基于CORBA集群进程管理服务的研究与设计*)

孙 明 周明天 詹瑾瑜
(电子科技大学计算机科学与工程学院 成都610054)

Research and Implementation of Process Management Based on CORBA Cluster

SUN Ming ZHOU Ming-Tian ZHAN Jin-Yu
(Computer Science and Engineering College UESTC Chengdu 610054)

Abstract As one of the popular interoperability technology, CORBA provides a distributed computing environment for enterprise application system. Although CORBA makes the development of the distributed object application more convenient and rapid, it does not define a separate process management service to manage and control the server processes of the ORB domain. Conforming to the CORBA 2.x specification, this paper analyzes the features and the basic mechanisms of the process deployment for cluster servers in the distributed computing environment, and presents a design of process management model by integrating SAD and SMD. In particular, the dynamic extension technology and how SMD locates- SAD are also discussed in the paper.

Keywords CORBA, Cluster, Distributed computing environment, Process management , Locality-aware request destribution

1 引言

CORBA[1]作为一种流行的中间件平台,通过定义一系列规范屏蔽了异构网络的底层差别,为灵活开放的系统应用提供了技术基础,使分布计算环境中的构件可以很好地协同工作.然而现有的CORBA 2.X规范中并没有定义一种独立的服务,来管理和控制客户请求访问服务器对象(服务器进程)的过程,以灵活简便地部署企业应用系统服务器集群.传统CORBA客户访问服务器一般是通过对象引用文件或直接网络访问的方式,前者多适用于单服务器系统,并且和系统结合紧密而且很难在系统运行过程中动态扩展服务器,而后者则使用复杂,在一定程度上丧失了服务器的"透明性".本文根据CORBA规范和分布计算环境中进程部署的基本特点,提出一种ORB之上基于CORBA集群的进程管理服务,使得客户可以简单、"透明"地访问集群中的服务器对象.这种进程管理实现了服务器集群系统的负载均衡、提高了服务器进程的执行效率,同时在保证系统中运行的服务器不受影响的情况下可以动态扩展服务器.

2 进程管理服务体系结构

2.1 分布计算环境中进程部署的特点

企业应用系统中部署服务器集群,具有以下一些特点:

·分布异构环境:企业本身在地理上是分布的,使用的硬件、操作系统、网络和协议也是千差万别的,必须集成分布异构环境下的多个服务器,组成一个逻辑上整体的大型系统.

·可伸缩性:系统由多个或多组服务器组成服务器集群,以一致的接口为客户提供服务;服务器的数量和组成情况可以根据需要动态变化,适应不同规模企业的需要.

·高可用性:开发和部署的系统是稳定的;系统具有自检测、自恢复能力;可在不停机的情况下向系统增加新的服务器、新的构件和新的业务.

·灵活性:支持多种服务器策略,满足不同客户、不同应用的要求.

·简化部署、管理与维护:有统一的管理服务来管理庞大的服务器集群;系统具有灵活的、丰富的配置环境,允许应用和业务对象在不同服务器、不同域的主机上移动,而不影响它们的客户.

2.2 进程管理服务的设计思想

在上述特点中,CORBA2.X规范本身只能解决分布异构平台的问题,其余各部分需要具体ORB的实现来提供.

我们提出一种基于CORBA集群进程管理服务的设计方案,对系统中整个ORB域范围内的服务进程提供管理和控制的功能.该方案将进程管理[4,5]看成是ORB之上的一层,以协调各进程之间关系的一种独立系统服务.整个进程管理服务由两种不同的CORBA应用服务器组成:SMD(Server Management Daemon,服务器管理守护进程)和SAD(Server Activation Daemon,服务器激活守护进程).其中,SAD负责按客户需要动态启动以及关闭服务器进程,而SMD负责在ORB域范围内对SAD动态激活的服务器进程进行定位.所有部署了CORBA服务器,并向相同SMD注册的计算机系统,组成一个ORB域.

