaptively. This approach has been implemented in OnceAS, and experimental results demonstrate that it can effectively enhance the performance of replica selection.

Key words: replica selection; fault-tolerant; adaptive; QoS (quality of service); middleware

摘 要: 提出了一个基于域的自适应副本选择模型 DARSM(domain based adaptive replica selection model).该模型将组件副本划分为强一致性域和弱一致性域,域间通过一致性窗口机制进行状态同步.基于 DARSM 模型,给出了一种基于分区加权的自适应副本选择算法 PWARS(partition-weighted based adaptive replica selection).该算法利用动态

* Supported by the National Natural Science Foundation of China under Grant No.60573126 (国家自然科学基金); the National Basic Research Program of China under Grant No.2002CB312005 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01Z19B, 2007AA01Z134 (国家高技术研究发展计划(863)); the National Key Technology R&D Program of China under Grant No.2006BAH02A01 (国家科技支撑研究发展项目)

Received 2006-07-13; Accepted 2007-01-04

性能度量信息来选择满足时间约束和一致性约束的组件副本集合.为了适应请求一致性约束的动态变化,还提出了一致性窗口自适应重配算法 CWAR(consistency window adaptive reconfiguration).通过引入的一个一致性约束的可能性模型,该算法动态地对一致性窗口进行重配,从而实现了副本一致性的自适应控制.通过在 OnceAS 应用服务器集群中的原型实验及性能评价,表明该方法能够明显地提高副本选择的性能.

关键词: 副本选择;容错;自适应;服务质量;中间件

中图法分类号: TP393 **文献标识码:** A

网络分布式环境下,应用服务器已经成为支撑分布式应用的基础中间件平台[1],如 CORBA[2]和 J2EE[3],为创建、部署、运行、集成和管理分布式应用提供了一系列运行时服务(如消息、事务、安全等).当前,主流的应用服务器[4]往往通过组件复制技术来提高应用的可靠性、可用性和伸缩性.在这种组件复制的环境下,客户往往要求应用提供的服务满足一定的 QoS(quality of service)约束,如时间约束和一致性约束.

目前,部署在应用服务器上的许多应用往往要求客户的请求在一定时间内得到处理,这就需要解决如何为客户请求选择满足其时间约束的组件副本问题.已有的副本选择算法如轮转(round-robin)算法、随机(random assignment)算法、均衡算法(proportional algorithm)、贪婪(greedy selection)算法[5]、加权选择(weighted selection)算法[6]等都是仅仅选择唯一的副本来处理客户的请求,当副本出现失效时,则只能通过重传请求的方式继续请求的处理.显然,这种时间冗余的方式会延长请求的处理时间,从而导致时间约束得不到满足.

此外,当前的应用往往是有状态的,因此,除了时间约束以外,选择副本时还需要考虑状态的一致性约束,而当前的选择算法都没有考虑副本状态的一致性约束问题.当出现大量的客户并发访问时,为了缩短请求的响应时间以满足较为严格的时间约束,副本之间常常采取弱一致性的状态同步模型.在这种模型中,允许各个副本的状态存在临时的不一致,但是确保它们最终会达到一致.由于不需要等到副本之间完成状态同步,从而缩短了请求的响应时间.这不同于强一致性模型,它始终保证各个副本具有相同的状态,因此,请求的处理必须在副本之间的同步操作完成之后才能进行.虽然弱一致性模型通过牺牲副本状态的一致性程度加速了请求的处理,但是不同的客户对于这种松散程度的需求是不同的.也就是说,为了使不同客户请求的响应结果有意义,处理请求的副本必须满足一定的状态一致性约束.此外,由于客户的一致性约束是动态变化的,因此还需要一种自适应的机制来动态控制副本之间的状态同步的周期.当客户请求的一致性约束较低时,则应延长同步周期来降低更新发布的速度,以节约系统开销来服务更多的请求;当客户请求的一致性约束较高时,则应缩短同步周期来提高更新发布的速度,以避免由于等待同步操作而延长请求的响应时间,导致请求的时间约束得不到满足.

为了在动态变化的应用 QoS 需求(时间约束和一致性约束)和系统运行环境下能够合理地选择组件副本,本文提出了一个基于域的自适应副本选择模型 DARSM(domain based adaptive replica selection model).该模型将副本组划分为强一致性域和弱一致性域,域间通过一致性窗口机制进行状态同步.在此模型基础上,给出了一种基于分区加权的自适应副本选择算法 PWARS(partition-weighted based adaptive replica selection).该算法利用动态性能度量信息来选择满足约束的组件副本集合.此外,为了适应应用一致性约束分布的动态变化,提出了一致性窗口自适应重配算法 CWAR(consistency window adaptive reconfiguration).该算法通过一致性约束的可能性模型对一致性窗口进行动态重配,从而实现对各个副本的一致性进行自适应的调整.我们在 OnceAS 应用服务器[7]上进行了原型实验,结果表明,该方法能够明显地提高组件副本选择的性能.

本文第 1 节对相关工作进行比较.第 2 节详细介绍基于域的自适应副本选择模型 DARSM 的结构.第 3 节介绍基于分区加权的自适应副本选择算法 PWARS.第 4 节介绍一致性窗口自适应重配算法 CWAR.第 5 节给出 DARSM 模型的性能评价.最后是对本文工作的总结.

1 相关工作比较

副本选择是分布式系统研究的热点问题,目前的副本选择算法可以分为静态选择和动态选择两类.静态方法按照固定的预定义规则为客户请求选择副本.轮转算法将副本组织成一个环,按顺序进行副本选择.随机算法

按照等可能性随机地选择副本.在均衡算法中,各个副本的处理能力被预先评估,然后根据处理能力的相对大小分配其相应的比例值,再根据各个副本的比例值对请求进行分配.然而,比例值的大小不可以在运行时进行修改.这些静态方法由于没有考虑当前的系统负载和运行环境,很难适用于客户请求带有特定 QoS 需求的情况.与静态方法不同,动态方法则是以运行时的某些度量信息作为根据来进行副本选择.最常见的是贪婪算法[5].该算法对预先定义的性能指标进行动态度量,每次总是选择性能最优的副本.这样会出现 Thundering Herd 问题[8],即当发现一个副本性能最优时,在很短时间内的所有请求都会选中这个副本.加权选择算法[6]解决了这个问题.该算法根据当前性能信息为每个副本动态地分配权值,然后根据权值代表的可能性大小进行副本选择.以上这些算法都没有考虑副本的一致性约束问题.文献[9]基于容错 CORBA 中间件 Aqua[10]提出了一种同时支持响应时间和一致性约束的动态选择算法,通过域的方式对副本进行组织,每次选择时都要根据历史响应时间计算分布函数,引入了较大的性能开销,并且没有解决一致性约束动态变化的问题.

