PKUAS 中负载平衡机制的设计与实现

郑子瞻, 王千祥, 黄 罡, 梅 宏
(北京大学信息科学技术学院软件研究所, 北京 100871)

摘 要: 尽管在 J2EE(Java 2 platform, Enterprise Edition) 规范中尚未明确提出与负载平衡相关的公共服务, 为了提高网络软件的伸缩性, 在中间件层次上支持负载平衡已成为 J2EE 应用服务器提供商事实上必须关注的内容. 本文在分析现有的面向 J2EE 平台的负载平衡技术的基础上, 提出了一种支持多种负载平衡策略、可灵活定制的负载平衡框架, 并详细描述了该框架在一个产品化的 J2EE 应用服务器 PKUAS(Peking University Application Server) 中的实现细节.

关键词: 负载平衡; 中间件; J2EE; EJB
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2004) 12A-180-05

Design and Implementation of the Load Balancing Mechanism in PKUAS

ZHENG Zi-zhan, WANG Qian-xiang, HUANG Gang, MEI Hong
(*Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing* 100871, *China*)

Abstract: J2EE specifications haven't yet definitely defined any common service which provides load balance. To meliorate the scalability of network software, however, supporting load balance at middleware level has become an important issue which should be focused on by all J2EE application server providers. In this paper, we first analyze load-balance related technologies currently used on J2EE platforms. After that, we present a flexible and customizable load balancing framework which supports several load balancing strategies and describe the implementing details of the framework on a J2EE application server PKUAS.

Key words: load balance; middleware; J2EE(Java 2 platform, enterprise edition); EJB(enterprise Java beans)

1 引言

随着 Internet 技术的不断发展, 提高网络软件的伸缩性(scalability) 及可用性(availability) 正受到越来越多的重视. 负载平衡与容错为这两个问题提供了典型的解决方案. 随着中间件技术的广泛运用, 以中间件服务的形式支持负载平衡的做法日益普遍. 与在操作系统或网络层实现负载平衡相比, 在中间件层实现负载平衡的优点是提供了更大的灵活性[1], 这主要表现为可以基于应用(负载) 类型进行负载平衡决策. 目前, 主流的中间件技术如 J2EE 与 CORBA(Commom Object Request Broker Architecture) 等都对负载平衡提供了支持, 并且提供了多种负载平衡算法与负载度量标准供选择[1,2]. 然而, 现有的实现往往仅支持单一的负载平衡点与负载平衡粒度, 尚不能很好地适应各种不同的应用环境. 本文首先探讨了在 J2EE 平台上实现负载平衡的几个关键问题, 分析了各种负载平衡策略各自适用的场合, 而后详细描述了一种可灵活定制的负载平衡框架在一个产品化的应用服务器 PKUAS 中的实现.

PKUAS 是一种符合 J2EE1.3 规范的反射式构件运行支撑平台[3]. PKUAS 基于 JMX 实现了微内核的体系结构, 支持四类容器系统(无态会话容器、有态会话容器、实体容器和消息驱动容器) 与三种互操作协议(IIOP(Internet Inter-Orb Protocol)、JRMP(Java Remote method Protocol) 与 SOAP(Simple Object Access Protocol)), 并内置了命名服务、安全服务、事务服务、日志服务、数据连接服务与集群服务.

2 J2EE 平台上的负载平衡

J2EE 平台以集群的形式同时为负载平衡与容错提供支持. J2EE 集群由一组应用服务器组成, 透明地为客户提供企业级服务. 集群又称为组, 这是因为集群一般使用组通信进行成员管理, 并保持各节点状态的同步. 这里所说的状态既包括实现负载平衡需要的对象(引用) 的副本, 也包括实现容错所需的会话状态等. 为支持企业级计算, J2EE 平台采用了分层的体系结构. 相应的, 可以在不同层上实现负载平衡:

(1) Web 层的负载平衡[4] 常见方案有: (a) DNS 轮循法, 即由 DNS 服务器维护域名与一组(集群内节点的) IP 地址的映射, 对于每一个域名解析请求, DNS 服务器依次选择列表中的下一个 IP 地址返回; (b) 硬件负载均衡器法, 即整个集群