2.2.1 服务器激活守护进程SAD SAD是一个基于CORBA体系结构的系统服务器,用来激活CORBA应用服务器进程,定位服务对象,管理进程策略以及实现服务器动态调度等功能.在部署了CORBA系统的每台计算机上,都需要有一个激活的SAD负责管理整个服务器进程的生命周期.

SAD中包含两个重要的Harsh表:对象注册信息表和活跃对象列表.对象注册信息表中存放了包含对象实现的服务器进程信息,客户请求可以在SMD获得SAD对象引用后,根据该表的具体内容来激活需要的服务器进程.而活跃对象列表中存放了所有SAD激活的服务器进程中的活跃对象,SAD通过对该表信息的判断可以灵活地、动态地启动和关闭服务器进程.由于SAD全权负责服务器进程的启动和关闭,使得客户方与系统管理员从复杂的服务器管理中解脱出来,

*)本文得到四川省重点科技攻关项目(SG95.17.1)基金资助.孙 明 硕士,主要研究领域为分布对象技术、Web技术等.周明天 教授,博士生导师,主要研究领域为计算机网络、分布对象技术、并行处理等.詹瑾瑜 硕士,主要研究领域为网络多媒体技术、嵌入式实时操作系统等.

这特别适合大型系统的部署和开发。

2.2.2 *服务器管理守护进程* SMD SMD也是一个基于CORBA/ORB体系结构的系统服务器，用在整个ORB域范围内定位和管理服务进程。SMD为客户提供了访问ORB域中CORBA服务器集群的统一入口，且具有智能Agent的作用：根据客户的请求，定位一个合适的服务器，由服务器计算机上的SAD自动启动一个服务器进程，获得其中服务对象的对象引用，返回给客户方。

SMD中包含SAD管理表，用来存放集群系统中激活的SAD信息。当SAD启动时，根据系统的SMD配置文件定位SMD，并向SAD管理表注册。SMD支持冗余和负载均衡，当客户数量增多时，通过应用构件技术以及SMD/SAD的级联，系统可以动态增加服务器而不影响已有系统的运行。

2.2.3 *SMD和SAD的集成* SMD和SAD的集成可以共同完成系统进程管理的任务。服务器运行信息，包括池ID、对象名、对应可执行程序、参数、端口号、服务器策略等，先通过SADUtil(负责服务器对象信息注册的工具程序)在SAD对象注册表中注册。SMD提供给客户方唯一的入口，当客户请求服务器对象时SMD根据请求信息定位一个SAD，并启动服务器进程，获得服务器方对象引用返回客户方。

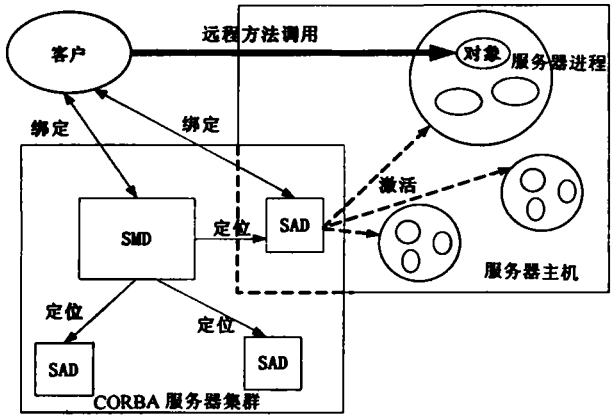


图1 SMD/SAD系统进程管理过程

2.3 通过SMD/SAD访问服务器对象过程

当CORBA客户通过SMD/SAD访问集群中的服务器对象时，首先应该满足：网络上至少启动一个SMD以及集群中每个服务器上都已经启动SAD。然后通过以下步骤获得服务器对象引用：