在副本一致性研究方面,Web 缓存中使用的 Time-To-Live(TTL)[11]是最简单的从时间上限制副本一致性的方法.文献[12]中提出的 Timed Consistency 模型和文献[13]中提出的 Delta Consistency 模型也是从时间的维度来对一致性进行限制,即要求在一个副本上的写操作的效果必须在一定的时间范围 t 内被其他副本看到.文献[14]对一致性程度的度量进行了探讨,从尝试性写的数量、数值的差额和时间 3 个维度来对一致性进行度量,并且设计和实现了中间件 TACT 来支持副本之间一致性程度的调整.以上这些工作都没有考虑如何为具有一致性约束的客户请求进行副本选择的问题,同时也没有考虑如何根据应用需求的变化适应性地调整副本之间的不一致性的问题.

当前,主流 J2EE 应用服务器如 IBM Websphere[15],BEA Weblogic[16]和 JBOSS[17]等在进行副本选择时主要还是采用 Round-Robin 和 Greedy 算法,缺乏对客户 QoS 需求尤其是一致性约束的支持.

与已有的研究工作相比,本文的贡献在于:(1) 提出了一种基于域的自适应副本选择模型,副本域之间利用一致性窗口机制进行状态同步;(2) 在副本选择的同时考虑了客户请求的时间约束和一致性约束,能够根据当前副本的实际运行状态自适应地选择满足约束的副本集合;(3) 一致性窗口的自适应重配机制能够根据请求的一致性约束的动态变化,自适应地调整副本之间状态同步的速率,从而实现副本之间不一致程度的动态调整.

2 基于域的自适应副本选择模型DARSM

本文的自适应副本选择模型基于常见的本地网络互联的分布式系统模型.在这个系统模型中,一个服务(service)由一组相同的组件(例如 EJB 组件)实现,它们分布在相同或不同的主机(host)上.这些组件及其所驻留的容器称为副本(replica),副本的集合称为副本组(replica group,简称 RG).客户发送请求来调用所需的服务,这些请求被分配到不同的副本进行处理,并且请求的处理必须遵守客户指定的 QoS 约束.每个副本都存在一个 FIFO 队列(request queue,简称 RQ)来接收客户请求.基于域的自适应副本选择模型 DARSM 如图 1 所示.为了更加清楚和严格地描述该模型,我们给出如下定义:

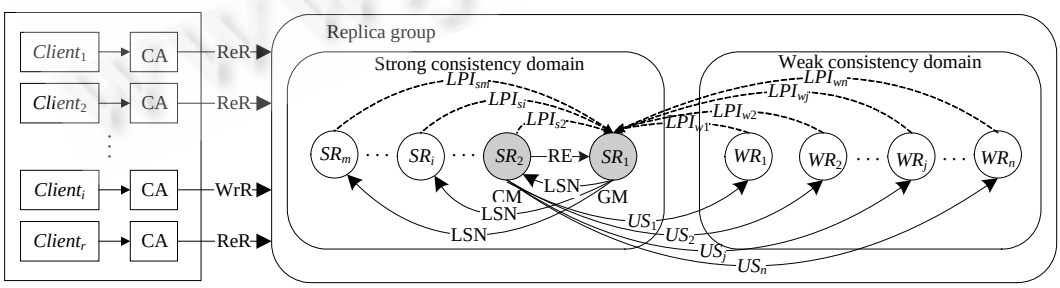


Fig.1 Domain based adaptive replica selection model

图 1 基于域的自适应副本选择模型

定义 1(请求及类型). 客户发出请求 r 的时刻记为 $ST(r)$,得到响应的时刻记为 $ET(r)$.请求的类型记为 $type(r)$,

$type(r)\in\{read\text{-}only,write\text{-}only,write\text{-}read\}$.其中,*read-only* 类型的请求仅仅读取副本的状态而不对其进行任何改变,将这类请求记为 ReR.*write-only* 和 *write-read* 类型的请求都会改变副本的状态,将这类请求记为 WrR.ReR 请求得到的副本 x 的版本记为 $V(x)$,副本当前的最新版本记为 V_{latest},x 当前的陈旧度定义为 $STALE(x)$,$STALE(x)=V_{latest}-V(x)$.每个更新操作关联一个基于逻辑时钟[18]的逻辑序列号(logic sequence number,简称 LSN).它是一个自然数,按照从小到大的顺序创建,即越早的更新其 LSN 越小,越晚的更新其 LSN 越大.每个副本的 V_{latest} 为它提交的最后一个更新操作的 LSN.

定义 2(请求 QoS 约束). 设任意时刻客户发送 ReR 请求 r 来调用副本 x 的服务,则 r 的 QoS 约束定义如下:

(1) 响应时间约束 RT_c,表示 r 允许的最大响应时间,即 $RT_c\geq ET(r)-ST(r)$;

(2) 一致性约束 CC_c,表示 r 允许的最大陈旧度,即 $CC_c\geq STALE(x)=V_{latest}-V(x)$.

当 ReR 请求 r 的 $CC_c=0$ 时,标记 r 为强一致性读(strong consistency read,简称 SCR);当 $CC_c>0$ 时,标记 r 为弱一致性读(weak consistency read,简称 WCR).

定义 3(组件副本域). 自适应选择模型基于域的方式组织 RG 中的副本,包括两个副本域:强一致性域(strong consistency domain,简称 SCD)和弱一致性域(weak consistency domain,简称 WCD),即 $RG=SCD\cup WCD$.$SCD=\{SR_i|SR_i\in RG\},i=1,...,m,m$ 为 SCD 包含的强一致性副本 SR 的数目;$WCD=\{WR_j|WR_j\in RG\},j=1,...,n,n$ 为 WCD 包含的弱一致性副本 WR 的数目.SCD 域中的副本处理 SCR 和 WR 类型的请求,并且始终具有当前最新的版本,即对于任意 $SR_i\in SCD,i\leq m$,则 $STALE(SR_i)=0$;WCD 域中的副本仅处理 WCR 类型的请求,并且具有较老的版本,即对任意 $WR_j\in WCD,j\leq n$,$1\leq CC_{\min}\leq STALE(WR_j)\leq CC_{\max}$,其中,$CC_{\min}$ 和 $CC_{\max}$ 分别表示 WCD 域的最小和最大副本陈旧度,因此,CC_c 属于区间$[CC_{\min},CC_{\max}]$.