收稿日期: 2004-09-09; 修回日期: 2004-11-04
基金项目: 国家重点基础研究发展规划项目计划(973 计划) (No. 2002CB312003); 国家自然科学基金(No. 60125206, No. 60103001, No. 60303004); 国家高技术研究发展计划(863 计划) (No. 2004AA112070)

对外而言只有一个 IP 地址(即均衡器的地址),所有请求都发往该地址,由均衡器根据选定的负载平衡算法转发请求;(c) Web 服务器代理法,即由 Web 服务器(可以实现为 HTTP 服务器上的插件)或部署于其上的 Web 应用实现对 Servlet/JSP 请求的负载平衡.

(2) EJB(Enterprise Java Beans) 层的负载平衡　由 J2EE 应用服务器通过增强命名服务与通信服务实现对 EJB 及公共服务(如 JNDI(Java Naming and Directory Interface)、JDBC(Java DataBase Connectivity)、JMS(Java Message Service) 等)的负载平衡.

(3) 数据层的负载平衡　一些数据库供应商提供数据库集群产品,支持不同数据库服务器间的数据复制,透明(相对于使用数据库的 Web 服务器与 EJB 服务器)地实现了 JDBC 连接在不同数据库节点间的负载平衡.

本文主要关注 EJB 层的负载平衡.

2.1　负载平衡点

EJB 层的负载平衡决策可以在不同位置做出,如负载均衡器、应用服务器(JNDI 命名服务器、容器等)以及客户代理(home stub 与 remote stub)等.

由负载均衡器转发请求的方案(图 1)有明显的缺陷. 首先,为了避免单点失效与性能瓶颈,一般都使用几个负载均衡器. 然而即便如此,仍然不能避免所有均衡器均失效而导致系统瘫痪(即使服务节点仍可用)的情况. 此外由于每次请求都需由均衡器转发,不可避免的会带来性能开销.

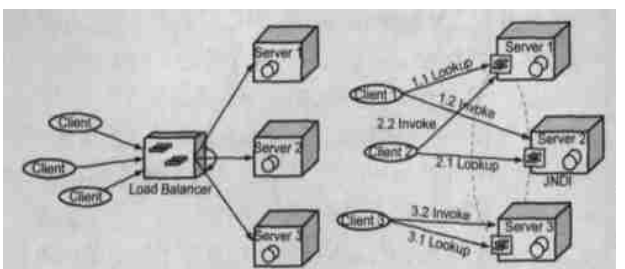

图 1　负载均衡器均衡负载示意图　图 2　命名服务均衡负载示意图

为了克服上述缺点,本文提出了一种由命名服务实现负载平衡的方案(图 2). 命名服务维护了 JNDI 名与一组对象引用(或客户代理)的映射,在执行查找请求时根据一定的负载平衡算法选择一个特定于某个节点的对象引用返回给客户. 由于命名服务位于集群的每个节点上,一般不会出现上述的第一种情况. 同时由于无需第三方转发请求,因而降低了性能开销. 当然,由此带来的一个问题是仅能支持较粗粒度的负载平衡(见 2.2 节). 这种方案的优点是实现比较简单,运行时刻开销小. 此外,由于命名服务比较容易获取全局信息(如各节点的负载状况),因而易于实现动态负载平衡算法. 服务器端负载平衡还可由容器或通信服务中的接收器实现,但这种做法进一步加重了服务器的负担,一般很少采用.

上述两种方案的共同之处是由服务器实现负载平衡(负载均衡器也应视为一种服务器),这意味着随着客户数量的增加,服务器用于负载平衡的开销也会相应增加. 对于以事务处理为主的应用领域(如 CORBA 应用),由于并发客户数较少且客户请求的处理时间较长(因而负载平衡开销所占的比例较小),这类方案是合适的. 而对于基于 Web 的应用领域,常常需要面对大量客户,并且所处理的多为交互式请求,则需要考虑降低服务器用于负载平衡的开销.

