Journal of Computer Applications
计算机应用,2013,33(12):3335-3338,3349
ISSN 1001-9081
CODEN JYIIDU
2013-12-01
http://www.joca.cn

文章编号:1001-9081(2013)12-3335-04
doi:10.11772/j.issn.1001-9081.2013.12.3335

基于 OpenStack 的清华云平台构建与调度方案设计

赵少卡[1,2*],李立耀[1],凌　晓[2],徐　聪[2],杨家海[2]
(1.福建师范大学福清分校 数学与计算机科学系,福建 福清 350300;　2.清华大学 网络科学与网络空间研究院,北京 100084)
(*通信作者电子邮箱 zska@cernet.edu.cn)

摘　要:从一般云计算的体系结构与清华大学的实际需求出发,利用先进的 OpenStack 平台,采用分层设计的方法设计实现一个可对云资源进行综合管理的清华云平台。分析了该系统的优势和应具备的主要模块功能,重点研究系统中的资源调度关键技术,提出了一种基于任务调度和负载均衡的策略,并通过对调度方案的实验与分析,验证了该调度策略在保证服务性能和执行效率的基础上能够均衡服务器的资源负载,使云平台处于相对稳定的状态。

关键词:云计算;云平台;OpenStack;体系结构;资源调度
中图分类号:TP393.027;TP311　　**文献标志码**:A

Architecture and scheduling scheme design of TsinghuaCloud based on OpenStack

ZHAO Shaoka[1,2*], LI Liyao[1], LING Xiao[2], XU Cong[2], YANG Jiahai[2]
(1. *Mathematics and Computer Science Department, Fuqing Branch of Fujian Normal University, Fuqing Fujian* 350300, *China*;
2. *Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing* 100084, *China*)

Abstract: Based on cloud computing's architecture and the actual demands of Tsinghua University, followed by utilizing the advanced OpenStack platform, adopting hierarchical design method, the TsinghuaCloud platform that could be used to perform integrated management on cloud resources was designed and implemented. The advantages and main required module functions of this system were analyzed. Focusing on the resource scheduling, a strategy based on task scheduling and load balancing was proposed. The experiment and analysis of the scheduling plan verify that the scheduling strategy can balance server's resource load on the basis of ensuring its service performance and execution efficiency, so as to make the cloud platform relatively stable.

Key words: cloud computing; cloud platform; OpenStack; architecture; resource scheduling

0　引言

工业界以 Google、Amazon、IBM、Microsoft 等为代表的公有云计算平台发展迅猛,在网站数据处理、在线服务等领域作用显著。学术界对云计算也开展了许多研究,但遇到了诸多困境,如难以取得资源池上的测量参数指标等,限制了对云计算系统的全面了解,因此建立一个小型的云计算基础平台就显得尤为重要。为此,本文在 OpenStack 开源软件包的基础上,根据清华大学教学科研的实际需要,利用虚拟化技术构建了清华云(TsinghuaCloud)服务平台,该平台对云端资源池的分配合理高效。

1　一般云计算 IaaS 体系架构

云计算的体系结构的特点包括:设备众多,规模较大,利用了虚拟化技术,提供任意地点、各种设备的接入,并可以定制服务质量等。图 1 所示是一种面向市场应用的云计算体系结构[1]。

用户/代理可以从任意地点提交服务请求任务,服务等级协议(Service Level Agreement,SLA)资源分配器接收用户提交的请求,经过相应的处理,再提交到后端处理。

