第 34 卷 第 5 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.5 Computer Engineering March 2008
·博士论文· 文章编号：1000—3428(2008)05—0039—03 文献标识码：A 中图分类号：TP311.5

分布式高可扩展负载平衡中间件技术研究

王 俊，郑 笛，吴泉源
(国防科学技术大学计算机学院，长沙 410073)

摘 要：现有的负载平衡中间件大多采用单一的基于对象的负载平衡粒度，更多地关注同一对象组的不同副本之间的平衡，复杂的分布式应用往往存在多种不同类型的服务对象组共存的情况。该文基于 CORBA 技术，论述提高负载平衡中间件可扩展性的关键问题、解决方法及其实现，给出了相关的测试结果。

关键词：负载平衡；中间件；可扩展性；复杂分布式应用

Research of High Scalable Distributed Load Balancing Middleware Technologies

WANG Jun, ZHENG Di, WU Quan-yuan
(School of Computer, National University of Defense Technology, Changsha 410073)

【Abstract】 Most of the current load balancing middleware adopt the per-object load monitoring granularity, and pay more attention to the balance among the different replicas of the same group. This paper discusses several key problems about how to improve the scalability of the load balancing middleware. Implementations and simulation results are given.

【Key words】 load balancing; middleware; scalability; complex distributed application

随着网络和通信技术的发展，节点的计算能力越来越强，分布式应用规模逐渐增大，因此，必须采用负载平衡机制解决应用层的性能及其伸缩性和可靠性问题。分布计算中间件是实现分布计算的关键技术之一，基于中间件的负载平衡有其特殊的优点。本文基于 CORBA 技术设计实现了一种高可扩展的负载平衡中间件模型。

1 背景介绍

为了能够支持大量在线客户对数据的请求并提供不间断的高质量可靠服务，对分布式计算系统伸缩性和可靠性的要求越来越高。通过采用负载平衡机制，可以有效地改善系统的整体性能，可在网络、操作系统、中间件和应用本身 4 个层次上提供负载平衡机制。但基于网络和基于操作系统的负载均衡体系结构在作负载均衡决策时不支持应用定义的负载度量方法，并且因为缺少来自副本的负载反馈，对副本的负载难以进行有效控制。而基于应用的负载平衡与应用核心紧密结合，不具备通用性，代价太高。相反，基于中间件的负载均衡在多方面优于以上层次的负载均衡，当今的企业级关键业务领域绝大多数都采用了基于中间件的分布式软件系统。利用中间件分布计算的优势，对多台计算机的集群提供负载平衡，可以对集群中各类冗余服务进行有效管理，使系统的伸缩性和可靠性最大化。当前，CORBA[1]已经成为分布式应用开发和系统集成的主流技术之一，它为应用开发者屏蔽了分布复杂性，减少了开发代价。

很多已有的中间件实现提供了不同的方法以支持分布式应用负载平衡的需求。例如，需要平衡工作负载的无状态的分布式应用往往采用和名字服务集成在一起的负载平衡服务机制[2]。这种负载平衡机制只支持静态非自适应的负载平衡，不能满足复杂负载平衡应用的需要。而自适应的负载平衡机制，在进行负载平衡决策时能够动态地考虑负载状况，避免发生某个节点负载过重的偏载情况，并使系统获得较好的可扩展性。但在现有的自适应负载平衡中间件实现中[3-5]，负载监测往往是简单地采用基于对象的粒度进行的，负载平衡器着力于保持同一对象组的副本之间的平衡，并不考虑主机上是否存在其他副本的影响。在多种不同类型的服务对象组共存的环境下，传统的负载平衡中间件模型存在以下不足：

(1)重复的负载监测。对于主机上驻留的分布式应用服务对象组来说，在负载度量上往往会存在一种聚集。而对于针对对象粒度的负载监测来说，各个对象对应的负载监测线程频繁地完成同样的任务，形成了一种功能上的冗余，对负载平衡系统的性能造成了不必要的影响。

(2)服务对象组的扩展能力不强。对于某一种负载度量方式来说，服务对象组的增加会影响到已有的对象组，从而使其他多个对象组所拥有的负载度量信息失效，最终导致过载。要避免这种负载不平衡的情况发生，就必须根据服务对象组的增加相应地减小已有对象组的负载监测和收集周期，这些监测线程的频繁调用将大大影响负载平衡系统的性能。

(3)静态的副本管理机制。在松耦合、高分布的服务计算环境下，服务负载可能经常发生波动，甚至出现过载。对于上述的传统的负载平衡中间件来说，往往采用静态、固定的副本管理。当针对某种服务的请求急剧增大时，系统不得不在各副本之间进行负载迁移。然而负载迁移机制并不适用于