正是由于上述原因,由客户代理转发请求(图 3)成为 J2EE 平台上最广泛采用的实现机制[2,5,6]. 客户代理维护了一组对象引用,其内封装了部署了相应 EJB 或提供相应服务的一组节点的地址. 每次方法调用中,客户代理根据选定的负载平衡算法及其它信息(如各节点的负载)选择一个对象引用,向相应的节点发出请求. 这种实现机制的优点是:(1) 具有较好的可伸缩性. 由于负载平衡开销主要由客户负担,服务器无需考虑由于客户数量增加引起的负载平衡开销的增加;(2) 支持细粒度的负载平衡(见 2.2 小节). 然而,客户代理需要从服务器获取必要的信息以及细粒度负载平衡都意味着运行时刻开销的增加. 此外,需要为每一种互操作协议开发支持负载平衡的客户代理也意味着实现难度的增加.

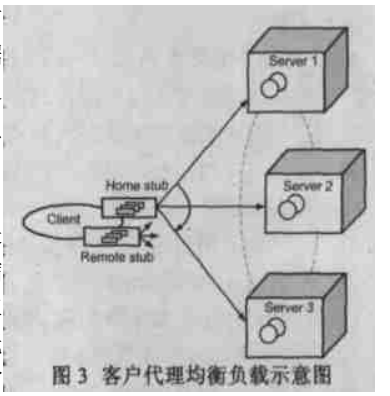

图 3　客户代理均衡负载示意图

考虑到 2、3 两种方案各有特定的适用环境,PKUAS 同时实现了两种方案. 本文将在第三部分详细描述实现细节.

2.2　负载平衡粒度

客户使用 EJB 的一次典型流程如图 4 所示. 首先,客户从命名服务查找 home 对象,得到其客户代理(home stub). 而后客户通过 home stub 调用 home 对象的 create 方法,创建一个 remote 对象,得到 remote 对象的客户代理(remote stub). 最后客户通过 remote stub 调用 remote 对象的某个方法,完成实际的业务操作. 本文认为,对于 EJB 而言存在以下三种负载平衡粒度:

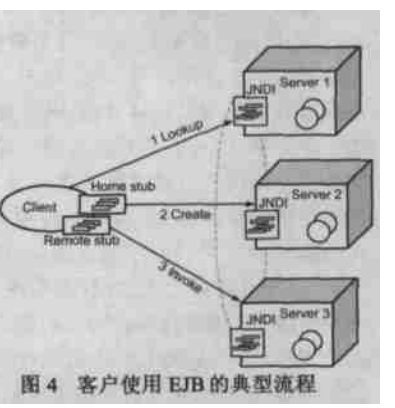

图 4　客户使用 EJB 的典型流程

(1) 查找级**(per lookup)**　向一个 home 对象以及该 home 对象创建的所有 remote 对象发出的请求都由一个固定的节点处理,仅支持同一个 EJB 的不同 home 对象及不同 EJB 的负载平衡. 其效果可由上述的命名服务转发请求实现,故名.

(2) 会话级**(per session)**　向 home 对象发出的不同请求可以负载平衡,而向 remote 对象发出的请求则始终由创建该对象的节点处理. 有态会话构件一般只能做到这一级,故名.

(3) 请求级**(per request)**　home 与 remote 对象的请求都可以负载平衡. 一般只有无态会话构件与只读实体构件能做到这一级. 尽管会话状态复制为有态会话构件的请求级负载

平衡提供了支持,但是考虑到性能开销,在实际应用中会话状态复制一般仅用于需要支持容错的情况.

从级别 1 到级别 3,负载平衡粒度逐渐变细.一般来说,粒度越细,负载平衡效果越好,但运行时开销也越大.2.1 小节提到的方案 2 支持查找级,方案 3 支持会话级与请求级.本文将在第三部分描述这三种负载平衡粒度在 PKUAS 中的实现与定制方法.

