提高冗余服务性能的动态容错算法*

钱 方, 贾 焰, 黄 杰, 顾晓波, 邹 鹏

(国防科学技术大学 计算机学院,湖南 长沙 410073)

E-mail: jiayan@nudt.edu.cn

http://www.nudt.edu.cn

摘要: 针对分布式应用的性能要求,引入了负载平衡机制,以便对 active replication 和 primary backup 容错算法进行权衡,提出一种基于冗余服务的动态容错算法 RAWA(read-any-write-any),能根据系统负载状况动态改变请求的 quorum,不但提高了请求的处理速度,而且以一种简单、有效的方式实现了负载平衡.结合所提出的一致性维护和互斥访问机制,该算法可以适用于嵌套访问和状态服务.另外,还分析了 RAWA 算法的性能,并通过在 CORBA 平台上与其他容错算法的对比测试,证明 RAWA 算法在不改变系统可用性的前提下有效地提高了冗余服务的性能.

关键词: 客户/服务器;冗余服务;容错;负载平衡;quorum

中图法分类号: TP302 **文献标识码**: A

在以客户/服务器为模式的分布计算环境中,许多系统采用冗余服务以提高可用性.随着大规模事务处理和分布实时等应用的发展,越来越多的系统要求在不改变系统可用性的前提下,改进冗余服务的性能,提高系统的吞吐率.然而现有的容错算法大多只针对冗余服务的可用性,而对它们的性能考虑得不够全面.

目前,基于冗余服务的容错算法大体上分为两类:primary backup 和 active replication[1].在分布式系统中,执行某个请求的冗余服务的集合被定义为它的 quorum[2].假设系统中冗余服务的个数为 N.在 primary backup 算法中 quorum 的大小为 1,所有的请求由一个 primary 服务串行执行,因此请求的响应时间长,系统负载不均衡,但它占用的服务资源少;在 active replication 算法中,读写请求的 quorum 是系统中所有冗余服务的集合,大小为 N,所有的冗余服务同时响应客户的请求,处理请求的速度快,但它耗费系统资源,而且由于系统资源有限,冗余服务同时执行请求,从总体上降低了系统的性能.为了使 active replication 和 primary backup 算法适用于状态服务[3](stateful server)和嵌套访问,文献[4]和 hot replication 算法[5]对它们进行了改进,但 quorum 的大小没有改变.ROWA(read-once write-all)算法对 active replication 和 primary backup 进行了权衡,但它只是对读写请求进行分类处理,quorum 的大小还是固定在 1 或 N,因此,它只适用于读频率较高的情况,而且降低了读请求的可用性.综上所述,为了对 active replication 和 primary backup 算法进行真正而有效的权衡,需要改变请求的 quorum,使它可以定义在 1 和 N 之间,即在冗余服务中选择一组服务来响应请求.对无嵌套访问的无状态服务[3](stateless server)而言,这是允许

* 收稿日期: 1999-06-02; 修改日期: 2000-02-29

基金项目: 国家 863 高科技发展计划资助项目(863-306-ZD02-02-5.7)

作者简介: 钱方(1973-),女,江苏无锡人,博士,讲师,主要研究领域为分布计算,系统管理;贾焰(1960-),女,四川成都人,博士,教授,博士生导师,主要研究领域为数据库,分布计算;黄杰(1976-),男,陕西西安人,博士生,主要研究领域为分布计算;顾晓波(1976-),男,江苏南京人,助理工程师,主要研究领域为分布计算;邹鹏(1958-),男,山东高密人,教授,博士生导师,主要研究领域为操作系统,分布计算.

© 中国科学院软件研究所 http://www.jos.org.cn

的;然而对状态服务和嵌套访问而言,需要对服务状态进行一致性维护,或者保证嵌套访问的服务状态不会被重复修改.