1)客户访问SMD：通过ORB::bind(Repository)方法建立和SMD的绑定。

2)在SMD中先查询已经运行的服务器对象；

3)SMD根据服务器注册的信息，通过SAD动态启动一个服务器进程。当然，如果有合适的服务器进程已经启动，就省略2～3过程，直接执行4；

4)服务器启动，准备接受客户请求。在Server进程中，执行boa->impl_is_ready()后，对象引用同时也被注册到SMD。这是SMD根据对象实现注册在SAD中的"主机号，端口号，服务名"来获得的。

5)SMD返回给客户方对象引用。

6)客户方通过对象引用，引发对象实现的方法。

3 系统设计实现关键技术

3.1 服务器动态扩展

通过SAD和SMD两种服务管理器，可以实现本地服务器进程的管理及网络范围内的各个主机上的应用进程管理。SMD能够方便地实现负载均衡，它可以很容易地根据某种算法选择合适的主机来运行服务器，当客户请求增多时，通过动态扩展服务器就能够很好地缓解服务器系统的压力。

3.1.1 *应用构件技术* SMD/SAD服务对象的实现、相应骨架及具体CORBA产品核心开发库联合编译而成的构件形式的应用模块，可以通过系统配置的手段加入已存在的服务端系统，已存在系统不要求重新编译。通过这种应用构件技术，可以在不影响系统运行的情况下，扩展进程管理服务，做到应用系统的平滑升级。而这一切对客户都是透明的，不会增加客户的编程开销。

3.1.2 *SMD/SAD集群* SMD和SAD组成的进程管理服务是一个开放性的结构，可以组成一个级联的大型系统。客户方首先根据系统配置文件定位一个ORB域上的SMD，由SMD定位一个SAD。如果找不到合适的服务器满足客户要求，SAD就将客户请求分派给它所管理的下级SMD，在下层SMD/SAD系统中定位合适的服务器进程。此过程可以反复递归多次，以便由多级SMD组成一个大型CORBA服务器集群系统。

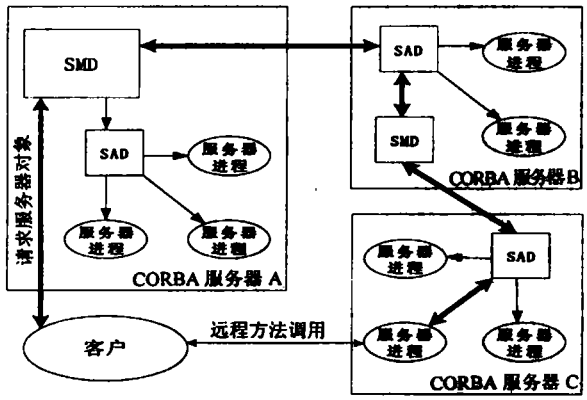


图2 SMD/SAD集群进程管理

由于有了层次型的服务器/对象定位机制，CORBA应用系统的动态扩展就变得比较容易。当需要扩充已经存在一个CORBA/ ORB域时，只需在其上层增加新的SMD管理该ORB域即可。两个不同的CORBA/ ORB域也可以通过这种级联的方式，组成逻辑上相同的ORB域，满足系统可扩充性的需要。

3.2 SMD定位SAD的策略

3.2.1 *面向集群改进的LARD策略* SMD可以通过会话亲和的策略定位SAD并激活服务器进程，这种方式虽然简单高效，但是容易引起服务器的负载不均，导致有些服务器因为负载过重而崩溃。我们采用CBRD(Content-Based Request Distribution，基于内容的请求分派)策略：前端SMD根据客户请求的内容或服务对象以及SAD负载情况，具体选择由哪一个SAD来负责处理客户请求，以达到提高请求执行效率和负载均衡的目的。

LARD(Locality-aware request distribution)[7]是一种简单而实用的CBRD策略。面向集群改进的LARD算法可以在兼顾负载均衡的同时通过提高后台服务器缓存中SAD的命中率来达到提高请求执行效率的目的，而缓存命中率的提高则是通过进一步动态划分SAD的工作设置情况来获得的。图3描述了基于LARD策略的SMD/SAD进程管理的原理。