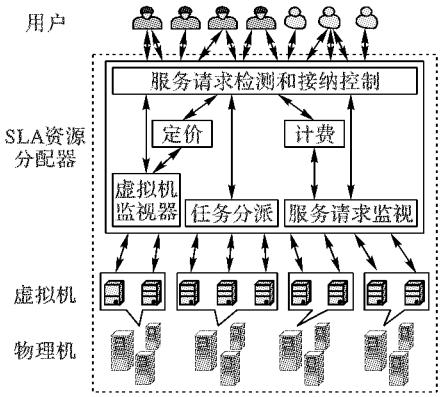

图 1　面向市场应用的 IaaS 体系架构

SLA 资源分配器的子模块主要有:服务请求检测和接纳控制模块、定价模块、计费模块、虚拟机(Virtual Machine,VM)监视器模块、任务分派模块以及服务请求监视模块等。服务请求检测和接纳控制模块主要完成当服务请求首次提交时,服务请求检测和接纳控制模块检测该请求的服务质量(Quality of Service, QoS)需求,决定是否接纳该请求,该机制确保不会出现资源过载,但是可能会因此导致部分请求因为资源短缺问题而被拒绝,它需要协同 VM 监视模块的资源可用信息和服务请求监视器模块的负载处理信息;定价模块主

收稿日期:2013-07-10;**修回日期**:2013-08-15。　**基金项目**:教育部—中国移动研究基金资助项目(MCM20123041);福建省教育厅科技项目(JB13198,JA13343,JA12352);福建师范大学福清分校科研创新基金资助项目(KY2013001)。

作者简介:赵少卡(1980-),男,福建福州人,讲师,硕士,CCF 会员,主要研究方向:云计算、软件工程;　李立耀(1970-),男,福建平潭人,副教授,硕士,主要研究方向:计算机网络;　徐聪(1986-),男,黑龙江东宁人,博士研究生,主要研究方向:计算机网络;　凌晓(1987-),男,广东湛江人,博士研究生,主要研究方向:计算机网络、软件工程;　杨家海(1966-),男,浙江云和人,教授,博士,主要研究方向:计算机网络。

要负责服务请求的计价方式选择;计费模块主要负责根据计价方式和实际用量计算用户应付的费用,同时会保存用户的资源请求信息;虚拟机监视器模块主要负责监测 VM 的可用情况和资源信息;服务请求监视模块主要负责监视跟踪已接纳服务的执行情况。

2 TsinghuaCloud 平台的选用

为了构建合理的清华云平台架构,保证云平台中各种应用的顺利实施,最根本的是需要建立一个稳定的基础架构即服务(Infrastructure as a Service,IaaS)层。IaaS 通过互联网提供给消费者的服务是对所有设施的利用,包括处理、存储、网络和其他基本的计算资源,用户能够部署和运行任意软件。用户不需要管理或控制任何云计算基础设施,就能进行操作系统的选择乃至储存空间、部署的应用,也有可能获得有限制的网络组件(如防火墙、负载均衡器等)的控制。

目前,可用于搭建 IaaS 服务的工具较多,主要有 Rackspace 和 NASA 联手推出的 OpenStack、美国加利福尼亚大学设计的 Eucalyptus 以及欧洲的 OpenNebula 等。清华云选用 OpenStack 作为云计算设计的平台工具,主要原因是由于其开源的特质,并具有良好的控制性、兼容性、可扩展性与灵活性。OpenStack 由美国国家航空航天局和 Rackspace 公司合作研发,目前已占据市场主流地位。它致力于提供规模化、灵活扩展易部署且功能丰富的全开源模式平台,协助运营商、企业、ISP/CP、科研机构等搭建并实现满足自身需求的公共云和私有云服务,有力地推动了云计算的创新发展。

OpenStack 的主要组件及其相互关系包括:OpenStack Compute(Nova),OpenStack Object Storage(Swift),OpenStack Image Service (Glance), Identity (Keystone), Dashboard (Horizon)以及 Network Connectivity(Quantum)等[3]。其中,Dashboard(Horizon)提供了一个 Web 前端到 OpenStack 其他服务的界面;Compute(Nova)组件主要负责管理虚拟机,存储和检索虚拟磁盘(Image)和 Image 上相关的元数据(Glance),并提供管理和维护系统镜像的服务;网络组件 Network (Quantum)提供构建与管理虚拟网络的功能,它将网络连接作为服务提供;块存储服务组件 Block Storage(Cinder)提供存储功能;Image(Glance)在对象存储组件(Swift)上能够完成虚拟磁盘文件的存储,该组件提供 petabytes 级别的、安全可靠的对象存储服务;所有的服务均需要利用 Keystone 进行身份验证。