因此,本文提出一个提高冗余服务性能的动态容错算法 RAWA(read-any-write-any),它是基于下述设计思想:(1) 通过将请求 quorum 的大小定义在 1 和 N 之间,对 active replication 和 primary backup 算法进行权衡,缩短了请求的响应时间,又不过多地占用系统资源;(2) 利用系统的负载信息动态地定义请求的 quorum,选择轻载的冗余服务响应客户的请求,quorum 的大小和内容随着系统负载的变化而动态地加以改变,既提高了请求的处理速度,又实现了负载平衡;(3) 针对状态服务和嵌套访问,提供状态的一致性维护机制和互斥访问机制,提高了算法的适用范围.本文在 CORBA[6]平台上对 RAWA 算法进行了实现,通过与其他算法的比较和测试表明,在保证系统可用性的前提下有效地提高了冗余服务处理请求的速度,能较好地满足分布式应用的性能要求.

1 算法设计

1.1 系统模型

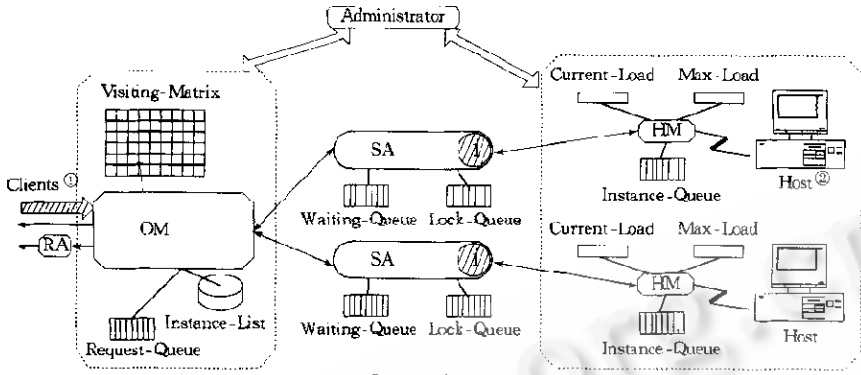
本文采用分布对象的观点建立系统模型.系统中有多个相互连接的主机,它们基于可靠的异步通信机制,即消息传递无丢失、无重发、顺序且不被损坏.系统中的服务分为两种类型[3]:状态服务将数据信息,即服务的状态封装在内部实现;而无状态服务只封装了计算能力.服务有两种错误类型:fail-stop 和 Byzantine.系统中客户可见的服务被抽象成服务类,它们由多个冗余服务实现,这些冗余服务被定义为服务类的各个实例[7].同一服务类的实例继承相同的服务接口,驻留在不同的主机上,对客户透明.它们基于主动对象(active object):不但向客户提供数据信息和计算服务,还可以访问其他对象.针对冗余服务,系统提供了可靠的组通信机制,它满足两个特性[1]:(1) 原子性(atomic):广播给一个服务类的消息,或者被全部实例接收,或者全不被接收;(2) 顺序性(order):同一个服务类的实例以相同的顺序接收广播给它们的消息.

1.2 算法框架与数据结构

RAWA 算法框架由 Administrator,OM(object manager),HM(host manager),SA(server agent)和 RA(request agent)5 个模块组成(如图 1 所示).它们采用 agent 模型,以一个统一的接口接受消息,通过消息适配器和转发器,对消息分别进行处理.

对客户而言,实例即冗余服务的实现是透明的,客户将申请某个服务类的请求都发送给这一服务类的冗余服务的管理者 OM,由它根据一定的机制将请求转发给适当的实例.OM 含有一个请求队列 Request-Queue 和一个实例列表 Instance-List,分别存储客户发送给它的请求信息(方法名、参数、结果类型和请求类型)以及它所管理的冗余服务信息(实例的地址、状态).当某个冗余服务失效时,OM 根据其他实例的状态和请求队列对失效实例进行恢复.针对嵌套访问,OM 还提供了 Visiting-Matrix,以保证对实例的互斥访问.HM 驻留在每个主机节点上,含有主机当前的负载状况 Current-Load 和负载阈值 Max-Load 以及等待执行的实例队列 Instance-Queue.SA 为每个实例 I(Instance)提供了管理接口,它一方面负责接收 OM 转发的请求,另一方面向 HM 查询当前主机的负载状况,决定是否执行请求.它含有一个等待队列 Waiting-Queue,存放暂时不能响应的请求.对于状态服务,它提供了锁机制和等待队列 Lock Queue.Administrator 含有全局的服务对象和主机信息,通过定期地向 OM 和 HM 发送检测请求,监控系统中的实例和主机的运行状况:一方面实时地显示给系统管理员,另一方面对失效的 OM 和 HM 进行恢复.RA 驻留在客户方,负责接