定义 4(组管理器). SCD 域中的一个副本被指定为当前的组管理器(group manager,简称 GM).当 GM 失效时,将通过选举产生新的 GM.GM 主要负责以下任务:

(1) 更新请求的 LSN 创建与分配.更新操作的执行次序也是影响副本一致性的重要因素,为了便于叙述,本文提供更新操作的顺序保证(sequential guarantee),该方法可以很容易地扩展到全序、先进先出、因果等次序保证.具体的排序过程是:当客户端的代理拦截到一个更新请求 r 时,将其组播到 SCD 域.当 GM 收到 r 时,立即为它创建新的 LSN,并将其组播到 SCD 中的其他域成员.当非 GM 的副本收到 r 时,首先检查是否已经从当前的 GM 得到其对应的 LSN:如果收到,则传递给组件执行;否则,将其缓存到等待队列(wait queue,简称 WQ)中,直到收到其对应的 LSN.

(2) 性能信息的收集.RG 中的每个副本(GM 除外)都配置了一个性能监控器(performance monitor,简称 PM),PM 周期性地向 GM 发送请求相关的本地性能信息(local performance information,简称 LPI).$LPI=(WT,PT)$,其中,WT 是请求在队列 RQ 中的等待时间,PT 是组件处理请求的时间.GM 为每个 PM 维护一个长度为 L 的 FIFO 队列来存放所收到的 LPI.当队列收到一个新的 LPI 时,如果已经存在 L 个 LPI,则将最早的 LPI 抛弃.这样,GM 仅仅保存了最新的 L 个性能信息.L 的长度可以按需调整,以保证所保存的性能信息能够正确地反映当前各个副本的负载情况.

(3) 副本的自适应选择.在模型中,GM 执行副本选择算法来选择目标副本集.当收到 ReR 请求时,GM 根据收集整理的 LPI 性能信息以及请求的 QoS 约束,同时考虑副本之间的负载平衡,为请求选择最合适的目标副本子集 TRS,$TRS\subset(SCD\cup WCD)$,然后通知 TRS 集合中的副本处理该请求,其他副本则忽略该请求.这里之所以要选择多个副本,是出于容错的考虑,因为副本在处理请求的过程中可能发生失效.如果仅仅选择单个副本,则失效将大量延长处理时间,这对于时间敏感的请求来说显然是不合适的.本文第 3 节将对自适应选择算法 PWARS 详细地加以介绍.

定义 5(一致性管理器). SCD 域中的一个副本被指定为当前的一致性管理器 CM,出于平衡负载的考虑,将 CM 和 GM 分别对应不同的副本.CM 主要负责以下任务:

(1) 更新发布.CM 基于一致性窗口的机制,将 SCD 域中的更新操作发布到 WCD 域中的所有副本.对于 WCD 域中的每个 WR 副本,CM 都分别为其维护了一个一致性窗口(consistency window,简称 CW).CW 集合为

$W=\{W_1,W_2,...,W_n\}$,其中,W_j 为对应于 WR_j 的 CW,W_j 的大小记为整数 WS_j.定义向量 $WS=(WS_1,WS_2,...,WS_n)$为窗口配置(consistency windows configuration,简称 CWC),CWC 中的窗口大小按照非递减顺序排列,即

$$WS_1\le WS_2\le...\le WS_n.$$

当产生新的更新时,CM 首先创建新的更新对象 UO,$UO=(Operation,LSN)$,其中,$Operation$ 表示执行的更新操作.然后,UO 缓存到所有的 CW 中,如果当前 CW 中包含的 UO 个数等于 CW 的窗口大小,则 CM 将更新序列 $US=(UO_1,UO_2,...,UO_m)$,以点对点的方式发送到对应的 WR 副本,然后,CW 被清空并加入新的 UO.也就是说,副本 WR_j 只有在其对应的 W_j 填满时,才能得到最新的 WS_j 个更新,即任何时候 $STALE(WR_j)\le WS_j$.图 2 中给出的是一个一致性窗口的例子,这时,WCD 域中包含 4 个 WR 副本,SCD 域中已经产生了 4 个更新,窗口配置为(2,3,4,5).根据前面的定义,W_1 和 W_2 向 WCD 发布了更新序列 US_1 和 US_2,从而导致各个 WR 副本具有不同的陈旧度,分别为 2,1,4 和 4.因此,WR 副本的一致性可以通过调节窗口配置来进行控制.

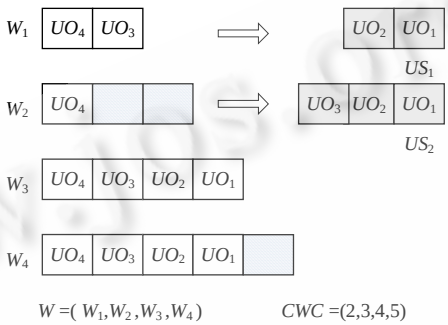


Fig.2 An example of consistency window

图 2 一致性窗口示例

(2) 一致性窗口的自适应重配.对于副本间的更新发布,目前主要采取固定周期发布的方法,如文献[19].SCD 域中的副本按照固定周期向 WCD 域中的所有副本发布最新的更新操作,这样在任何时刻,WCD 域中的副本都具有相同的陈旧度.然而,客户请求的一致性约束并不是恒定不变的,而是随着时间变化的.固定周期的发布方法无法适应这种动态变化的运行环境.为了便于讨论,假定 SCD 域的更新注入率 V_u 不变,更新周期为 T_u,则 WCD 域中所有副本的陈旧度相同,即 $STALE(WR_j)=P_u=V_u\times T_u,1\le j\le n$.如果此时所有客户请求的一致性约束都小于 P_u,则 WCD 域中的所有副本都不能满足客户所要求的一致性约束;如果所有请求的一致性约束都远远大于 P_u,则所有的副本都能满足所需要的一致性约束.然而我们知道,副本之间更新的传播和更新的本地提交都需要消耗系统资源,更新的频率越高,开销越大.很显然,此时浪费了大量的系统资源来维护一个过高的一致性水平,这显然是不合适的.此外,如果请求的一致性约束在一个区间(A,B)内分布,$A<P_u<B$,则由于 WCD 域中的所有的副本始终具有相同的陈旧度,必然会使得一致性约束在(A,P_u)之间的请求得不到满足.显然,固定周期的更新发布方法无法适应一致性约束需求动态变化的环境.

