**3. 在云中扩展WEB应用**

在下面几节中,我们将提出我们的扩展方案，一种新颖的架构设计通过动态扩展算法来扩展一个安装在云上虚拟机中的web应用程序。该方案是基于使用前端负载均衡器动态地将用户请求路由到后端web服务器主机上的web应用程序。web服务器的数量应根据当前活动会话的数量阈值，自动地在每个web服务器实例中扩展以维护服务对于质量的需求。

A．架构设计

我们考虑这样一个场景, 在任何时间，对一个给定的web应用程序在互联网上提供优质的网络服务。因为可能有潜在的无限数量的用户，所以访问该应用程序的请求数量是不可预知的。通常这种类型的工作负载需求应用程序短响应时间以及高度的可靠性和可用性的。因此，不管多少数量的并发用户访问系统，web应用程序应该提供没有停机时间和保障最快的响应时间。

这样的web应用程序的主要问题是不能提前计划没，特别是预测用户将访问的数量。一种解决方案是通过动态管理web应用程序的方式，根据需求让web服务器和web应用程序组件的数量增加(或减少)。一个可伸缩的架构可以有效地处理这种情况。见图2使用虚拟化的云计算环境。架构设计包括前端负载平衡器，包括许多web应用程序的虚拟机,配置子系统和具备动态扩展算法的服务监控子系统。

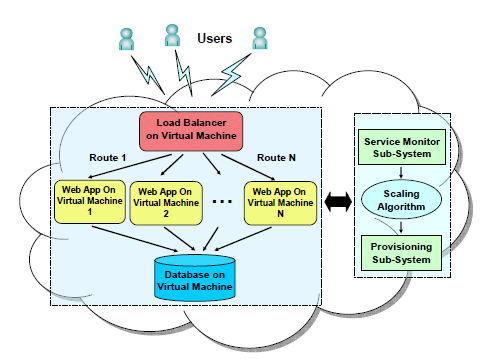


图2.云上扩展WEB应用的架构

基本上,前端负载平衡器是一个Apache HTTP负载均衡器,作为一个单一的web请求,入口点，用于将请求路由到web服务器主机上相同的web应用程序。web应用程序被部署于安装在Linux虚拟机上的Apache HTTP服务器。这些虚拟机根据需要通过配置子系统在云基础设施上配置和启动。

为了控制配置或解除的web服务器虚拟机实例行为，一种基于相关阈值或web应用程序的扩展指标的动态扩展算法被开发出来的。这里的扩展指标指的是在每个web应用程序上，我们选择活动会话或登录会话的数量。据我们对于性能和可伸缩性的测试结果，这里的扩展指标指是一个影响我们部署在Apache HTTP服务器的Linux操作系统中web应用程序性能的关键。在这项研究中,我们发现web应用程序只支持大约40000活跃并发会话,当活动会话数量超过这个阈值仍然增加时,web服务器系统将会开始失败的响应用户请求,然后最终将事件冻结起来。

B．负载均衡器与Web-App虚拟应用

使用如Apache HTTP负载均衡器的类似负载平衡器将允许传入的HTTP请求路由到web服务器主机上的web应用程序。因为Apache HTTP负载均衡器配置可以在系统运行时更新,这使得系统可以自动以及动态添加新的web服务器实例。附加的web服务器启用系统来扩展,从而对于传入的HTTP请求提供更好的响应时间。

为了简化配置额外的资源,web应用程序和其相应的web服务器预先安装在一个虚拟机设备应用镜像模板，并可在云上的镜像仓库中获得。通过系统命令使用相应的应用镜像模板,web应用程序的新实例可以十分容易的配置与创建，。

C．服务监视与配置子系统

服务监控子系统负责收集单个web应用程序的比例指标,然后计算他们的平均偏移。在我们目前的设计中,单个web应用程序上的活动会话数量将作为比例指标来使用。监控代理将安装在每个web应用程序上来跟踪活动会话的数量，并定期转发数量给服务监控子系统。

基于扩展的平均偏移,一种动态扩展算法将在下一节中给出,用于触发扩展事件给配置子系统。根据更新的统计显示, 规模向上或向下的行为可能会启动。规模上升或下降意味着一个事件，将触发指示配置子系统启动或关闭运行在云中的虚拟机服务器实例。在扩展web应用程序的情况下,新启动的虚拟机实例将会运行web应用程序。一旦web应用程序实例启动准备就绪,前端负载平衡器将更新配置文件更新，并刷新将它们分配为活动的服务。