© 中国科学院软件研究所 http://www.jos.org.cn

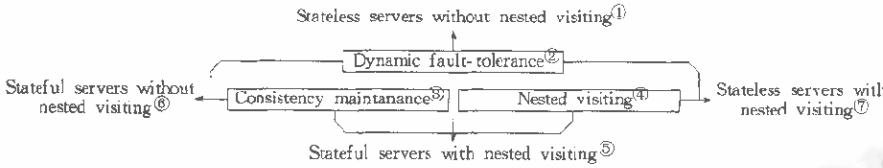


①客户,②主机.

Fig. 1 System architecture of algorithm RAWA
图1 RAWA算法的系统框架

收请求的结果.针对服务的 Byzantine 错误,RA 内部封装了客户定义的 acceptance-tests[8],对返回的结果进行测试.然后通过 majority-voting,选举正确结果.

RAWA 算法分为 3 个部分:动态容错、一致性维护和互斥访问.其中,动态容错只针对不含有嵌套调用的无状态服务.为了支持状态服务和嵌套访问,本文增加了一致性维护和互斥访问机制.通过三者不同的组合,可适用于各种服务类型的分布计算环境(如图 2 所示).



①无嵌套访问的无状态服务,②动态容错,③一致性维护,④互斥访问,⑤嵌套访问的状态服务,⑥无嵌套访问的状态服务,⑦嵌套访问的无状态服务.

Fig. 2 Structure of algorithm RAWA
图2 RAWA算法的结构

由于 RA 的实现,不但需要客户对可用性提出具体要求,而且需要客户针对不同请求定义 acceptance-tests,因此本文在这里只讨论 fail-stop 类型的错误.Administrator 定期地向 HM 和 OM 发出检测请求,可以及时发现失效的 OM 和 HM 对象,并根据状态信息对它们进行恢复.因此,这里假定 OM 和 HM 都是强壮(robust)的.

1.3 动态容错

动态容错的核心思想是随着系统负载状况的变化,动态地选择一组轻载的冗余服务响应客户的请求.由于冗余服务驻留在各个主机上,它们轻载与否由所在主机的负载状况决定.因此在 RAWA 算法中,OM 将请求转发给服务类的所有实例,负载平衡调度由各个主机上的 HM 在本地完成,这样既避免了 OM 成为新的处理瓶颈,也保证了负载信息收集的简单、实时和高效,同时也避免了因为响应时间超时而对请求进行重发.

系统中有 6 种消息类型:(1) request-message(RequestSeq,MethodName,Prameters,Result-Type):客户向 OM 发送的和 OM 向 SA 发送的请求,RequestSeq 是 OM 按照接收请求的先后顺序为请求分配的全局标识;(2) inquiry-message(InstanceAddr):SA 向 HM 发送的负载查询消息,InstanceAddr 为 SA 的地址;(3) load-message(HostLoad):HM 向 SA 发送的负载查询结果;

© 中国科学院软件研究所 http://www.jos.org.cn

(4) executed-message();SA 向 HM 发送的请求执行结束消息;(5) result-message(RequestSeq, Result):实例向 OM 发送的请求 RequestSeq 的结果;(6) received-message(RequestSeq):OM 向 SA 发送的消息,表明请求 RequestSeq 的结果已经收到.

OM 将客户的请求按照 FIFO 的顺序放入 Request-Queue,然后将它们以组通信的方式广播给所有冗余服务的 SA. SA 接收到 OM 转发的请求之后,向 HM 查询当前主机的负载状况,若主机轻载,则它创建新的线程对请求进行处理;否则,将请求按 FIFO 顺序放入 Waiting-Queue. 当冗余实例处理完请求之后,将结果返回给 OM. OM 将收到的结果返回给客户,并向其他实例广播一个请求结束消息 Received-Message. 其他实例收到 OM 的消息后,将该请求从等待队列 Waiting-Queue 中取出.