为了避免固定周期方法的缺陷,我们采用了一致性窗口自适应重配算法 CWAR.在 CWAR 算法中,CM 根据最近时间周期的请求一致性分布情况计算出当前合理的窗口配置,然后通过调整一致性窗口的大小来动态调节 CM 向各个 WR 副本发布更新操作的速率,使得各个副本保持不同的陈旧度.这种动态方法能够很好地适应一致性需求不断变化的运行环境.此外,当 CM 完成窗口的调整后,即向 GM 发出一个重配事件 RE 以通知其新的窗口配置.随即,GM 的选择算法根据新的窗口配置为客户请求选择副本集.本文第 4 节将对该重配算法予以详细介绍.

定义 6(客户代理). 每个客户对应一个客户代理 CA,CA 在客户端的 stub 中实现.CA 拦截客户发出的每一个请求,通过判断请求绑定的 QoS 约束确定接受请求的目标副本集(根据定义 2).如果是 WCR,则目标集包括 WCD 域的所有副本、GM 和 CM;如果是 SCR,则目标集包括 SCD 域中的所有副本.当 CA 收到每个请求 r 的第 1 个响应时,通过记录的 $ET(r)$和 $ST(r)$根据定义 2 判断是否满足其响应时间约束,如果满足则立即返回响应,

否则返回一个响应时间错误(response time fault,简称 RTF).对于随后收到的响应 CA 则直接忽略.

3 基于分区加权的自适应副本选择算法PWARS

为了适应客户请求规模和 QoS 需求的不规则动态变化的运行环境,我们提出了一种自适应动态选择算法 PWARS,其基本思想是:通过实时度量副本的性能信息,采取基于分区的加权方法来为请求选择满足其 QoS 需求的目标副本集 *TRS*.为了容错的目的,目标副本集的大小 $TN>1$,TN 可以静态指定,也可以运行时动态指定.

3.1 算法描述

PWARS 算法的主要步骤如下:

步骤 1. 计算副本平均响应时间.

一个客户请求的相应时间 *RT*=请求从客户端代理 CA 到副本的网络传输时间 *TT*+请求在队列 RQ 中的等待时间 *WT*+组件处理请求的时间 *PT*.由于在本地网络环境下,请求的网络传输时间相对稳定,因此,GM 通过 Probe 方法[20]周期性地测量两个副本之间的网络传输时间来近似地代表 *TT*.由前面的介绍可知,*WT* 和 *PT* 可以从副本的 LPI 信息获得.因此,GM 可以计算每个副本最近处理的 L 个请求的平均响应时间 *ART*:

$$ART_i=\left(\sum_{j=1}^{L}(WT_j+PT_j)\right)\Big/L+TT \tag{1}$$

当对所有的副本进行式(1)的操作后,得到 *ART* 的列表 $ARTL=(ART_1,ART_2,...,ART_S)$,其中,$S=m+n-1$(GM 不处理请求,故不包括在内),$m$ 是 SCD 域中的副本数,n 是 WCD 域中的副本数.对于每个请求,在副本选择时都进行式(1)的计算可以提高副本选择的准确度,但是,频繁计算 *ART* 也会引入相对较大的系统开销,需要在这两者之间进行权衡.在这里,对于每个副本,只有当 GM 收到 $L\times\alpha$个新的 LPI 时,才重新计算其新的 *ART*.α的大小需预先指定,调节 $\alpha(0\leq\alpha\leq1)$的大小,就可以控制 *ART* 的计算频率.

步骤 2. 选择目标分区.

首先,根据请求的一致性约束 CC_c 和当前的窗口配置 CWC_i 选出一致性目标分区 CTP,$CTP=\{R_j|R_j\in SCD$ or $(R_j\in WCD$ 且 R_j 的一致性窗口大小 $W_j\leq CC_c,W_j\in CWC_i)\}$,$j=1,...,U$,$U$ 是 CTP 所包含的副本数.CTP 包含了所有的强一致性副本(除 GM)和部分弱一致性副本.

然后,根据请求的响应时间约束 RT_c 从 CTP 分区中选出时间目标分区 TTP,$TTP=\{R_j|R_j$ 的平均响应时间 $ART_q\leq RT_c,R_j\in CTP,ART_q\in ARTL\}$,$j=1,...,V$;$q=1,...,S$.

步骤 3. 选择目标副本集 *TRS*.

为了防止出现 Thundering Herd 现象,PWARS 算法采取了加权选择方法.对于 TTP 中的副本,其当前的 *ART* 越小则说明其负载越低,也就越有可能满足请求的 QoS 约束;相反,其 *ART* 越大,则满足 QoS 约束的可能性就越小.具体方法是:首先计算 RT_c 与 TTP 中的每个副本的 *ART* 的差值,得到序列 $X=(X_1,...,X_i,...,X_V)$,X_i 越高则表示对应的副本满足请求的 QoS 约束的可能性越高.然后,以 X_i 作为权值将 X_i 映射到区间[0,1],得到区间序列 $A=(A_1,...,A_i,...,A_V)$,区间 A_i 对应 TTP 中的副本 R_i.

$$A_1=\left[0,X_1\Big/\sum_{j=1}^{V}X_j\right),\ i=1 \tag{2}$$

$$A_i=\left[\sum_{k=1}^{i-1}\left(X_k/\sum_{j=1}^{V}X_j\right),\sum_{k=1}^{i-1}\left(X_k/\sum_{j=1}^{V}X_j\right)+X_i\Big/\sum_{j=1}^{V}X_j\right),\ i>1 \tag{3}$$

$$\sum_{i=1}^{V}A_i=1 \tag{4}$$

其中,权值 X_i 越大,所分配的区间越大,即 A_i 越大.然后,利用在[0,1]区间平均分布的随机函数 F 产生[0,1]之间的随机数列 $RN,RN=(RN_1,...,RN_i,...,RN_{TN})$,$TN$ 为目标副本集的大小.如果随机数 RN_i 属于区间 A_j,则其对应的副本被选入目标副本集 *TRS*,即 $TRS=\{R_j|RN_i\in$ 副本 R_j 对应的区间 $A_j,R_j\in TTP,RN_i\in RN\}$,$i=1,...,TN$;$j=1,...,V$.如果 $V\leq TN$,则

TRS=*TTP*.