3 TsinghuaCloud 系统架构

在选定好虚拟化平台软件 OpenStack 的基础上,利用操作系统 Ubuntu,构建稳定的 IaaS 层应用,完成对平台的安装部署,并基于该平台,从系统的实际需求出发,按照软件工程的思想进行模块的划分与构建,对系统的功能结构进行提炼和分解,最终设计实现一个可对云资源进行综合管理的清华云系统。

3.1 硬件组成

TsinghuaCloud 是基于开源的虚拟化管理软件 OpenStack,采用 Smarty MVC 框架,综合使用 PHP、Shell、Java、Python、HTML 等编程语言实现的一个构架于服务器、存储、网络等基础硬件资源和单机操作系统、中间件、数据库等基础软件之上的云平台综合管理系统,同时也是一个云计算后台数据中心的整体管理运营系统。

目前 TsinghuaCloud 采用标准多节点部署的方式,共部署了两个区域:第一个区域部署在学校的主楼机房,这也是清华云平台的主体;硬件资源为 4 台 DELL R720 服务器、2 台 HP 服务器、24 TB 存储(SAN RAID6)。第二个区域部署在未来信息技术大楼(FIT 楼),主要用于平台的后续开发测试;硬件资源为共 4 台 DELL 电脑(单核 4 个处理器 Intel Core i5-2400 CPU @ 3.10 GHz,80 GB 内存,千兆网卡)。清华云的控制中心及 Web 服务部署在第一个区域的其中一台 HP 服务器上,Glance 镜像服务单独部署在 R720 服务器上,网络服务(Nova-network)部署在每个计算节点上,存储服务(Nova-volume)在物理层面挂载在两台 R720 服务器上(连接 SAN 24 TB 存储),被所有设备通过网络共享,现已能对所有机器提供存储服务。主楼区域清华云平台的系统架构与组建如图 2 所示,在每台物理机创建的虚拟机可以通过内网互相通信,并通过外网和云外设备通信。

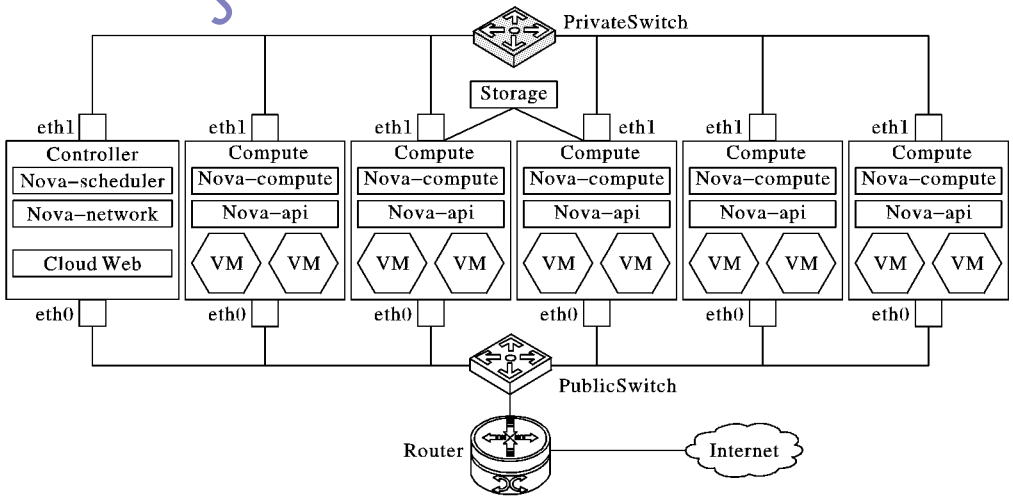

图 2 清华云平台的系统架构与组建

经监测,整个平台所运行的各类操作系统的虚拟机总数达到数百台,大部分时间内每台计算节点的物理主机的 CPU 利用率均可以达到 70% 以上,显然比传统数据中心的 CPU 利用率高出很多。