1.4 一致性维护

针对状态服务,本文对上述的动态容错算法进行了改进,对读写请求分别进行处理:由于读请求不改变服务的状态,因此仍然采用上述的动态容错算法;对于写请求,则进行了两处修改,即增加了更新消息和锁机制.

首先,对 request-message 和 result-message 的消息格式进行改进,增加了描述请求类型的参数 RequestMode 和描述请求执行后被修改的对象状态的参数 ChangedStates. 同时,增加了一种新的消息类型:OM 向 SA 发送的更新状态消息 update-message,其格式为(RequestSeq,ChangedStates). 当实例执行完写请求,它将请求的结果封装在 result-message,返回给 OM,同时也将改变的状态 ChangedStates 通知 OM. OM 接收到写请求的结果后,将 result-message 转换成相应的 update-message. 其他实例接收到 update-message 后,若写请求还在等待队列中,则把它替换成 update-message. 这样,当执行到这一请求时,只需对实例状态进行修改,而无须执行整个请求.

在第 1.3 节的动态容错算法中,由于服务没有状态,且无嵌套调用,因此 SA 可以为每个请求创建线程,并行执行. 但在状态服务中,请求对服务的状态进行读写操作,因此必须保证各个请求对服务状态的互斥访问. 本文通过对服务状态加锁并在 SA 中实现一个 Lock-Queue,来控制读写请求对服务状态这一共享资源的访问.

1.5 互斥访问

在嵌套服务中,执行同一个请求的服务类的多个实例可能对外部对象进行重复访问. 为了保证外部对象的状态不被重复修改,本文引入了印象(image)[4]这一概念,并在 OM 中增加了访问矩阵 Visiting-Matrix. 印象,被定义为不同实例对同一请求的执行标识. 它满足两个特性:(1) 每个独立的请求都有一个全局惟一的标识;(2) 同一请求的所有印象含有相同的请求标识.

在 RAWA 算法中,发送给同一服务类的请求由它的 OM 统一接收,因此,本文将请求的全局标识 RequestId 定义为 ObjectName(所请求的服务类名)+RequestSeq(OM 赋予请求的序号). 由于服务类名 ObjectName 全局惟一,请求的序号 RequestSeq 在各个服务类中也是惟一的,因此,请求的全局标识 RequestId 满足印象的特性(1). 当各个实例执行请求时,请求的全局标识 RequestId +实例对外部对象的访问序号 VisitingSeq 被定义为请求的印象 RequestImage. 假设系统中的对象都是确定的(deterministic),各个实例以相同的次数和顺序同步地访问外部对象,因此,印象的定义满足特性(2). 当被访问服务类的 OM 接收到含有印象的请求时,它为请求赋予新的序号,则请求获得一个新的标识:ObjectName+RequestSeq. 这样,各个实例在执行请求时又可以进行下一轮的嵌套访问,嵌套访问的层数和次数因而不受限制.

© 中国科学院软件研究所 http://www.jos.org.cn

通过引入印象的概念,本文对动态容错算法中的OM进行了改进.在OM中增加了一个二维的数组变量Visiting-Matrix,它的横坐标是所有访问OM的客户地址(有可能是服务对象),纵坐标是请求的印象RequestImage(如图3所示).例如,$V(i,j)\in$Visiting-Matrix,是由第i个客户发来的第j个请求,它的格式为(RequestState,Result,Client-Queue).其中,RequestState代表该请求是否已被响应过,Client-Queue是等待请求结果的客户地址.当客户的请求已被响应过,则直接将结果返回;若正在响应,则将客户地址放入等待队列;若没有响应,则执行这一请求.当OM接到实例返回的结果时,将结果返回给所有等待的客户(算法的具体描述略).