基金项目：国家“973”计划基金资助重点项目(2005cb321804)；国家“863”计划基金资助项目(2004AA112020)
作者简介：王 俊(1978－)，男，博士研究生，主研方向：分布式计算技术；郑 笛，博士研究生；吴泉源，教授、博士生导师
收稿日期：2007-03-28 **E-mail：**junwang@nudt.edu.cn

有状态的副本，同时服务副本的数目也难以确定，过多就会造成浪费，过少则不能解决重载问题，还会因为负载的频繁迁移导致系统的不稳定。

2 复杂分布式应用的高可扩展负载平衡中间件

2.1 可扩展的负载监测模式

本文通过 Agent 技术将主机上所有的服务对象组副本组织在一起，形成了虚拟组，如图 1 所示。对应的副本 G1-R1, G2-R1, G3-R1, G4-R1, G5-R1 驻留于同一主机上，通过设定相应的 Agent，各副本根据自己的负载度量与 Agent 相关联构成虚拟组。

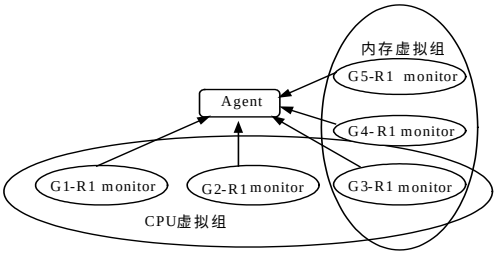

图 1 基于 Agent 的虚拟组

这种基于 Agent 的虚拟组可带来如下好处：

(1)高效的负载监测。对于同一主机上所有的服务副本成员来说，它们对应同一个 Agent，可以通过负载监测周期进行调节。在如图 1 所示的虚拟组中，给 Agent 设置较小的负载监测周期，使之进行实际监测，而各副本所对应的副本检测器则设置较大的负载监测周期，负载信息从 Agent 处获得。通过这种负载监测周期的区分实际上达到了减少监测线程开销的作用，将有效减少系统的额外开销。

(2)较强的服务对象组扩展能力。当向某台主机部署新的服务对象组时，由于有且只有 Agent 在进行负载监测，因此可以保持较小的监测周期，从而使新的服务对象组加入的影响最小化，并且不会由于负载监测线程的增加给负载平衡系统增加额外的开销。

2.2 可扩展的动态副本管理

可扩展副本管理的关键就是根据主机负载控制副本的创建和删除。设 h_j 代表系统的第 j 台主机，则系统节点集合 H 可表示为 $H=\{h_1,h_2,\cdots,h_i\}$。又假设 s_k 代表系统的第 k 个服务，则系统所有的服务集合可表示为 $S=\{s_1,s_2,\cdots,s_l\}$。同时，令 N_k 表示系统中第 k 个服务的副本数量，则 k 个服务的副本集合可以被表示为 $R(S_k)=\{S_{k1},S_{k2},\cdots,S_{kN_k}\}$。相应的，第 l 个服务的第 m 个副本所在的主机节点可表示为 $H(S_{lm})$，并且该主机在 t 时刻的负载可以表示为 $L_{H(S_{lm})}(t)$。

在正常的情况下，新的客户端请求将会派发给最合适的副本。然而，在服务所有副本所驻留的主机可能都处于高负载的情况下，新请求的派发会导致过载的发生，因此，必须创建新副本以分享负载。本文设置了 *replica_addition* 门阈值来帮助进行决策。也就是说，如果所有副本的负载超过了门阈值，将会创建新副本。例如，对于系统的第 i 个服务来说，如果式(1)为真，就创建新的副本。

$$\forall x(L_{H(S_{ix})}(t)\geq replica\_addition)\quad x\in R(S_i) \tag{1}$$

但副本的创建必须遵循最大副本数的限制，因此，如果特定服务达到了最大副本数，就不会创建新副本，这说明整个系统处于较高的负载，而客户端请求只能在派发队列中等待响应。

通过负载度量可以计算出主机的不同负载。本文为每台主机设置了 *replica_deployment* 门阈值，如果某些主机负载超过了该门阈，则新副本不能部署到该主机上，否则将可能导致被部署的主机因过载而失效，整个系统也将变得不稳定。如式(2)，对于系统中所有的 i 台主机节点，如果该表达式成立，则可以在该节点上部署新的副本。因此，本文选择可能存在的具有最合适的主机创建新副本。

$$\exists x(L_{H_x}(t)<replica\_deployment)\quad x\in\{1,2,\cdots,i\} \tag{2}$$

因为应用可能由多种服务组成，并且这些服务可能同时需要创建新的副本，所以应首先创建具有高优先级服务的副本，如果没有新的合适的主机时，具有较低优先级的服务将等待一段时间。图 2 为不同优先级的服务分别保持一条服务副本创建等待队列，根据所采用的队列管理方法进行管理。