3.2 模块设计

具体地,可将清华云平台划分为前台和后台两大部分进行设计开发,各模块功能划分如图 3 所示。

前台主要分为基础设施层、应用服务层和访问控制层等

三个层面:基础设施层为云平台的正常运行提供必要的设备级支持;应用服务层主要基于 OpenStack 的 Keystone、Nova、Glance、Swift 组件进行开发,实现对租户、云主机等的相应应用;访问控制层主要用于不同身份用户的注册、登录与安全认证等。

后台主要负责对整个平台进行运行管理和控制,实现对服务器的统一管理,并将存储、网络等硬件设备抽象为资源池进行集中管理;提供用户租户管理、权限控制、余额查询、订单统计等;根据需要分配、调度和回收资源;负责接收和处理用户的资源申请,进行自动化的环境配置;进行性能监控和故障管理等。

图 3　清华云平台主要功能模块划分

4　平台的调度方案设计

4.1　调度设计的背景

在云计算平台当中,随着数据中心的虚拟机数量越来越多,对虚拟机集群的资源调度提出了新的挑战。在大规模的虚拟机集群中,虚拟机数目和虚拟机的负载会随用户和应用的需求而经常变化,静态的资源分配往往会使虚拟机产生资源浪费或资源不足的情况,而人工的动态资源调整会有明显的滞后性,因此虚拟机需要进行动态的资源调度。在虚拟机数量偏少和平均负载偏低的情况下,将虚拟机集中迁移到较少的物理机上,并将一部分物理机停机,以达到节能和提高计算/能耗比的目的;在虚拟机数量偏多和平均负载偏高的情况下,启动更多的备用物理机并进行负载平衡。同时,由于虚拟机中的应用负载会随时间变化,因此应及时响应虚拟机负载的变化,适当为高负载的虚拟机分配更充裕的资源,以适应虚拟机对资源的需求[6-7]。

由于 OpenStack 中的调度算法都过于简单,例如其中的 ChanceScheduler、SimpleScheduler、MultiScheduler 等算法,具体的实现机制较为简单且存在诸多不完善之处,不足以适应大规模的资源调度场景,对于一个相对庞大的集群,这些简单算法很难做到资源的负载均衡,无法真正在服务性能和执行效率的基础上保证服务器的资源负载均衡,因此需要进一步研究和设计清华云平台资源均衡利用的调度机制。

4.2　算法设计

清华云平台设计了一种新的调度算法 QAR(排队与资源负载均衡),其核心思想是:首先引入随机模型,使用基于优先级别的队列,减少轻量级服务请求被丢弃的概率以及请求积压的数量,提高系统的吞吐率;接着综合考虑服务器负载情况以及资源请求的积压状况,选择响应时间最短同时负载情况相对较小的目标主机进行虚拟资源的分配,使得调度策略在优化服务性能指标的同时,在服务器负载均衡方面效果更佳。

QAR 算法分为两步完成:第一步是对暂时无法处理的服务请求进行排队,选择合适的任务进行操作。当等待队列未满,即等待分配的任务总量小于其所容许的最大值时,此时单位时间内请求的丢失率接近于 0,则优先选择消耗资源较大的任务继续分配处理,以减少大消耗量任务的积压与堆积,避免造成后续等待队列满负荷时请求丢失率的升高;当等待队列中待分配的任务达到极限时,则考虑任务在队列中积压时间与其将消耗资源的乘积(通常消耗的资源与完成任务的时间成正比),以选出乘积最大的任务,较大限度地在等待队列趋于不饱和状态时充分考虑任务的分配效率。

具体地,通过式(1)计算总请求数 $L(t)$,$L_m(t)$ 为 t 时间单元内 m 类任务的请求数目,$A_m(t)$ 为新到达的请求数,$H_m(t)$ 为完成的请求数。之后判断式(2)是否成立,成立则选取待分配任务中所需消耗资源 D_{Mapu} 最大的任务进行操作;否则根据式(3)选取 W_u 值最大的任务。

$$L(t) = \sum_{m=1}^{M} L_m(t) = \sum_{m=1}^{M} [L_m(t-1) + A_m(t) - H_m(t)] \tag{1}$$

$$\sum_{m=1}^{M} \left[L_m(t) + \sum_{i=1}^{N} B_{mi}(t-1) \right] \leqslant queueLength_{\max} \tag{2}$$

$$W_u = D_{\text{Mapu}} T_u \tag{3}$$

其中:$queueLength_{\max}$ 为最大等待队列长度,$B_{mi}(t-1)$ 为 $t-1$ 时间单元内积压待分配的 m 类任务,W_u 为待分配任务的权重;D_{Mapu} 为待分配任务所需消耗的资源量,T_u 为待分配任务在等待序列中积压的时间单元数。