3.2 算法分析

在 PWARS 算法中,计算 *ART* 序列的时间复杂度为 $O(S\times L)$,*S* 是副本组中除 GM 之外的副本数,*L* 是 GM 维护的 LPI 队列的长度.选择目标分区的时间复杂度是 $O(S)$.选择 *TRS* 的时间复杂度为 $O(V^2)$.由于 $V\le S, S\le L$,所以,PWARS 算法的时间复杂度为 $O(S\times L)$.

4 一致性窗口自适应重配算法CWAR

4.1 算法描述

为了适应客户一致性约束的动态变化,一致性管理器 CM 根据一致性约束的分布情况动态地对各个副本的一致性窗口进行调整.从图 3 中可以看出,一致性窗口重配过程主要包括 4 个阶段:监控(monitor)、分析(analyze)、控制(control)和执行(execute).监控阶段首先对收到的请求进行过滤,将 WRC 类型的请求缓存到长度为 *LL* 的队列 *Q* 中,$Q=(WCR_1,WCR_2,...,WCR_m)$.分析阶段周期性地结合当前的 CWC 对 *Q* 中请求的一致性约束分布进行分析,如果发现当前的 CWC 不合理,则进入控制阶段进行重配.在控制阶段,通过一个可能性模型计算出新的 CWC,然后进入执行阶段进行实际的 CWC 调整过程.下面分别对分析、控制和执行 3 个阶段予以详细介绍.

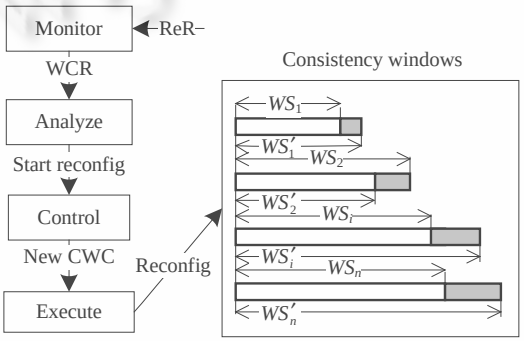


Fig.3 Consistency window adaptive reconfiguration

图 3 一致性窗口自适应重配

4.1.1 分析阶段

分析阶段根据队列 *Q* 中记录的最近的请求信息,判断当前的窗口配置是否合理.假设当前的窗口配置 $CWC=(WS_1,WS_2,...,WS_i,...,WS_n)$.由于 $WS_i<WS_{i+1}$,因此,可以把 *CWC* 看成若干连续的区间:$([WS_1,WS_2],[WS_2,WS_3],...,[WS_i,WS_{i+1}],...,[WS_n,CC_{\max}])$,其中,$W_1\ge CC_{\min}$,区间$[WS_i,WS_{i+1}]$是副本 WCR_i 的对应区间.如果一个请求 *r* 的一致性约束 $CC_c\in[WS_i,WS_{i+1}]$,则称 *r* 属于区间$[WS_i,WS_{i+1}]$,或者称区间$[WS_i,WS_{i+1}]$包含 $r.RC=(RC_1,...,RC_i,...,RC_n)$表示每个区间所包含的请求数.由前面 PWARS 算法可知,如果一个请求 *r* 属于区间$[WS_i,WS_{i+1}]$,那么,所有的区间$[WS_j,WS_{j+1}](j\le i)$都满足该请求的一致性约束,都有可能被选中来执行 *r*.因此,区间越靠前(*i* 越小),被选中的次数可能就越多.为了平衡各个副本的负载,合理的区间分配应该是 *RC* 中的元素是按递增顺序排列的,即 *i* 越靠前,则 RC_i 越小.具体的区间分配比例可以通过实际测试获得,这里,我们假设 *RC* 中元素之间存在(1:2:3:...:*n*)的比例关系时的窗口配置是合理的.

由于请求分布的变化往往是连续的,因此,可以通过对最近的请求分布信息的分析来判断当前的窗口配置是否合理.主要的分析过程如下:

步骤 1. 根据一致性约束,对队列 *Q* 中的请求进行统计,得到当前每个窗口所对应的区间所包含的请求数,即 $RC=(RC_1,...,RC_i,...,RC_n)$.

步骤 2. 对 *RC* 进行分析,如果存在某个窗口的请求数相对于请求平均数的偏离距离超过阈值 ε 时,则说明现

有的窗口配置不能适应当前请求的一致性约束分布,需要对窗口进行重配.触发重配的公式如下:

$$\mathrm{Max}\left(\left(RC_{\max}-\left(\sum_{i=1}^{n}RC_i\right)\Big/n\right),\left(\left(\left(\sum_{i=1}^{n}RC_i\right)\Big/n\right)-RC_{\min}\right)\right)\Big/\left(\left(\sum_{i=1}^{n}RC_i\right)\Big/n\right)>\varepsilon \tag{5}$$

在式(5)中,$RC_{\max}$ 和 $RC_{\min}$ 分别表示 RC 中的最大值和最小值.

4.1.2 控制阶段

为了确定新的窗口配置,我们引入了一个一致性约束的可能性模型.其基本思想是:首先,计算最近一段时间内的请求一致性约束的可能性分布,然后基于可能性,按照合理的比例(例如,前面假设的(1:2:3:…:n))对一致性约束的分布进行划分,计算满足这种可能性划分的一致性约束值的序列,该序列即作为新的一致性窗口配置.主要的计算过程如下:

步骤 1. 首先定义一个离散型随机变量 CCR 用来表示请求的一致性约束,CCR 在整数区间[$CC_{\min}$,$CC_{\max}$]取值.然后遍历队列 Q 中记录的最近 LL 个请求,对区间[$CC_{\min}$,$CC_{\max}$]内的每个 c_i 值,计算其在队列 Q 中出现的次数 N_i.最后根据式(6)计算 CCR 的概率分布:

$$P\{CCR=c_i\}=N_i/LL, c_i\in[CC_{\min},CC_{\max}],\ i=1,2,\ldots \tag{6}$$

步骤 2. 按照前面定义的(1:2:3:…:n)的比例关系对整个可能性区间[0,1]进行划分,生成包含 n 个元素的可能性序列 $Z=(Z_1,\ldots,Z_i,\ldots,Z_n)$,$Z_i$ 可以根据式(7)得到:

$$Z_i=i\Big/\sum_{j=1}^{n}j,\ 1\le i\le n,\ \sum_{i=1}^{n}Z_i=1 \tag{7}$$

然后根据步骤 1 得到的概率分布,从 Z_1 开始依次对可能性序列 Z 中的前 n–1 个元素分别寻找满足式(8)和式(9)的最大的 CCR 值 nc_i,依次得到一致性约束值 $nc_1,nc_2,\ldots,nc_{n-1}$:

$$P\{CCR\le nc_1\}\le Z_1 \tag{8}$$

$$P\{nc_{i-1}<CCR\le nc_i\}\le Z_i,\ 1<i\le n-1 \tag{9}$$

这样就得到了新的窗口配置 $CWC=(CC_{\min},nc_1,nc_2,\ldots,nc_{n-1})$.下面通过图 4 所示的例子来说明新窗口配置的计算过程.假定 WR 副本数为 4,即存在 4 个一致性窗口,$CC_{\max}=1$,$CC_{\min}=10$.假设根据式(6)计算得到 CCR 的概率分布为 $P\{CCR=2\}=0.1$,$P\{CCR=3\}=0.1$,$P\{CCR=5\}=0.1$,$P\{CCR=6\}=0.1$,$P\{CCR=7\}=0.2$,$P\{CCR=8\}=0.2$,$\{CCR=9\}=0.2$,当 CCR 为其他值时的概率均为 0.由式(7)计算得到可能性序列 $Z=(0.1,0.2,0.3,0.4)$.根据前面计算得到的概率分布和 Z,依次得到满足式(8)和式(9)的 CCR 值:$nc_1=2$,$nc_2=5$,$nc_3=7$.得到新的一致性窗口配置为 $CWC=(1,2,5,7)$.

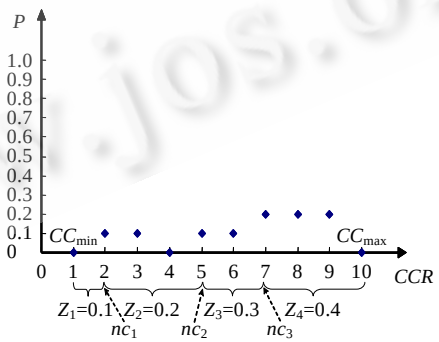


Fig.4 An example of computation for new CWC

图 4 新一致性窗口配置计算示例

4.1.3 执行阶段

根据前面得到的新窗口配置 $CWC'=(WS_1',\ldots,WS_i',\ldots,WS_n')$,CM 对各个副本的一致性窗口的长度进行相应的调整.假设当前的窗口配置为 $CWC=(WS_1,\ldots,WS_i,\ldots,WS_n)$,则存在下面 3 种调整过程:

A. 对于副本 WR_i 的一致性窗口 W_i,如果 $WS_i'=WS_i$,则维持 W_i 的长度不变.

B. 对于副本 WR_i,如果 $WS_i'<WS_i$,则直接将 W_i 的长度缩短到长度 WS_i',并以 RE 事件的方式通知 GM.PWARS 算法根据这个新的窗口长度为新的请求选择副本.这时不会产生不一致的情况.假设在调整窗口长度时,副本 WR_i 的队列 RQ_i 中存在请求序列 $R=(R_1,...,R_i,...,R_m)$,由 PWARS 算法可知,对于 R 中的任何一个请求 R_i 的一致性约束 CC_i,必然存在 $CC_i \geq WS_i$,则 $CC_i > WS_i'$ 成立.因此可以看出,对于在窗口调整时 RQ_i 中已经存在的请求以及在调整后新进入 RQ_i 的请求来说,新窗口长度 WS_i' 都能满足它们的一致性约束.

C. 对于副本 WR_i,当 $WS_i'>WS_i$ 时,如果直接增加窗口的长度,则可能出现不一致的情况.这是因为 WR_i 的队列 RQ_i 中已有的请求序列 R 中可能存在一些请求的一致性约束 CC_i 属于区间 $[WS_i, WS_i']$.在 WS_i' 生效后必然导致这些请求的一致性约束与 WR_i 实际的陈旧度不一致.为了避免这种不一致情况的出现,我们引入了延迟重配协议(deferred reconfiguration protocol,简称 DRP),如图 5 所示.DRP 协议的具体过程如下:

(a) CM 以 RE 事件的方式通知 GM 新的窗口大小 WS_i',PWARS 算法根据 WS_i' 进行组件副本选择.

(b) CM 向副本 WR_i 发送 Reconfig_Start 消息,表示开始窗口重配.CM 维持窗口大小不变.

(c) 副本 WR_i 一旦收到 Reconfig_Start 消息,则立即标记出其请求队列 RQ_i 中的最后一个请求 R_{last}.当 WR_i 完成对 R_{last} 的处理后,立即向 CM 发送 Reconfig_Ready 消息.

(d) CM 收到 Reconfig_Ready 消息后,将 WR_i 的一致性窗口的长度增加到 WS_i',本次窗口重配结束.

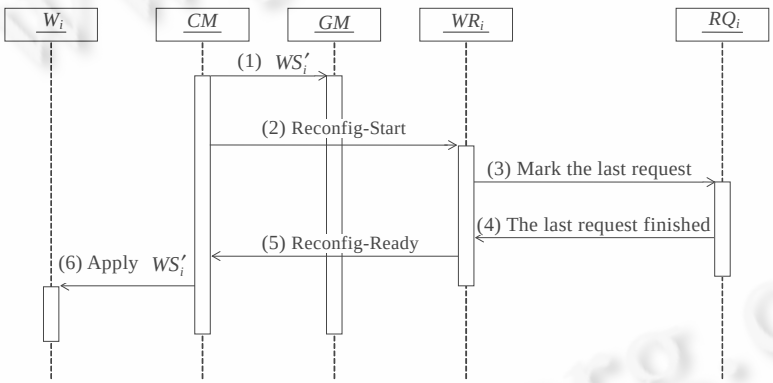


Fig.5 Deferred reconfiguration protocol

图 5 延迟重配协议

4.2 算法分析

由以上对算法的介绍可以看出,分析阶段的时间复杂度为 $O(LL\times n)$.由于队列 Q 的长度 LL 通常大于 WCD 域的副本数 n 和 $(CC_{\max}-CC_{\min})$,因此控制阶段的时间复杂度为 $O(LL^2)$,执行阶段的时间复杂度为 $O(n)$.综上所述,CWAR 算法的时间复杂度为 $O(LL^2)$.