2.3 负载平衡算法

根据决策时是否使用运行时刻的信息(如各节点的负载状况),负载平衡算法可以分为静态算法与动态算法两大类.静态算法不使用运行时刻的信息,常用的静态算法有轮循、随机及静态加权算法等几种.其中轮循与随机算法适合于所有节点处理能力相当的情况.静态加权算法适合于节点处理能力不同的情况.动态算法也称为适应性算法,根据运行时的系统状态信息做出负载平衡决策.理论上,动态算法具有更大的灵活性,其实际效果则依赖于所选择的负载分发策略、负载度量标准以及运行时数据的获取[1].目前主流的 J2EE 应用服务器一般只支持静态算法[5,6],而 CORBA 产品则大都支持动态算法.这同样与两类中间件主要面向不同类型的应用有关.目前,PKUAS 仅支持三种静态负载平衡算法,并且支持在运行时刻替换缺省的负载平衡算法.

3 负载平衡机制在 PKUAS 中的实现

3.1 PKUAS 负载平衡框架概述

如前文所述,J2EE 平台上的负载平衡涉及到许多相关的策略(不同的负载平衡点、负载平衡粒度以及算法),这些策略各有其特定的适用环境.为了同时支持多种策略并允许客户根据实际需要加以选择,PKUAS 实现了一个可灵活定制的负载平衡框架.PKUAS 负载平衡建立在 PKUAS 互操作框架[7]与 JGroups[8]提供的组通信服务之上.JGroups 是基于 Java 的开源组通信中间件,被广泛运用于各种 J2EE 平台的开源实现中.JGroups 提供了可靠有序的组通信、组成员管理以及组内 RPC 等特性.在此基础之上,PKUAS 实现了集群服务、全局命名服务,并在针对 IIOP 互操作协议的客户代理中增加了对负载平衡的支持,从而实现了上述的两种方案.目前,PKUAS 仅支持使用 RMI/IIOP 互操作协议的 EJB 的负载平衡.

3.2 集群服务

集群服务负责在 PKUAS 启动时加入组,监听 JGroups 发出的消息以及进行消息转发等.集群服务允许其它服务注册所关心的消息,并在收到消息时通知相应服务.其它服务可以通过集群服务发出组内 RPC 调用,从而实现组内节点的状态同步.为了使 EJB 容器可以在方法调用前后对集群服务进行必要的控制,在 PKUAS 现有的截取器链中增加了集群截取器.目前,集群截取器仅在调用返回前完成两项工作:从集群服务获得当前视图 id,插入调用上下文中;对于返回 remote 对象的 home 方法,从构件元数据获取 remote 接口的负载平衡策略,并将其封装为对象引用的一部分.

3.3 全局命名服务

在 PKUAS 中,命名服务维护了 JNDI 名字与 IOR(Inter-Operable object Reference)(可互操作对象引用)的映射,而全局命名服务则维护了所有需要负载平衡的 EJB(及 RMI 对象)的 JNDI 名字与一组 IOR 的映射.

对于 EJB,PKUAS 会在应用部署时刻为其生成 home 接口的 IOR,并将 IOR 绑定到本地命名服务上,对于需要负载平衡的 EJB 则还会将其绑定到全局命名服务上.全局命名服务通过集群服务将名字信息组播到组内其它节点上,以保持状态同步.在查找过程中,如果全局命名服务在自身维护的绑定信息中找不到相应的对象引用,则首先将请求转发给本地命名服务,如果仍然查找不到,则通过集群服务将请求转发给组内其它节点的全局命名服务.当得知某个节点失效时(由集群服务通知),全局命名服务将属于该节点的名字信息删除.

对于需要查找级负载平衡的构件,全局命名服务在查找时根据一定的负载平衡算法选择相应的对象引用返回给客户.对于需要会话级与请求级负载平衡的构件,全局命名服务在查找时将对应于 JNDI 名的一组仅含单个地址的 IOR 重组为一个含多个地址的 IOR 返回给客户.