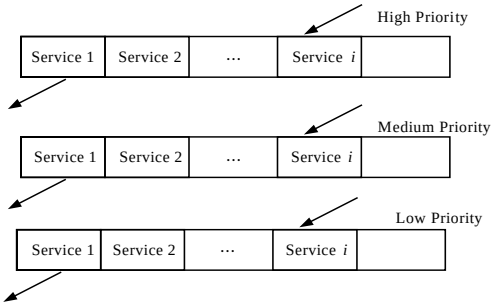

图 2 不同优先级的服务队列

必须避免低优先级的服务因为副本创建请求长期达不到服务而饿死，同时，在某些特殊的情况下，一些普通服务会变成热点内容。对某些服务来说，客户端请求将无规律地进行波动。除了动态管理服务的副本数以外，由于最大服务副本数限制的存在，还须考虑在必要的情况下改变服务的优先级。例如，对于低优先级请求来说，如果在很长时间间隔内创建新服务副本的要求都得不到满足，则须考虑服务优先级的提升。当服务请求大量增加时，低优先级的服务需要获得更高的优先级，以创建更多的副本来满足客户请求的需要。同时，当客户请求下降，热点内容恢复正常，这些服务的优先级将相应降低，多余的副本也将被释放。这些管理都通过所配置规则的监控来完成，而对于某些特殊的应用来说，还可以配置特殊的规则。

因为请求的到达可能是波动的，如果请求的数目比较小，则依然维持多个副本，在这些副本之间进行派发请求和负载计算等控制是没有必要的，将带来额外的开销。应在必要的时候释放多余的副本以减少额外开销，提高系统的效率。因此，本文设定了 *replica_elimination* 门阈值以控制副本的删除。也就是说，当某个副本的负载小于该门阈值时，将有某个负载小于门阈值的副本被释放，直到所有副本的负载都高于门阈值，或副本的数目达到了初始副本数才停止。

3 性能测试

本文的负载平衡中间件 StarLB 的运行环境为：RedHat Linux 9, Pentium4, 1.7 GHz, 256 MB 内存。服务主机和客户端的运行环境为：Windows 2000, Pentium4, 2 GHz, 512 MB 内存，通过百兆交换机相连接。负载平衡试验平台如图 3 所示。

首先测试基于 Agent 的负载监测机制的有效性。分别考虑 2 台服务器和 4 台服务器的情况，每台服务器上预计部署 8 种不同的服务，每台主机上均部署 1 个服务副本，同时为了简化测试，这些服务都是运算密集型的服务，主机的负载

主要来源于 CPU 的消耗。实验初始阶段每台服务器上只部署 Service1 的副本，而后逐渐开始部署新的服务，同时记录 service1 的平均响应时间。

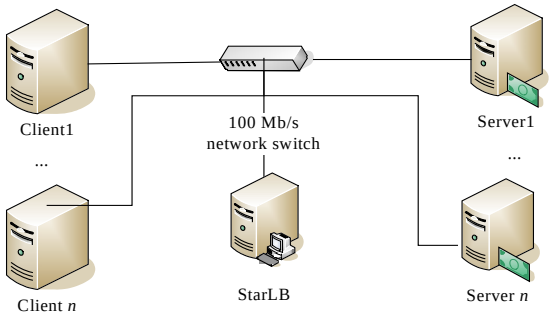

图 3　负载平衡试验平台

如图 4 所示，随着越来越多的副本被部署到主机上，平均响应时间将因为更多的客户端请求和负载检测而逐渐增加。可以发现，通过使用 Agent，平均响应时间会比不使用 Agent 的情况相应地降低，这也意味着检测所带来的额外开销被有效地降低了，可以灵活地只对 Agent 负载报告间隔参数进行调控。

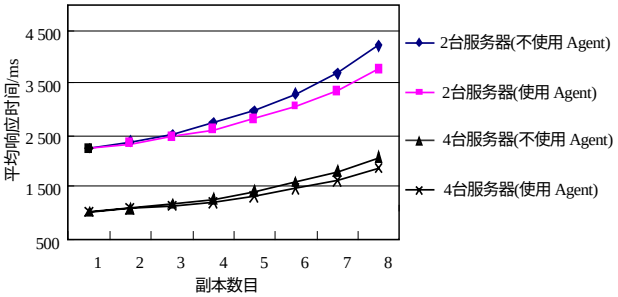

图 4　高效的负载监控

然后对动态可扩展的副本管理机制进行测试。为简化说明，选用同一优先级的服务，服务数和主机节点数都为 3。*replica_addition* 门阈值设为 85%，*replica_deployment* 门阈值设为 70%。另外，*replica_elimination* 门阈值设为 50%。服务副本的初始设置如图 5 所示，其中，S1 代表 Service1；S2 代表 Service2；S3 代表 Service3。