第二步是选择最适合的服务器进行任务的分配,优化资源配置,从而避免个别服务器不堪重负,而其他服务器资源闲置的不均衡状况,实现负载均衡:通过式(4)求出能处理任务 n 的服务器集合 Set,通过式(5)与式(6)求出集合 Set 中各服务器的平均负载 Avg_{DC},通过式(7)计算各服务器的负载方差 $\sigma_{\text{DC}}(t)$,当方差 $\sigma_{\text{DC}}(t)$ 最小时所分配的服务器为最终选定的服务器。

$$D_{nk} + \sum_{m=1}^{M} N_{mi}(t-1) D_{mk} \leqslant C_{ik} \tag{4}$$

$$P^i_{\text{DC}}(t) = \frac{1}{C_{ik}} \sum_{m=1}^{M} N_{mi}(t-1) D_{mk} + D_{nk} P_{ni}(t) \tag{5}$$

$$Avg_{\text{DC}} = \frac{1}{N} \sum_{i=1}^{N} P^i_{\text{DC}}(t) = \frac{1}{NC_{ik}} \sum_{i=1}^{N} \sum_{m=1}^{M} [N_{mi}(t-1) D_{mk} + D_{nk} P_{ni}(t)] \tag{6}$$

$$\sigma_{\text{DC}}(t) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} [P^i_{\text{DC}}(t) - Avg_{\text{DC}}(t)]^2} \tag{7}$$

其中:C_{ik} 为服务器 i 资源 k 的总量,$N_{mi}(t-1)$ 为 $t-1$ 时间单元积压在服务器 i 上的 m 类任务,M 为任务种类的上限,D_{mk} 为 m 类任务所消耗资源 k 的量,D_{nk} 为任务 n 所需资源 k 的量,$P^i_{\text{DC}}(t)$ 表示 t 时间单元服务器 i 上的负载,$P_{ni}(t)$ 为 t 时间单元需分配的任务 n 分配到服务器 i 的概率,N 为云平台中服务器的数量。具体的实现伪代码如下:

1)　初始化;
2)　while true do
3)　　读取新请求的个数 *taskNumber*;
4)　　*task*[*taskNumber*] ←按照请求时间先后和请求类型优先

```
        级进行降幂排序;
5)      for i = 0 to task. length do
6)        宿主机按剩余资源量进行降幂排序并选择前 K 台;
7)        比较这 K 台服务器中 task[i] 对应的 model[task[i]] 类
          型的队长,取队长最短的 serverId;
8)        让 task[i] 插入该 serverId 中 model[task[i]] 的队列中,
          同时队长加 1;
9)        for j = 0 to server. number do
10)         while server[j] 中所有 model 队长之和不为零 do
11)           计算 server[j] 中各 model 的权重,取权重最高的
              modelId 的队头请求;
12)           如果资源足够分配给该请求,队长减 1,同时扣除资
              源;
13)           否则将等待;
14)         end while
15)       end for
16)     end for
17)     sleep(20000);
18)   end while
```

4.3 实验与结果分析

基于 OpenStack 开源软件构建一个包含 6 台服务器的云计算 IaaS 平台,具体服务器配置如表 1 所示。该平台以虚拟资源的方式为用户提供资源的使用,提供的虚拟资源的模板种类如表 2 所示。为了对本文算法提出的调度策略的性能指标进行详细评估,将该算法以 Python 语言加以实现,并整合到 OpenStack 开源软件中。