3.4 支持负载平衡的 IIOP 客户代理及相关设施

PKUAS 实现了开放的互操作框架[7],自主开发了符合 IIOP 规范的 RMI-IIOP 互操作协议.为了实现在 IOR 内封装多个节点的地址与端口号,引入了标准 IOR 组件 TAG_ALTERNATE_IIOP_ADDRESS.对于需要会话级与请求级负载平衡的构件,全局命名服务返回一个包含此组件的 IOR 给客户,保存在 home stub 中.Home stub 在创建 remote 对象的方法中将自身维护的地址列表复制到 remote stub 中.为了实现 2.2 小节所述的各级负载平衡,stub 需要对 IOR 中的地址列表进行不同的处理.对于会话级负载平衡,home stub 的每次方法调用可以选择地址列表中的不同地址,remote stub 则仅使用创建该对象的 home 方法所用的地址;对于请求级负载平衡,home stub 与 remote stub 都可以使用不同的地址.

为了实现请求转发,stub 必须获得特定于该对象的负载平衡策略与其它相关信息,考虑到 IIOP 规范中并无相应的标准组件,引入了自定义 IOR 组件 TAG_LOADBALANCE_POLICY.其内封装了负载平衡粒度、算法、组成员视图 *id* 以及各节点的静态权.负载平衡算法与各节点的静态权分别来自部署描述文件与服务配置文件(见 3.6 小节).组成员视图 *id* 从调用上下文获得.当 stub 发现当前的视图 *id* 与自身保存的 id 值不一致时,将从全局命名服务获取包含了最新节点地址列表的 IOR,从而避免了不必要的容错开销.

3.5 负载平衡中的安全问题

PKUAS 安全服务[9]支持"单次登录"(Single Signon):客户在执行首次 EJB 业务操作前,要到认证服务器显式登录一次,认证服务器为客户分配一个令牌作为临时身份凭证,并将该令牌加入自身维护的一个缓存中.以后客户的每次请求都将携带该令牌,由安全截取器调用认证服务检查令牌的合法性.在集群环境中,由于客户的每次请求可能由不同节点处理,为了保持"单次登录"的特色,认证信息需要在集群范围内共享.为此将认证服务注册到集群服务中,当新的令牌加入缓存或原有令牌失效时,通过组内 RPC 维护各节点认证信息的同

步. 具体来说, 当客户在某个节点登录后, 该节点的认证服务为客户分配一个令牌并加入自身的缓存, 同时通过组内 RPC 将该令牌加入组内其它节点的认证服务缓存中. 而后携带该令牌的客户可以在组内的任意节点上通过身份认证, 而无需重新登录.

3.6 负载平衡相关策略的定制

PKUAS 基于微内核的体系结构允许对服务进行灵活定制. 就负载平衡而言, 用户可以选择使用或者不使用负载平衡, 可以选择不同的负载平衡粒度和负载平衡算法. 目前, 除缺省的负载平衡算法可以在运行时刻设置外, 其它配置工作需要在部署时刻完成. 下文将依次加以说明.

管理员可以通过修改 PKUAS 的服务配置文件来启动或关闭负载平衡服务. 客户可以通过使用不同的客户代理(由代理生成器自动生成) 以及命名服务(本地或全局) 以决定是否使用负载平衡服务. 部署人员可以在特定于应用的部署描述文件中为每个 EJB 指定负载平衡策略, 如图 5 所示. 其中负载平衡算法是可选的, 如果没有提供, 则使用缺省的负载平衡算法. Level0~ 2 分别代表查找级、会话级与请求级. 此外为了支持静态加权算法, 部署人员需要在服务配置文件中指定本节点的静态权.

```
〈ejb〉
    〈ejb-name〉OrderSes〈/ejb-name〉
    〈ejb-type〉Session〈/ejb-type〉
    〈jndi-name〉OrderSes〈/jndi-name〉
    〈cluster〉
      〈clustered〉true〈/clustered〉
      〈level〉2〈/level〉
      〈home〉RoundRobin〈/home〉
      〈remote〉RoundRobin〈/remote〉
    〈/cluster〉
…
〈/ejb〉
```

图 5　EJB 负载平衡策略的配置

PKUAS 提供了一个基于 Web 的管理工具, 该工具从 PKUAS 内核(通过反射) 获取服务及应用构件的元信息, 并允许管理员在运行时刻对一些配置参数进行调整. 对集群服务而言, 管理员可以从管理界面看到节点的静态权、组名等信息, 并可调整缺省负载平衡算法, 从而使负载平衡服务对环境的变化具备一定的适应性.