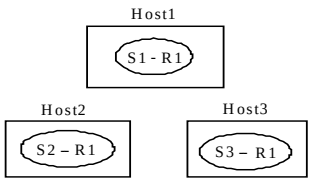

图 5　同一优先级时的初始副本设置

如图 6 和图 7 所示，使 Service2 和 Service3 的请求保持稳定，同时逐渐增加针对 Service1 的请求。可以看出，当 Host1 的负载超过 85%时，将会在 Host2 上创建 Service1 的一个新副本。因此，针对 Service1 的高负载能够在一定程度上得到缓解。继续增加针对 Service1 的请求，则将在 Host3 上进一步创建一个新的副本。当负载稳定后，逐步减少针对 Service1 的负载，各主机节点的负载也相应减少。当 Host1 的负载低于 50%，Service1 的一个副本将被释放从而避免过多的额外控制开销，保证系统的效率。上述的副本创建和释放的过程如图 8 所示，其中，S1 代表 Service1；S2 代表 Service2；S3 代表 Service3。

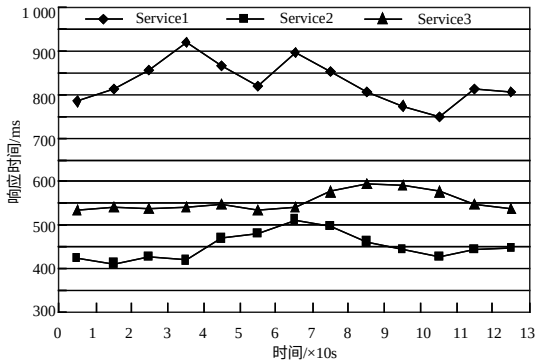

图 6　平均服务响应时间

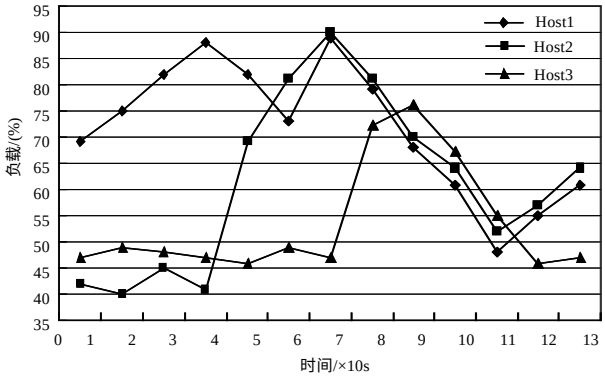

图 7　主机负载变化

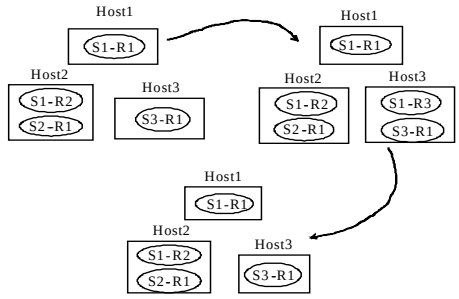

图 8　副本的创建和释放

在上述测试的基础上，本文还进行了不同服务优先级条件下和热点内容优先级提升的测试，测试结果表明，副本管理的可扩展性良好，能够很好地达到动态副本创建、删除的要求，对分布式应用的动态负载平衡和过载控制能够起到很好的作用。

4　结束语

分布式应用的拓扑结构日益复杂，应用规模日益增大，现有负载平衡中间件的功能已不能完全满足复杂分布式应用的可扩展需求。本文针对已有工作的不足，对分布异构环境下高可扩展的负载平衡中间件模型展开了研究。实验表明，该模型可以有效提高系统整体的可扩展性、资源利用率以及在高负载条件下的稳定性。下一步将研究分散式负载平衡服务效率的提高、面向延时和节点失效的负载平衡算法以及有状态的副本管理等问题。

参考文献

[1] Object Management Group. The Common Object Request Broker: Architecture and Specification[Z]. 1999-06.

[2] IONA Technologies. Orbix 2000[Z]. [2007-03-28]. http:// www.iona-iportal.com/suite/orbix2000.htm.

（下转第 50 页）