5 实验及结果分析

DARSM 模型已经在中国科学院软件研究所软件工程技术中心自主开发的 J2EE 应用服务器 OnceAS 集群中进行了原型实现.本节通过与现有的 Round-Robin 算法和 Greedy 算法的比较来验证 PWARS 算法的有效性,同时探讨了自适应窗口重配对 PWARS 算法的影响.

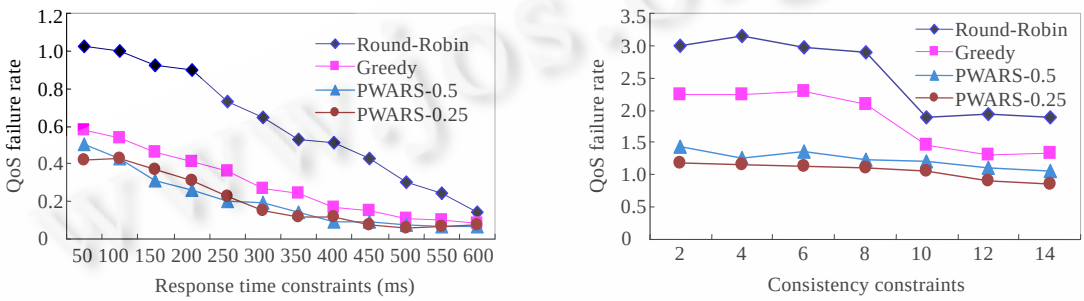
实验环境主要配置为:服务器为 CPU Pentium4 2.8G,内存 2G,客户机为 CPU Pentium4 2.0G,内存 512M,100M 以太局域网.一个由实体 Bean 组件组成的应用分别部署在 7 个服务器上,其中,3 个属于 SCD 域,4 个属于 WCD 域.实验中的参数设置见表 1.

Table 1 Experiment parameters list

表 1 实验参数列表

Parameters	Replicas of SCD	Replicas of WCD	CC_{min}	CC_{max}	L	LL	α	ε
Values	3	4	1	15	20	50	0.3	0.5

3 台客户机上的客户模拟器同时向服务器上的组件发送请求.在每一次测试中,一个模拟器负责注入更新操作,以 100ms 为间隔连续发送 1 000 个更新请求;一个模拟器以 1 000ms 为间隔连续发送 1 000 个读请求,请求的时间约束均为 300ms,一致性约束在区间[1,15]均匀地随机产生;最后一个作为测试模拟器,分两种情况实验:(1) 测试模拟器以 1 000ms 为间隔发送 1 000 个读请求,请求的一致性约束在区间[1,15]均匀地随机产生,每次测试采取不同的响应时间约束;(2) 测试模拟器以 1 000ms 为间隔发送 1 000 个请求,请求的时间约束均为 200ms,每次测试中,所有请求的一致性约束平均值为 t,最大偏差为 2.实验指标 QoS 失败率表示 QoS 约束没有得到满足的请求所占的比率.图 6 是分别采用 Round-Robin,Greedy,PWARS-0.25 和 PWARS-0.5 算法(其中, PWARS-0.25 和 PWARS-0.5 是指当ε取 0.25 和 0.5 时的 PWARS 算法)进行实验的结果.



(a) QoS failure rate comparison on response time constraints
(a) 不同响应时间约束 QoS 失败率对比

(b) QoS failure rate comparison on consistency constraints
(b) 不同一致性约束 QoS 失败率对比

Fig.6 Experimental results

图 6 实验结果

由图 6(a)可以看出,随着响应时间约束的变小,3 种选择算法的优劣逐渐明显.Round-Robin 算法的 QoS 失败率最高,当响应时间约束为 50m 时,QoS 失败率达到了 1.02%.与 Round-Robin 算法相比,Greedy 算法通过对当前副本的响应时间评估来进行选择,从而具有较低的 QoS 失败率.而 PWARS 算法不仅考虑了副本当前的性能,而且考虑了副本的一致性约束,其 QoS 失败率与 Round-Robin 算法和 Greedy 算法相比都有所降低,最高也仅有 0.45%的失败率.当ε取 0.25 和 0.5 时,由于一致性约束的分布相对稳定,PWARS 算法在这两种情况下的表现并没有明显的差异.

图 6(b)清楚地显示了 PWARS 算法在出现一致性约束分布动态变化时的优势.当一致性约束小于 10 时,Round-Robin 算法和 Greedy 算法表现出了较高的 QoS 失败率,而且比较稳定,这是由于算法采用的是固定周期的更新发布,较大部分 QoS 失败是由于等待更新的到达而造成的.当一致性约束大于等于 10 时,此时不需要等待更新以满足一致性约束,导致 QoS 失败率出现较大的下降.相比之下,PWARS 算法在不同的一致性约束下均表现出相对稳定的 QoS 失败率.此外,当ε取值较小时(ε=0.25),PWARS 算法选择效果较好,因为此时 CM 能够更快地适应一致性约束的变化,及时地调整副本的一致性窗口,从而提高了副本选择的准确性.从上述实验可以直观地看出,DARSM 方法显著地提高了组件副本选择的性能.

下面验证自适应窗口重配对 PWARS 算法的影响.实验中,测试模拟器以 1 000ms 为间隔发送 1 000 个请求,所有请求的一致性约束在区间[1,15]无规则变化,每次测试采取不同的响应时间约束.我们分两种情况进行了实验:(1) 使用自适应窗口重配(CWAR 算法);(2) 使用固定一致性窗口.其余参数与前面的实验相同,测试结果如图 7 所示.

从图 7 明显可以看出,自适应窗口重配使得 PWARS 算法表现出较低的 QoS 失败率.这是因为,当一致性约

束的分布发生动态变化时,固定的一致性窗口使得各个副本得到更新的速度相对稳定,这样必然会出现各个副本上请求分配不均的情况,即一些副本获得了过多的请求,而另一些副本则可能很少甚至没有分配到请求.这就增加了请求在队列中的等待时间,从而导致了 QoS 失败率的增加.显然,采用动态的窗口调整方法较好地适应了这种动态变化的情况.