供应子系统基本上是通过IBM Tivoli Provisioning Manager(TPM)软件产品[12]构造。这种TPM软件组件通常用于自动完成人工任务，在服务器集群上启动和配置服务器、操作系统、中间件、应用程序、存储和网络设备。为了启用和管理在云中的网络服务器和虚拟机,多种的TPM自动化工作流程已经被开发出来以应对不同的操作系统和虚拟化技术。这些工作流程作为TPM的插件工具以web服务api的方式来被云中的其他组件使用。

D．基于镜像的配置

基于镜像的配置是一种部署和激活机制，它通过克隆一个“真实”的虚拟镜像来创建新的虚拟机实例。克隆虚拟映像的挑战之一是处理不同操作系统、网络和应用程序中特殊的定制。从一个“真实””镜像模板[13]自动化配置新虚拟机可以通过添加自动化功能到模板镜像,并结合外部的自动化脚本来控制部署。

通过我们的工作, 在Red Hat Enterprise Linux系统[5][14]中，基于镜像，我们执行自动化的基于Xen hypervisor技术的Linux虚拟应用的配置。我们利用一个简化的配置过程,它是一种利用Linux桌面镜像挂载磁盘到克隆的Xen镜像，在启动前修正主机名与网络。整个过程可以归纳为两个简单的阶段：

第一阶段是复制“真实”的镜像应用模板到目标Linux主机系统上,主机系统挂载镜像文件,在挂载的文件系统中执行修复主机名和网络。修复涉及的第一步是修改/etc/hosts文件，/etc/sysconfig/network文件和/etc/sysconfig/networkscripts/ifcfg-ethx文件为正确的新主机名和网络设置实体。第二步修复是复制主机系统的RSA公钥到虚拟机的/root/.ssh/authorized\_keys文件中。前一步将允许新的虚拟机启动新的主机名和静态IP地址,从而允许外部网络访问。后者步骤将允许主机系统远程执行虚拟机的secure shell脚本,而不需要提供密码。第二阶段的配置过程涉及到启动虚拟机,和VM上secure shell脚本的远程执行没，密码更改以及应用的特殊配置。

E．扩展算法

正如前文所言, 扩展算法是通过服务监控子系统实现的,用于在服务监控子系统中控制和触发基于扩展统计指标地扩大或减少虚拟机实例的数量。我们的动态扩展算法,在图3给出了伪程序,基于人工智能在每个扩展指标云中的虚拟机实例。为了演示, 在我们的实现中我们选择每个实例的web应用程序的活动会话数量作为扩展指标。

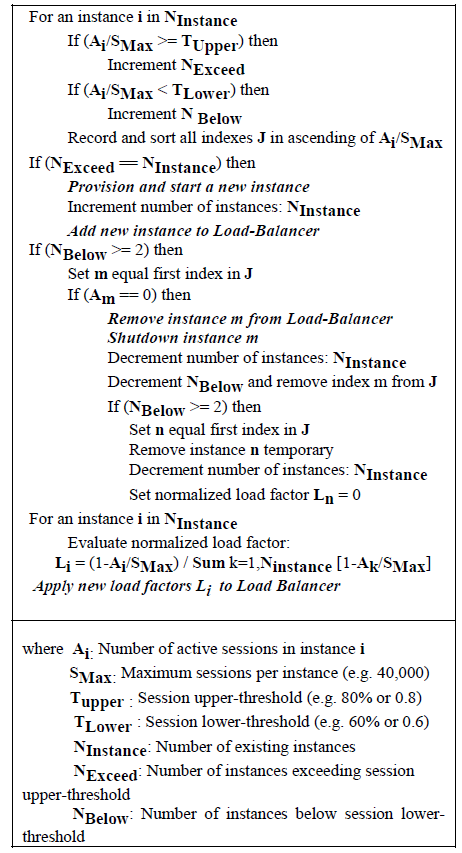


图3.对于云中虚拟机实例的动态扩展算法

算法首先判断当前的web应用程序实例与活跃的会话数量是否高于或低于给定的阈值。如果所有实例有活跃的会议超过给定的阈值,将启动一个新的web应用程序实例,然后将其添加到前端负载平衡器中。如果有与活跃的会话实例低于给定的阈值低并且至少有一个实例没有活跃的会话,这个空闲实例将从负载平衡器删除并关闭系统。在所有情况下,所有活动实例都将重新计算负载因子,然后应用到负载平衡器中来重新分配请求处理到每个实例中。