表 1 测试服务器配置

服务器名称	CPU 数量	内存容量/GB	硬盘容量/GB	网卡数量
Server-43	8	8	100	2
Server-38	8	8	100	2
Tsinghuacloud-nc1	24	80	500	4
Tsinghuacloud-nc2	24	80	500	4
Dell-cloud1	24	96	1 000	4
Dell-cloud2	24	96	1 000	4

表 2 虚拟资源的模板种类

虚拟资源主机类型	CPU 数量	内存容量/GB	硬盘容量/GB	带宽
超微主机	1	0.5	8	不限
微型主机	1	1	15	不限
小型主机	1	2	30	不限
标准主机	2	4	60	不限
大型主机	4	8	120	不限
超大型主机	8	16	240	不限

利用采集到的实际用户对于不同类型服务的需求频率和需求量等数据,对实际的 IaaS 资源调度场景进行模拟:对每种主机类型模拟出特定的服务请求到达的序列,每种主机类型的请求到达频率服从强度不同的泊松分布,每种主机的申请时间服从均匀分布,之后考察此场景中不同调度算法在 IaaS 平台到达稳定状态时的性能。

在具体实验中,将表 2 中列出的服务类型按照占用资源由小到大,编号为服务 1 ~ 服务 6,对应每种服务模拟出一串服从特定强度的泊松到达请求序列。根据实际需求量的统计,用户对微型、小型主机的需求比对大型、超大型主机的需求量明显要大,因此在本实验中服务 1 ~ 服务 6 的请求到达频率依次递减。采用了不同的调度策略用于对服务请求进行调度并对虚拟资源进行分配,本实验考察的调度策略有目前使用较为广泛的 Best-Fit、Min-min 调度策略,OpenStack 开源软件提供的 SimpleSchedule、ChanceSchedule 分配策略,其他的随机调度策略以及本文清华云的调度策略。

图 4 所示是清华云平台资源调度方法与其他调度方法在任务请求积压量上的比较情况。由于采用了随机模型的优先队列排序,在选择任务时充分考虑到了阶段性调度的总体性能,从而降低了平台整体任务积压量。

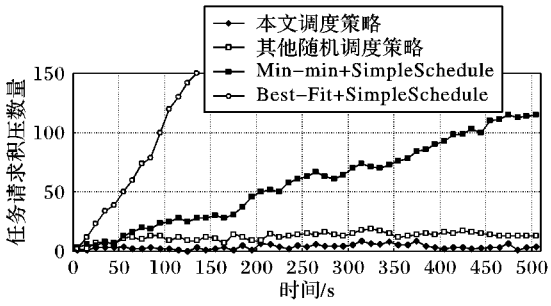

图 4 各调度算法任务请求积压量比较

图 5 ~ 7 所示是随着调度过程的进行,使用不同的调度方法在系统达到稳定状态时服务器 CPU、内存、磁盘利用率的变化,其中调整策略 A 表示 Min-min + SimpleSchedule,B 表示 Best-Fit + ChanceSchedule,C 表示其他随机调度策略,D 表示本文调度策略。为了使其他基于代数模型的调度算法也能够达到稳定状态,以便更好地比较各种调度策略在负载均衡方面的优劣,本实验中降低了服务请求到达的频率。可以看出,由于同时考虑了任务积压和资源负载,使得本方法在负载均衡方面相比其他调度方法也有明显的优势,服务器各项指标的负载均衡效果最好。