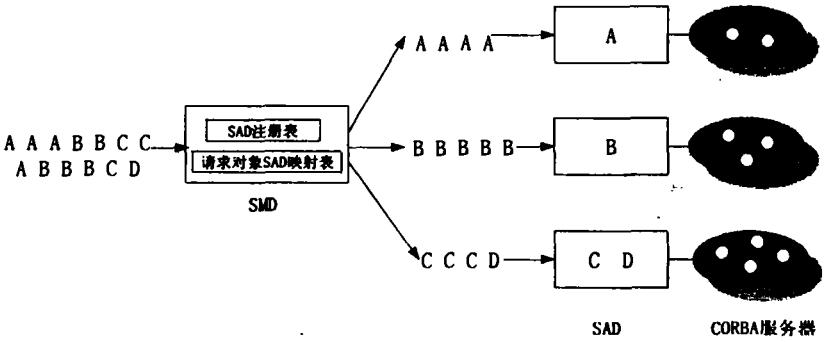


图3 面向集群系统的 LARD 策略

相比轮转算法中对同一个对象的请求分派到不同服务器上的策略来说，LARD 算法大大提高了客户请求在服务器缓存中找到服务对象的几率。当然也存在一种极端的情况，当所有请求都是同一对象时，会破坏整个系统负载均衡。这就需要改进 LARD 算法，在提高请求缓存命中率的同时尽量满足服务器系统负载均衡的要求。

在 SMD 中建立一个请求对象到负责处理该请求的 SAD 的映射，通过查询该映射表中"请求对象"以及 SAD 注册表中的"负载计数"，可以将客户请求分派到处理该请求内容的负载最小的 SAD 上。如果 SMD 检测到负载不均的情况出现(即某种请求的数量过大)SMD 就会新选择一个轻负载的 SAD 增加对该内容请求的处理。另一方面当 SMD 检查到有多个 SAD 处理某种内容的请求，并且在一段时间(K)内没有新的请求到达，SMD 就会断开请求内容和处理该请求内容之间的映射，确保了服务器的可利用率。面向集群改进的、可动态划分 SAD 设置环境的 LARD 算法如下：

```
While (true)
    SMD 取得下一个请求 r;
    If SADset[r. 请求对象] = Φ then
        n ← 最轻负载的 SAD;
        添加 n 到 SADset[r. 请求对象]中;
    else
        n ← SADset[r. 请求对象]中最轻负载的 SAD;
        m ← SADset[r. 请求对象]中最重负载的 SAD;
        if (n. 负载计数>Th && SADset[r. 请求对象]中所有 SAD
              的负载计数<Tl)
            || n. 负载计数>2Th then
        p ← 最轻负载的 SAD;
        添加 n 到 SADset[r. 请求对象]中;
        n ← p;
    if | SADset[r. 请求对象]| > 1 &&
              (time()-SADset[r. 请求对象]. lastMod)> K then
            从 SADset[r. 请求对象]中移除 m;
    将请求 r 分派到 n;
    if 本次过程中 SADset[r. 请求对象]发生改变 then
        SADset[r. 请求对象]. lastMod ← time();
```

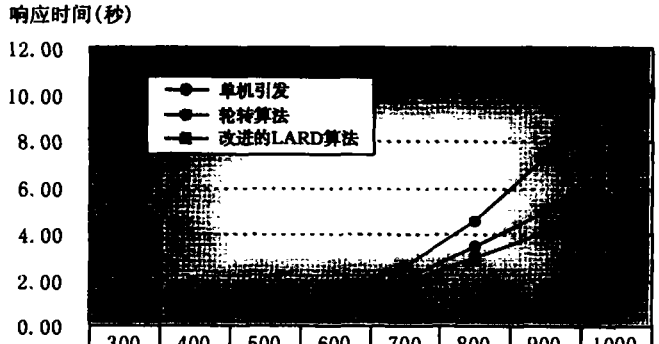
其中，Th 定义为 SAD 负载计数的上限，超过该值就有可能导致处理请求的延迟；Tl 定义为 SAD 负载计数的下限，低于该值说明该 SAD 还有能力处理更多的请求。