**4. 相关工作**

在过去的十年中,分布式系统的目标都集中在接口与实现的解构,代理模型,基于订阅的计算、面向服务架构(SOA)[15]和社会协作上。最近,依托互联网分布式、多租户[16]应用来连接到内部业务应用程序,被称为软件即服务(SaaS)[6]，越来越受欢迎。企业利用互联网中心托管数据，由第三方供应商的虚拟云计算环境来缓解硬件、软件、维护、可用性、可靠性和可伸缩性的担忧。

特别地,可伸缩性是关于云计算讨论中的主要话题之一。可伸缩性是一种应用程序通过在一个服务器池上扩展以满足复制和分发请求的需求。web应用程序的动态可伸缩性在目前的虚拟云计算概述工作中还没有得到太多讨论。大部分之前的网络可伸缩性工作[19]已经被报告通过静态负载均衡和服务器集群的解决方案来实现。

**5. 结论**

总而言之,我们提出了一个扩展方案来解决web应用程序在虚拟云计算环境下的动态可伸缩性。系统由前端负载均衡器来均衡分配用户请求路由到相应的部署在云中虚拟机实例的web服务器上的web应用程序。也介绍了一种动态扩展算法基于活动会话的数量自动配置虚拟机资源。我们的工作已经证明了云的引人注目好处是能够处理突然激增的负载需求,并按需以一个更好的和更便宜的方法向用户提供IT资源。自动配置、动态分配以及云的快速扩展功能的基本要素是提供更高的资源利用率,从而减少基础设施和管理成本。

**引用**

[1] G. Gruman, "What cloud computing really means", InfoWorld, Jan. 2009.

[2] R. Buyya, Y. S. Chee, and V. Srikumar, “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities”, Department of Computer Science and Software Engineering, University of Melbourne, Australia, July 2008, pp. 9.

[3] D. Chappell, “A Short Introduction to Cloud Platforms”, David Chappell & Associates, August 2008.

[4] VMware ESX Server, VMware Inc., <http://www.vmware.com/products/vi/esx/>

[5] Xen Hypervisor, <http://www.xen.org/>

[6] E. Knorr, “Software as a service: The next big thing”, InfoWorld, March 2006.

[7] Virtualization Technology, <http://www.kernelthread.com/publications/virtualization/>

[8] VMware Inc., "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", VMware, 2007, <http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf>.

[9] Kernel-based Virtual Machine (KVM), http://www.linuxkvm. org/page/Main\_Page

[10] Force.com, <http://www.salesforce.com/platform/>

[11] Microsoft Virtualization, <http://www.microsoft.com/virtualization/>

[12] IBM Tivoli Provisioning Manager Products, <http://www.ibm.com/software/tivoli/products/prov-mgr/>

[13] L. He, S. Smith, R. Willenborg and Q. Wang, “Automating deployment and activation of virtual images”, IBM developerWorks, Aug. 2007. http://www.ibm.com/developerworks/websphere/techjournal/0708\_he /0708\_he.html

[14] Red Hat Enterprise Linux, <http://www.redhat.com/rhel/>

[15] T. Erl, “Service-oriented Architecture: Concepts, Technology, and Design”, Upper Saddle River, Prentice Hall, 2005.

[16] F. Chong, G. Carraro, and R. Wolter, “Multi-Tenant Data Architecture”, Microsoft Corporation, 2006.

[17] V. Ungureanu, B. Melamed, and M.Katehakis, “Effective Load Balancing for Cluster-Based Servers Employing Job Preemption”, Performance Evaluation, 65(8), July 2008, pp. 606-622.

[18] L. Aversa and A. Bestavros. “Load Balancing a Cluster of Web Servers using Distributed Packet Rewriting”, Proceedings of the 19th IEEE International Performance, Computing, and Communication Conference, Phoenix, AZ, Feb. 2000, pp. 24-29.

[19] V. Cardellini, M. Colajanni, P. S. Yu, “Dynamic Load Balancing on Web-Server Systems”, IEEE Internet Computing, Vol. 33, May-June 1999, pp. 28 -39.