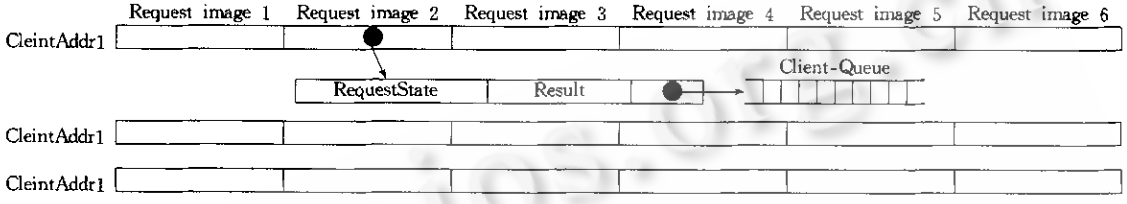


Fig. 3 Structure of Visting-Matrix
图3 访问矩阵Visting-Matrix的结构

2 性能分析

本文对RAWA算法、active replication算法、primary backup算法以及ROWA算法的平均响应请求时间进行了比较.假定各种算法中冗余服务的个数相同,因此,RAWA算法和其他算法一样具有相同的可用性.

假设系统中有n个主机,它们处理请求的速度分别为$p_1,\ldots,p_n$.系统中有m个服务类,各个服务类分别由$R_i(i=1,\ldots,m)$个实例实现.实例失效的概率为f,检测实例失效的时间为t_t,对失效实例进行恢复的时间为t_r.正常实例执行一个请求的平均时间为t_e,请求中写请求的概率为ρ,则在各种算法中,请求的平均响应时间为系统正常工作的概率f_n乘以请求的平均执行时间t_a与系统故障概率f_f乘以实例的检测、恢复时间与执行时间$(t_t+t_r+t_e)$之和:

$$\bar{t}=f_n t_a+f_f(t_t+t_r+t_e).$$

(1) 在active replication算法中,各个实例同时响应客户的请求,系统中只要有一个正常实例,客户请求就可以及时得到响应,因此,系统正常工作的概率为$(1-f_n)$.正常实例执行请求的平均时间t_a等于当前系统正在执行的请求数除以系统的处理速度:$t_a=\frac{N_r}{p}$.由于在active replication算法中,请求的quorum等于各个服务类所有的实例集合,因此系统当前正在执行的请求数N_r等于系统中实例的个数:$N_r=\sum_{i=1}^{m}R_i$;P为系统中所有主机的处理速度之和,$P=\sum_{i=1}^{n}p_i$.综上所述,active replication算法中请求的平均响应时间为

$$\bar{t}_a=(1-f^n)t_a+f^n(t_t+t_r+t_a), \tag{1}$$

其中$t_a=\frac{\sum_{i=1}^{m}R_i}{\sum_{i=1}^{m}p_i}$.

(2) 在primary backup算法中,所有请求都由primary服务串行执行,因此系统正常工作的概率为$(1-f)$.正常实例执行请求的平均时间t_p等于primary服务当前正在执行的请求数除以

© 中国科学院软件研究所 http://www.jos.org.cn

primary服务所在主机的处理能力 P_p.假定系统中所有服务类的 primary 都驻留在同一主机上,则 $t_p=\frac{N_t}{P_p}$,其中 N_p 等于系统中服务类型的个数 m.因此,primary backup 算法中请求的平均响应时间为

$$\bar{t}_p=(1-f)t_p-f(t_t+t_r+t_p), \tag{2}$$

其中 $t_t=\frac{m}{P_p}$.

(3) ROWA 算法是 active replication 和 primary backup 算法的折衷:将读请求按照 primary backup 算法处理,将写请求按照 active replication 算法处理.因此,ROWA 算法中请求的平均响应时间为

$$\bar{t}_a=(1-\rho)[(1-f)t_p-+f(t_t+t_r+t_p)]+\rho[(1-f^n)t_a+f^n(t_t+t_r+t_a)]. \tag{3}$$

(4) RAWA 算法根据系统的负载状况选择轻载服务响应请求,请求的 quorum 小于或等于各个服务类的实例集合.因此,对每个服务类而言,正在响应请求的实例个数 R'_i 小于或等于实例总数 R_i,则系统正在执行的请求数 $N'_r=\sum_{i=1}^{m}R'_i\leqslant N_r$;正常实例执行请求的平均时间 $t_w=\frac{N'_r}{P}\leqslant t_a$.因此,RAWA 算法中请求的平均响应时间为

$$\bar{t}_w=(1-f^n)t_w+f^n(t_t+t_r+t_w), \tag{4}$$

其中 $t_w=\frac{\sum_{i=1}^{m}R'_i}{\sum_{i=1}^{m}p_i}$.