3.7 性能评测

测试 1 与测试 2 评估了上述框架对单个节点的响应时间产生的影响. 测试环境如下: 两台配置为 P4 2.4G, 512M 内存的 Dell PC, 其中一台作为服务器, 安装了 RedHat9 操作系统, 一台作为客户端, 安装了 Windows XP 操作系统. 节点间通过 100Mbps 以太网联接.

测试用例是一个简单的无态会话 EJB 和一个 standalone 的 Java 客户程序. 该 EJB 接受客户传来的一个字符串, 并将其直接返回给客户. 该 EJB 部署在单个服务节点上, home 与 remote 接口都使用轮循算法. 客户程序依次向服务器发送不同字节数的字符串(各 1 万次, 以避免测试结果的随机性), 并记录响应时间. 三条曲线分别代表关闭负载平衡、命名服务转发请求(查找级) 与客户代理转发请求(请求级) 三种情况. 在测试 1 中, 客户仅在程序开始运行时调用 JNDI 命名服务. 在测试 2 中, 客户在每次调用前都进行查找与创建工作. 图 6 与图 7 分别给出了两种情况下的测试结果. 当选择命名服务作为负载平衡点时, 负载平衡机制引起的响应时间开销仅存在于每次查找过程中, 可以看到其影响是比较小的(对于一次查找、多次调用的情况几乎可以忽略不计) . 当选择客户代理作为负载平衡点时, 由于每次调用都需要运行负载平衡算法分派请求, 对响应时间有一定影响.

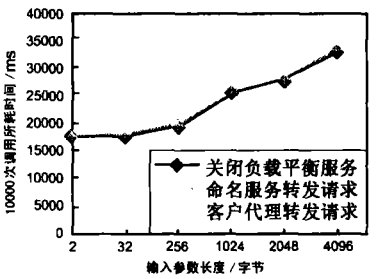

图 6　响应时间测试 1(一次查找, 多次调用)

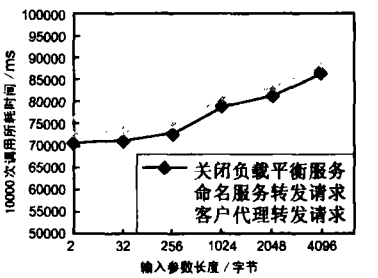

图 7　响应时间测试 2(一次查找, 多次调用)

4 相关工作

当前主流的应用服务器都提供了对负载平衡的支持. 考虑到 J2EE 应用的特点, 大多数实现(如 JBoss、JOnAS 与 WebLogic 等) 均选取客户代理作为负载平衡点.

JBoss 作为当前最成功的开源应用服务器为负载平衡与容错提供了全面支持. JBoss 集群[5] 使用 JGroups 作为底层的组通信基础设施, 实现了集群服务、高可用命名服务以及特定于 JRMP 互操作协议的客户代理及相关设施. 与 PKUAS 相比, JBoss 负载平衡的优点在于支持分布热部署, 即仅需在集群的一个节点上部署需要负载平衡的构件, 热部署过程自动将该构件部署到集群的所有节点上. 此外, JBoss3.2 以上版本支持代理族(ProxyFamily) . 同一个 EJB 接口的位于同一个 JVM 内的所有客户代理组成一个代理族, 可在其内共享一些信息(如视图 id, 地址列表等), 从而使集群动态变化时客户代理的更新工作得以简化, 并为某些特殊的负载平衡策略提供了支持. 其不足之处在于, 由于 home 与 remote 接口的客户代理各自独立地实施负载平衡算法, JBoss 不能支持查找级与会话级负载平衡(有态会话构件除外) . 此外, JBoss 不支持运行时刻修改缺省的负载平衡算法.