3.2.2 *按策略启动服务器进程* 服务器的策略是在进程一级的，和对象策略不同的是它处于 ORB 之上，由 SAD 负责管理。进程管理服务支持以下策略：Shared，Unshared，Per-method。Shared 指不同客户可以共享同一进程；Unshared 指不同客户不能共享同一进程，即每个客户有不同的服务进程；Per-method 指客户的每次方法引发，都由不同的进程来服务。客户请求通过 SMD 定位到 SAD 后，SAD 根据不同的服务器策略来启动服务器进程，管理服务的灵活性和可动态配置性。

4 实验结果及分析

本文所采用的实验平台是我们基于 CORBA 规范2.x 所开发的 UestBroker 分布计算环境。它支持多线程的 ORB，为测试集群系统 SMD/SAD 管理服务的性能和效率提供了一个较好的测试环境。实验中，我们采用 Per-method 进程策略，分别测试了单 CORBA 服务器、3台基于轮转算法的 CORBA 服务器集群以及3台基于改进 LARD 算法的 CORBA 服务器集群上客户通过 SMD/SAD 引发多个不同对象请求的响应时间，实验结果如图4所示。

通过比较我们可以得出集群服务器处理客户请求的能力大大高于单服务器。同时，在集群服务器的情况下相比轮转算法而言，基于改进的 LARD 算法进程管理服务的性能提高得



	300	400	500	600	700	800	900	1000
单机引发	0.1935	0.3974	0.701	1.447	2.571	4.563	7.325	11.348
轮转算法	0.1931	0.3131	0.546	1.127	2.012	3.478	4.931	6.531
改进的LARD算法	0.1933	0.3131	0.573	1.102	1.897	2.971	4.101	5.252

图4 实验结果

(下转第75页)

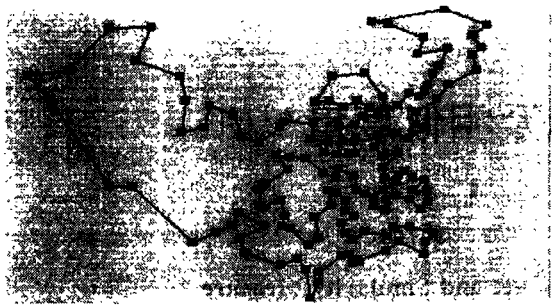


图2 PGA 算法求得的 CHN144最优路径图(最短路径值=30353.86099652386)

结论 实验说明使用文中提出的组合算子作为主要遗传操作算子,和采用具有进化周期性特点的算法框架的 PGA 算法具有较高的求解质量,连续25次实验,每次算得的结果都能达到已知最优解,所用的时间也较短,100个以内的城市平均也就几秒钟,200个左右的城市平均2分钟左右,这个时间远少于基于杂交操作的遗传算法所用的时间。值得指出的是,对于中国144个城市的 TSP 问题,PGA 算法基于浮点运算求得的最优解明显好于已知基于浮点运算求得的最优解(文[6,7]给出的解)。

从实验看到,求解52个到226个城市的 TSP 问题所用的群体规模都是16个个体,而其他的遗传算法使用的群体规模都在50个个体以上,这说明 PGA 算法能有效地维持群体的多样性,从而不会发生算法的过早收敛,也说明文中提出的组合算子具有优良的搜索性能。PGA 算法使用了很少的参数,如果除开群体规模和算法的终止条件,则只使用了一个参数及一个进化周期所进化的代数 EG。PGA 算法是所有的进化周期都固定为相同的进化代数 EG,下一步对算法的改进目标是使 EG 能自适应地调节,及一个进化周期的结束不由固定的参数来控制,一方面减少算法的参数,使算法更容易控制,一方面进一步加快算法的求解速度。