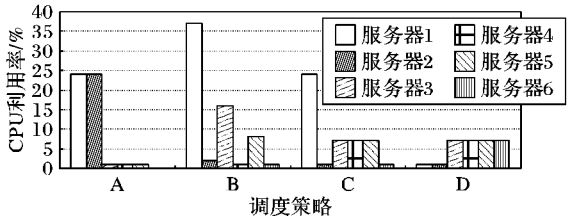

图 5 系统达到稳定状态时 CPU 负载情况

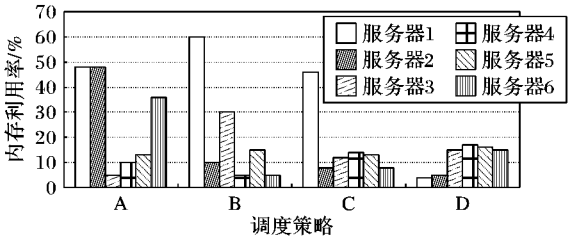

图 6 系统达到稳定状态时内存负载情况

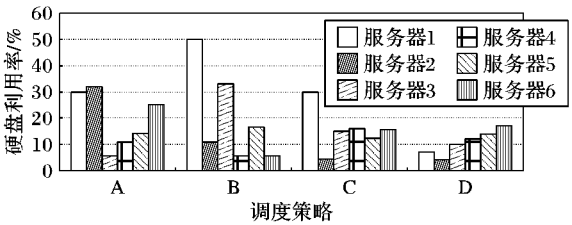

图 7 系统达到稳定状态时硬盘负载情况

5 结语

本文在分析现有云计算基本体系结构和目前最具有代表性的云计算框架的基础上,利用先进的OpenStack平台,设计

(下转第 3349 页)

从表 2 和表 3 可知,本文 Hadoop 系统的查准率和查全率略高于传统 Hadoop 图像检索系统以及 B/S 单节点图像检索系统,优势不十分明显。然而对于大规模的医学图像检索系统,系统性能优劣主要通过图像检索效率来衡量,而从图 9 可知,本文的 Hadoop 分布式系统有效降低了医学图像检索时间,提高了医学图像检索效率,较好地解决了海量医学图像检索效率低的难题,得到了比较令人满意的检索结果。

4　结语

CBMIR 的医学图像检索是一个数据密集型计算过程,传统 B/S 单节点检索系统存在效率低、可靠性差等缺陷,为此,提出一种 Hadoop 的医学图像检索系统。仿真测试结果表明,Hadoop 的医学图像检索系统提高了图像存储和检索效率,获得较优的检索结果,可以满足医学图像检索的实时性要求,尤其当处理大规模医学图像时,具有传统 B/S 单节点不可比拟的优势。但是相对于当前 Hadoop 的医学图像检索系统,优势不太明显,因此,未来的工作重点是提高 Map 任务与 Reduce 任务之间数据传输速度,减少更多由于传输信息所产生的时间消耗,进一步提高现有图像检索系统的执行效率。

参考文献:

[1] 宋真,颜永丰. 基于兴趣点综合特征的图像检索[J]. 计算机应用,2012, 32(10):2840 - 2842.

[2] 张泉,邰晓英. 基于 Bayesian 的相关反馈在医学图像检索中的应用[J]. 计算机工程, 2008, 44(17):158 - 161.

[3] 余胜,谢莉,成运. 基于颜色和基元特征的图像检索[J]. 计算机应用,2013, 33(6):1674 - 1708.

[4] CHANG F, DEAN J, GHEMAWAT S, *et al*. Bigtable: a distributed storage system for structured data [C]// OSDI 2006: Proceedings of the 7th Symposium on Operating Systems Design and Implementat. Berkeley: USENIX Association, 2006, 7: 276 - 290.

[5] KEKRE H B, THEPADE S D, SANAS S. Improving performance of multileveled BTC based CBIR using sundry color spaces [J]. International Journal of Image Processing, 2010, 4(6):620 - 630.

[6] 利业鞑,林伟伟. 一种 Hadoop 数据复制优化方法[J]. 计算机工程与应用,2012, 48(21):58 - 61.

[7] 王贤伟,戴青云,姜文超,等. 基于 MapReduce 的外观设计专利图像检索方法[J]. 小型微型计算机系统, 2012, 33(3):626 - 232.

[8] GHEMAWAT S, GOBIOFF H, LEUNG S-T. The Google File System [C]// SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles. New York: ACM, 2003: 29 - 43.

[9] 梁秋实,吴一雷,封磊. 基于 MapReduce 的微博用户搜索排名算法[J]. 计算机应用,2012, 32(11):2989 - 2993.

[10] DEAN J, GHEMAWAT S. MapReduce: a flexible data processing tool [J]. Communications of the ACM, 2010, 53(1):72 - 77.