作为当前最成功的商业应用服务器产品之一, WebLogic 提供了功能更强大的集群服务[7] . 与 PKUAS 相比, WebLogic 除支持 2.3 小节所述的三种负载平衡算法外, 还支持各种服务亲和(Server affinity) 策略, 同时允许用户对负载平衡行为进行较低层的控制. 此外, 除支持 EJB 调用的负载平衡外, WebLogic 还支持 JDBC 对象(JDBC 数据源、连接池) 以及多个 JMS 服务器的负载平衡. 其不足之处在于, 尽管 WebLogic 集群服务支持运行时刻修改集群的缺省设置, 但是需要在应用重新启动后才能生效, 因此在对集群的配置管理上不如 PKUAS 灵活.

5 结束语

本文提出了一种支持不同的负载平衡点、负载平衡粒度

与负载平衡算法,可灵活定制的负载平衡框架,并详细描述了其在 PKUAS 中的实现细节与定制方法.进一步的工作主要包括四个方面的内容.首先需要对 EJB 层的容错提供支持.当前主流的解决方案是由客户代理实现容错逻辑,由集群服务负责状态复制,并且大多采用 in-memory-replication 策略,而不是借助于文件或数据库来保存状态.为了降低性能开销,一般采用主副本技术(如 WebLogic)或将集群划分为子分区,在子分区内进行状态复制(如 JBoss).然而,这些技术都不可避免地会带来因多米诺效应引起的性能下降或负载不均[10].ROC(Recovery Oriented Computing)[11]项目提出的 SSM(Session State Management)技术为会话状态复制提供了一种新思路.其次,需要扩大负载平衡的范围.一方面在 EJB 层需要支持除命名服务外的其它公共服务(如 JMS、JDBC 等)的负载平衡.另一方面需要支持 Web 层的负载平衡与容错.第三,需要进一步提高框架的灵活性.例如,需要支持构件在集群范围内的动态部署与卸载,全面支持各种负载平衡策略的动态切换等,并提供相应的管理界面.最后我们希望为 IIOP 以外的其它互操作协议提供负载平衡支持,同时考虑支持多种负载度量标准,并在此基础上开展与网格计算有关的研究.

参考文献:

[1] Jaiganesh Balasubramanian, Douglas C Schmidt, et al. Evaluating the performance of middleware balancing strategies[A]. Submitted to the 8th International IEEE Enterprise Distributed Object Computing Conference[C]. IEEE Computer Society, 2004. 135- 146.

[2] Abraham Kang. J2EE Clustering [Z/OL]. http://www.javaworld.com/jw-02-2001/jw-0223-extremescale.html.

[3] 黄罡,王千祥,曹东刚,梅宏.PKUAS:一种面向领域的构件运行支撑平台[J].电子学报,2002,30(12A):1938- 1942.

[4] Vivek Viswanathan. Load Balancing Web Applications[Z/OL]. http://www.onjava.com/pub/a/onjava/2001/09/26/load.html.

[5] Sacha Labourey, Bill Burke, The JBoss Group. JBoss Clustering[Z]. JBoss Group, 2002- 09- 27.

[6] BEA System Inc. Using WebLogic Server Clusters[Z]. Version 8.1 Revised, October 20, 2003.

[7] 吴翔,黄罡,郑子瞻,王千祥,梅宏.一种支持多协议的开放互操作框架[J].电子学报,2003,31(12A):2113- 2118.

[8] The JGroups Project[Z/OL]. http://www.jgroups.org.

[9] 曹东刚,王千祥,刘天成,梅宏.PKUAS 的安全机制[A].全国软件与应用学术会议[C].北京:机械工业出版社,2002,303- 310.

[10] Benjamin C Ling, Emre Kiciman, Armando Fox. Session state: Beyond soft state[A]. Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation[C]. San Francisco, CA: USENIX Association, 2004. 295- 308.

[11] The Berkeley/Stanford Recovery-Oriented Computing (ROC) Project [Z/OL]. http://roc.cs.berkeley.edu/.

作者简介:

郑子瞻 男,1979 年 10 月生于江苏,北京大学计算机科学技术系硕士研究生,主要从事软件复用、软件构件和分布计算技术等方面的研究.E-mail: zhzzh@sei.pku.edu.cn.

王千祥 男,1970 年 2 月出生于山东莱州,博士,北京大学计算机系副教授,主要研究领域为软件工程,网络计算环境.E-mail: wqx@cs.pku.edu.cn.