参考文献

1 Garey M, Johnson D. Computers and Intractability. W. H. Freeman, San Francisco, 1979

2 Goldberg D E, Lingle R. Alleles, loci, and the Traveling Salesman Problem. In: Proc. of an Intl. Conf. on Genetic Algorithms and Their Applications, 1985. 154～159

3 Davis L. Job Shop Scheduling with Genetic Algorithms. In: Proc. of an Intl. Conf. on Genetic Algorithms and Their Applications, 1985. 136～140

4 Smith D. Bin Packing with Adaptive Search. In: Proc. of an Intl. Conf. on Genetic Algorithms and Their Applications, 1985. 202～206

5 徐宗本,高勇. 遗传算法过早收敛现象的特征分析及其预防. 中国科学(E 辑),1996,26(4):364～375

6 Jiang Rui, Szeto K Y, Luo Yu-pin, Hu Dong-Cheng. A path-splitting scheme based distributed parallel genetic algorithm for large traveling salesman problems. In: proc conf. on Intelligent Information processing(WCC2000-IIP2000), 2000. 478～485

7 吴斌,史忠植. 一种基于蚁群算法的 TSP 问题分段求解算法. 计算机学报,2001,24(12):1328～1333

8 郭涛. 演化计算与优化:[武汉大学博士学位论文]. 1999. 50～51

(上接第22页)

更多,尤其是当请求服务器对象的种类和数量增大时,LARD 算法通过动态划分 SAD 工作环境将同种对象请求分派到相同服务器上,其性能提高更加明显。由此我们可以得出结论,使用面向集群改进 LARD 算法的 SAD/SMD 进程管理服务具有负载均衡、可动态扩展以及提高系统处理能力等特点。

结束语 在部署了应用服务器集群的系统中,引入进程管理服务后使得服务器位置透明化:客户方通过统一的入口访问系统,由系统来定位具体的服务器;进程管理可以在服务器集群间迁移,管理 ORB 域中服务器的生命周期和状态。同时,基于 LARD 算法的定位策略在保证服务器运算稳定的情况下,较大地提高了集群系统的性能。通过进程管理服务,系统可以随时了解服务器的运行信息,并根据系统的运行情况动态增删服务器,动态调整系统负载均衡。集成了该 SAD/SMD 进程管理服务的 CORBA 平台,目前在省级电力系统中得到了广泛的应用。我们下一步的工作是在现有进程管理服务框架的基础之上,深入研究分布计算环境中 CORBA 集群冗余服务容错以及 SAD/SMD 和名字服务集成的问题。

参考文献

1 Object Management Group. The Common Object Request Broker: Architecture and Specification 2.3, June 1999

2 Niderost B U, Gommans L, Kemmerling G, et al. Objectivity/Corba distributed database performance on a gigabit Sun-ultra-10 cluster. Real Time Conference, 1999. 442～445

3 Craske G, Tari Z. A property-based clustering approach for the CORBA Trading Service. Distributed Computing Systems, 1999. 517～525

4 Miller J A. CORBA-Based Run-Time Architectures for Workflow Management Systems. Journal of Database Management, Special Issue on Multidatabases, July 1996

5 Chien S-Y, Tsotras V J, et al. Efficient Management of Multiversion Documents by Object Referencing. The VLDB Journal, 2001. 291～300

6 Aron M, Sanders D. Scalable Content-Aware Request Distribution in Cluster-based Nework Servers. USENIX 2000 Annual Technical Conference, San Diego, CA, June 2000

7 Vivek S, et al. Locality-Aware Request Distribution in Cluster-based Network Servers. Architectural Support for Programming Languages and Operating Systems, 1998