[11] ATTEBURY G, BARANOVSKI A, BLOOM K, *et al*. Hadoop distributed file system for the grid [C]// Proceedings of the 2009 IEEE Nuclear Science Symposium Conference Record. Piscataway: IEEE, 2009: 1056 - 1061.

[12] JEFFREY D, SANJAY G. MapReduce: simplified data processing on large clusters [C]// OSDI 2004: Proceedings of the 6th Symposium on Operating Systems Design and Implementat. Berkeley: USENIX Association, 2004:107 - 113.

[13] 练秋生,李芹,孔令富. 融合圆对称轮廓波统计特征和 LBP 的纹理图像检索[J]. 计算机学报,2007,30(12): 2198 - 2204.

[14] 王中晔,杨晓慧,牛宏娟. Brushlet 域复特征纹理图像检索算法[J]. 计算机仿真, 2011, 28(5): 263 - 266, 282.

[15] ZHANG J, LIU X L, LUO J W, *et al*. DIRS: Distributed image retrieval system based on MapReduce [C]// ICPCA 2010: Proceedings of the 5th International Conference on Pervasive Computing and Applications. Piscataway: IEEE, 2010:93 - 98.

(上接第 3338 页)

出清华云的体系结构,并提出了一种基于任务调度和负载均衡的策略,充分考虑任务的特性,在云平台负载较轻时优化资源调度算法的吞吐率,在云平台资源负载较重时减小任务的积压量和请求的丢失率;在服务器选择策略上综合考虑了服务器的资源负载情况以及任务积压情况,在保证服务性能和执行效率的基础上尽量均衡了服务器的资源负载,使云平台处于相对稳定的状态,从而较好地解决了现有许多调度策略存在的问题。但如何根据虚拟机实际的应用服务性能负载情况,引入动态迁移场景建立长期负载模型,在不影响租户正常使用的前提下,使设计出来的资源调度算法在精确度和速度之间寻找一个平衡点仍需要作进一步的探讨和深入的研究。

参考文献:

[1] BUYYA R, YEO C S, VENUGOPAL S. Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities [C] // HPCC 2008: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications. Washington, DC: IEEE Computer Society, 2008: 5 - 13.

[2] MELL P, GRANCE T. The NIST definition of cloud computing (draft) [J]. NIST Special Publication, 2011, 800(145): 7.

[3] Solinea. OpenStack grizzly architecture [EB/OL]. [2013 - 06 - 15]. http://www.solinea.com/2013/06/15/openstack-grizzly-architecture-revisited.

[4] 怀进鹏,李沁,胡春明. 基于虚拟机的虚拟计算环境研究与设计[J]. 软件学报,2007,18(8):2016 - 2026.

[5] KHAN R H, YLITALO J, AHMED A S. OpenID authentication as a service in OpenStack [C] // IAS 2011: Proceedings of the 2011 7th International Conference on Information Assurance and Security. Piscataway: IEEE, 2011: 372 - 377.

[6] 盛宪锋,及俊川,周小军,等. 基于虚拟化技术的私有云 APCS 平台设计[J]. 计算机工程,2011,38(8):200 - 212.

[7] ZHAO W M, WANG Z L, LUO Y W. Dynamic memory balancing for virtual machines [J]. ACM SIGOPS Operating Systems Review, 2009, 43(3): 37 - 47.

[8] MAGULURI S T, SRIKANT R, YING L. Stochastic models of load balancing and scheduling in cloud computing clusters [C]// INFOCOM 2012: Proceedings of the 2012 IEEE International Conference on Computer Communications. Piscataway: IEEE, 2012: 702 - 710.

[9] SONG Y, WANG H, LI Y Q, *et al*. Multi-tiered on-demand resource scheduling for VM-based data center [C]// CCGRID 2009: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. Washington, DC: IEEE Computer Society, 2009: 148 - 155.

[10] SARIPALLI P, WALTERS B. QUIRC: a quantitative impact and risk assessment framework for cloud security [C]// CLOUD 2010: Proceedings of the 2010 3rd International Conference on Cloud Computing. Piscataway: IEEE, 2010: 280 - 288.