通过对公式(1)~(4)的比较可以得出,$t_w\leqslant t_a\leqslant t_o\leqslant t_p$,即与 active replication,primary backup 算法以及 ROWA 算法相比,RAWA 算法的请求平均响应时间最短.第 4 节中的实验测试验证了这一结论.

3 相关工作及其比较

RAWA 算法采用动态容错的思想:通过动态改变请求的 quorum 大小,对 active replication 和 primary backup 算法进行权衡.当系统轻载时,RAWA 算法类似于 active replication 机制,请求被大多数的冗余服务同时响应,既提高了请求的执行速度,又不浪费服务资源;当系统重载时,RAWA 算法类似于 primary backup 机制,请求只被少数的轻载服务,即 primary 所响应,不会过多地占用系统资源.

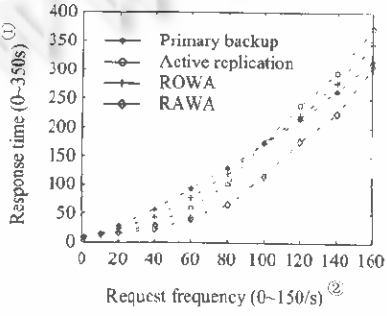
一些容错算法也采用这种基于 quorum 的动态容错思想,例如,voting-based 和 structure-based 算法[9].RAWA 算法与它们的最大区别在于:通过 OM 对冗余服务进行集中管理.因此,它具有以下几个优点:(1) 所有的请求由 OM 统一接收,有利于请求的全局排序以及维护冗余服务的状态一致性;(2) OM 起到了消息队列(message queue)的作用,可以有效地增加客户方的请求连接数目,同时也对服务方进行性能维护;(3) OM 减少了客户和服务方之间的通信开销以及客户方存储服务信息的开销.

© 中国科学院软件研究所 http://www.jos.org.cn

4 算法实现及性能测试

本文采用通用对象请求代理框架 CORBA[6]作为算法的实现平台,将算法框架中的 Administrator 作为系统服务实现;OM 和 HM 作为 CORBA 对象实现,其中 HM 驻留在系统的每个主机上,在主机加入系统时由 Administrator 自动创建;RA 和 SA 分别在对象的 stub 和 skeleton 中实现,由 IDL 编译器自动生成.基于 CORBA 平台,我们分别对 RAWA 算法、active replications 算法、primary backup 算法和 ROWA 算法进行了性能对比测试.

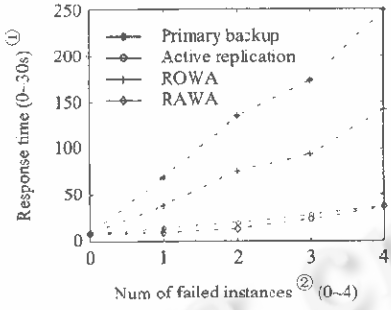
图 4 给出了在实例的冗余度为 3,系统中无失效实例时,改变请求频率对各个算法的请求响应时间的影响曲线.从图 4 中可以看出,随着请求频度的增加,请求的响应时间逐渐加大.与其他算法相比,RAWA 算法无论在轻载还是重载情况下,其响应时间都是最低的,而且它所能处理的请求频度最高,即正常工作时的吞吐率最高.图 5 给出了实例的冗余度为 5,请求频率为 200/sec,当失效实例个数增加时各个算法响应请求的时间曲线.从图 5 中可以看出,RAWA 算法与 active replication 算法的响应时间曲线类似:只要系统中还有一个正常实例,客户请求就能够及时得到响应,而且它的平均响应请求时间总是低于 active replication 算法的响应时间.