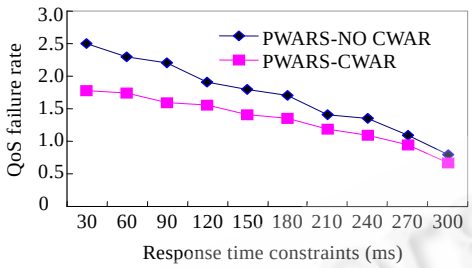


Fig.7 CWAR's influence on algorithm PWARS

图 7 CWAR 对 PWARS 算法的影响

6 结束语

组件复制是分布式计算中间件平台提高应用可靠性、可用性和伸缩性的关键技术.本文研究在动态变化的 QoS 需求和系统运行环境下,如何有效地选择组件副本的问题.首先,给出了基于域的自适应副本选择模型 DARSM;其次,基于此模型,提出了基于分区加权的自适应副本选择算法 PWARS;然后,为适应一致性约束的动态变化,设计了一致性窗口自适应重配算法 CWAR.本文提出的模型和算法已在我们自主研制的 J2EE 应用服务器 OnceAS 2.0 中实现.实验结果表明,相对于已有的算法,该方法明显地提高了副本选择的性能.

进一步的工作包括两个方面:(1) 如何对 PWARS 算法和 CWAR 算法进行优化,降低执行成本,提高系统的整体性能;(2) 如何提供容错支持,尤其是当组管理器 GM 或一致性管理器 CM 失效时,仍然能够保证副本选择的进行.

References:

[1] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

[2] Felber P, Defago X, Eugster P, Schiper A. Replicating CORBA objects: A marriage between active and passive replication. In: Kutvonen L, ed. Proc. of the 2nd IFIP Int'l Working Conf. on Distributed Applications and Interoperable Systems. Kluwer, 1999. 375−388.

[3] Shannon B. Java™ 2 Platform Enterprise Edition Specification, v1.4. Sun Microsystems Inc., 2003.

[4] Fan GC, Zhong H, Huang T, Feng YL. A survey on Web application servers. Journal of Software, 2003,14(10):1728−1739 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1728.htm

[5] Shen K, Yang T, Chu LK. Cluster load balancing for fine-grain network services. In: Proc. of the Int'l Parallel and Distributed Processing Symp. Fort Lauderdale: IEEE Computer Society, 2002. 0051b.

[6] Cardellini V, Colajanni M, Yu P. Request redirection algorithms for distributed Web systems. IEEE Trans. on Parallel and Distributed Systems, 2003,14(4):355−368.

[7] Huang T, Chen NJ, Wei J, Zhang WB, Zhang Y. OnceAS/Q: A QoS-enabled Web application server. Journal of Software, 2004, 15(12):1787−1799 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1787.htm

[8] Othman O, O'Ryan C, Schmidt DC. Designing an adaptive CORBA load balancing service using TAO. IEEE Distributed Systems Online, 2001,2(4):1541−4922.

[9] Krishnamurthy S, Sanders WH, Cukier M. An adaptive framework for tunable consistency and timeliness using replication. In: Frazier T, ed. Proc. of the Int'l Conf. on Dependable Systems and Networks. Bethesda: IEEE Computer Society, 2002. 17−26.

[10] Ren YS, Bakken DE, Courtney T, Cukier M, Karr DA, Rubel P, Sabnis C, Sanders WH, Schantz RE, Seri M. Aqua: An adaptive architecture that provides dependable distributed objects. IEEE Trans. on Computers, 2003,52(1):31−50.

[11] Podlipnig S, Böszörményi L. A survey of Web cache replacement strategies. ACM Computing Surveys, 2003,35(4):374−398.

[12] Krishnaswamy V, Raynal M, Bakken D, Ahamad M. Shared state consistency for time-sensitive distributed applications. In: Lanus M, ed. Proc. of the 21st Int'l Conf. on Distributed Computing Systems. Arizona: IEEE Computer Society, 2001. 606−614.

[13] Singla A, Ramachandran U, Hodgins J. Temporal notions of synchronization and consistency in beehive. In: Leiserson CE. Proc. of the 9th ACM Symp. on Parallel Algorithms and Architectures. Newport: ACM Press, 1997. 211−220.

[14] Yu H, Vahdat A. Design and evaluation of a continuous consistency model for replicated services. In: Proc. of the 4th Symp. on Operating Systems Design and Implementation. San Diego: USENIX Association, 2000. 305−318.

[15] IBM Inc. WebSphere Scalability: WLM and Clustering—Using WebSphere Application Server Advanced Edition. IBM. 2000. http://www.redbooks.ibm.com/redbooks/pdfs/sg246153.pdf

[16] BEA Systems Inc. Achieving scalability and high availability for e-business. BEA White Paper, BEA, 2003. http://dev2dev.bea.com/pub/a/2004/01/WLS_81_Clustering.html

[17] Labourey S, Burke B. JBoss Clustering v2.0. The JBoss Group, 2002. http://www.jboss.com/

[18] Lamport L. Time, clocks, and the ordering of events in distributed systems. Communications of the ACM, 1978,21(7):558−565.

[19] Petersen K, Spreitzer MJ, Terry DB, Theimer MM, Demers AJ. Flexible update propagation for weakly consistent replication. In: Proc. of the ACM 16th Symp. on Operating Systems Principles. St. Malo: ACM Press, 1997. 288−301.

[20] Fei ZM, Bhattacharjee S, Zegura EW, Ammar MH. A novel server selection technique for improving the response time of a replicated service. In: Proc. of the INFOCOM. San Francisco: IEEE Computer Society, 1998. 783−791.

附中文参考文献:

[1] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

[4] 范国闯,钟华,黄涛,冯玉琳.Web 应用服务器研究综述.软件学报,2003,14(10):1728−1739. http://www.jos.org.cn/1000-9825/14/1728.htm

[7] 黄涛,陈宁江,魏峻,张文博,张勇.OnceAS/Q:一个面向 QoS 的 Web 应用服务器.软件学报,2004,15(12):1787−1799. http://www.jos.org.cn/1000-9825/15/1787.htm

左林(1975－),男,四川成都人,博士,主要研究领域为网络分布计算,软件工程.

刘绍华(1976－),男,博士,助理研究员,主要研究领域为网络分布计算,工作流技术,服务协作理论,中间件.

魏峻(1970－),男,博士,研究员,CCF 高级会员,主要研究领域为软件工程,软件理论,网络分布计算.

冯玉琳(1942－),男,博士,研究员,博士生导师,主要研究领域为软件工程,形式化方法,分布对象计算.

范国闯(1974－),男,博士,副研究员,主要研究领域为应用软件工程,网络分布计算,中间件,企业应用集成.