①响应时间(0~350 秒),②请求频率(0~150/秒).

Fig. 4 Variable curve of response time with different request frequencies

图 4 响应时间随请求频率的变化曲线



①响应时间(0~30 秒),②失效实例个数.

Fig. 5 Variable curve of response time with different failed instance numbers

图 5 响应时间随失效实例个数的变化曲线

5 结束语

针对当前容错算法的性能缺陷,本文提出了一种提高冗余服务性能的动态容错算法 RAWA.它通过引入负载平衡机制,动态改变请求的 quorum,对 active replication 和 primary backup 算法进行权衡.针对状态服务和嵌套访问,本文在 RAWA 算法中增加了一致性维护和互斥访问机制,有效地扩大了 RAWA 算法的适用范围.目前,RAWA 算法适用于基于局域网的分布式应用环境,系统的通信开销与请求执行时间相比可以忽略不计.下一步,我们将针对广域网环境,将系统通信开销引入 RAWA 算法.

References:

[1] Rachid, G., Andre, S. Software-Based replication for fault tolerance. IEEE Computer, 1997,30(4):68~74.

[2] Mustaque, A., Mostafa, H. A. Performance characterization of quorum-consensus algorithms for replicated data. IEEE Transactions on Software Engineering, 1989,15(4):492~496.

© 中国科学院软件研究所 http://www.jos.org.cn

[3] Silvano, M. Client/Server term definition. In: Hemmendinger, D., Ralston, A., Reilly, E. D., eds. Encyclopaedia of Computer Science. Zurich: International Thomson Computer Publishers, 1998.

[4] Pankaj, J. Resilient objects in broadcast networks. IEEE Transactions on Software Engineering, 1989,15(1):68~72.

[5] Ganesha, B., Anish, K., Anil, G., *et al*. Fault tolerant objects in distributed systems using hot replication. Technical Report, TR-95-023, Texas, College Station: Department of Computer Science, Texas AM University, 1996. 89~95.

[6] Steve, V. New features for CORBA 3.0. Communications of the ACM, 1998,41(10):44~52.

[7] Qian, Fang, Zou, Peng, Chen, Yu, *et al*. A redundant server model based on distributed objects. Journal of Computer Research and Development, 1999,36(11):139.~1397 (in Chinese).

[8] Somani, A. K., Vaidya, N. H. Understanding fault tolerance and reliability. IEEE Computer, 1997,30(4):45~50.

[9] Rabinovich, M. Efficient replication management in distributed systems [Ph. D. Thesis]. Washington, DC: University of Washington, 1994.

附中文参考文献:

[7] 钱方,邹鹏,陈渝,等.基于分布对象的冗余服务模型.计算机研究与发展,1999,36(11):1391~1397.

A Dynamic Fault Tolerant Algorithm for Improving Performance of Redundant Services*

QIAN Fang, JIA Yan, HUANG Jie, GU Xiao-bo, ZOU Peng

(*Institute of Computer Science, National University of Defence Technology, Changsha* 410073, *China*)

E-mail: jiayan@nudt.edu.cn

http://www.nudt.edu.cn

Abstract: According to the performance requirements of distributed applications, the load balancing is introduced into fault tolerance in this paper to reach tradeoff between active replication algorithm and primary backup algorithm. A dynamic fault tolerant algorithm RAWA (read-any-write-any) is presented in the paper for redundant servers to dynamically change quorum with variability of system load. It not only improves the processing speed of requests, but also implements load balancing in a more efficient and simplier way. Integrated with consistency and mutual visiting mechanisms, RAWA algorithm can be applied to stateful servers and nested visiting. The performance of RAWA algorithm is also analyzed, and through experiments on CORBA platform, the comparisons with other fault tolerant algorithms indicate that RAWA has effectively improved the performance of redundant servers without damaging system availability.

Key words: client/server; redundant service; fault tolerance; load balancing; quorum

* Received June 2, 1999; accepted February 29, 2000

Supported by the National High Technology Development Program of China under Grant No. 863-306-ZD02-02-57

© 中国科学院软件研究所 http://www.jos.org.cn