Journal of Computer Applications
计算机应用,2013,33(11):3155-3159
ISSN 1001-9081
CODEN JYIIDU
2013-11-01
http://www.joca.cn

文章编号:1001-9081(2013)11-3155-05
doi:10.11772/j.issn.1001-9081.2013.11.3155

基于分层与容错机制的云计算负载均衡策略

陈　波*,张曦煌
(江南大学 物联网工程学院,江苏 无锡 214122)
(*通信作者电子邮箱 chenbo35120@gmail.com)

摘　要:针对混合动态负载均衡算法应用在云计算中,出现的站点信息交换过于频繁导致处理效率低下以及缺乏容错机制等问题,提出了基于分层与容错机制的负载均衡算法。算法融合集中式和分布式的优点,通过组织邻站点,使站点信息交换控制在邻站点范围之内,在任务调度时携带站点实时负载信息以解决频繁广播负载消息导致网络繁忙与服务器效率低下的问题。算法实现云系统负载均衡,减小请求响应时间,引入容错备份机制,以增强系统鲁棒性。实验结果表明,基于分层与容错机制的云计算负载均衡策略在任务分配时间、任务响应时间方面比传统算法提高20%以上,且在稳定性方面所提算法优于传统算法。

关键词:云计算;负载均衡;分层算法;容错机制;资源利用率
中图分类号:TP393.02　　**文献标志码**:A

Load balancing strategy of cloud computing based on multi-layer and fault-tolerant mechanism

CHEN Bo*, ZHANG Xihuang
(*School of Internet of Things Engineering, Jiangnan University, Wuxi Jiangsu 214122, China*)

Abstract: When hybrid dynamic load balancing algorithm is applied to cloud computing, some problems will occur, such as that frequent exchange of sites information leads to low processing efficiency and algorithm is lack of fault-tolerant mechanism. Hence this paper proposed a load balancing algorithm based on multi-layer and fault- tolerant mechanism. The algorithm mixed the advantages of centralized and distributed methods. By organizing neighbor sites, the sites information exchange was controlled within a range of neighbor sites. When site scheduled tasks, it appended the load information of itself and its neighbors to the job transfer request. The method resolved network business and low efficiency of servers which was caused by broadcasting load information frequently. The algorithm achieved load balancing in cloud computing and minimum response time. The method introduced fault-tolerant mechanism to enhance the scalability of cloud system. The experimental results show that the load balancing strategy of cloud computing based on multi-layer and fault-tolerant mechanism is superior to traditional algorithms more than 20% in task distribution time and response time. Besides, it surpasses the traditional one in the stability of the algorithm.

Key words: cloud computing; load balancing; multi-layer algorithm; fault-tolerant mechanism; resource utilization rate

0　引言

云计算(Cloud Computing)是一种通过Internet以服务的方式提供动态、可伸缩的虚拟化资源的计算模式。云计算将计算及存储能力分配在由大量计算机构成的资源池上,运用集群服务器模式以虚拟化技术向外提供服务。负载均衡是云计算服务的一个重要因素,它关系到云计算环境下处理海量数据的能力和资源合理分配提高资源利用率的问题。云计算环境下负载均衡和以往集群服务器下相比发生了新的变化,使基于软件的负载均衡变得尤为重要,大大提高了硬件支撑能力,将大规模运算分配到多台云节点上并行处理,分发所有可用资源,提高数据的访问速度,确保整个系统的可用性和可靠性。目前国内外研究人员提出了很多云计算负载均衡算法,研究方向主要有集中式策略和分布式策略。集中式主要是通过系统中心节点检测各服务器负载并进行任务调度,这种方法容易维护,但是中心节点会遇到瓶颈问题。文献[1]提出的Stochastic Hill Climbing是一种集中式算法,中心节点询问其他节点负载情况并生成状态表。文献[2-4]提出的Inspiration from the Honeybee是模仿蜜蜂觅食以达到优化负载的算法,服务器间的相互通信以及负载情况等通过"广告板"来转达。分布式策略没有中心调度节点,每个节点都可以定位其他负载较轻的节点,比较适合大规模集群服务器,而且系统扩展性好,不会遇到中心节点瓶颈问题。文献[5]提出随机游走算法,节点的负载通过一个图的连通度来表示节点入度(in-degree)映射空闲资源。当节点处理一个任务时减小其入度,当系统达到稳定状态时实现负载均衡。文献[6]提出了一种基于反馈机制的动态负载平衡算法,算法在循环计算中根据反馈的负载指标分配计算任务,动态适应负载变化,但该算法在负载变化频繁时效果不佳,且不进行任务迁移。文献[7]提出一种混合动态负载均衡(Hybrid Dynamic Load Balancing, Hybrid DLB)策略,该策略对云中物理服务器进行分簇,组成站点,服务器负载信息通过站内广播收集,并通过算法选出簇头,由簇头节点评价站点负载值。站点间通过Master Node交换站点负载情况。随着云系统任务逐渐增多,Master Node交换站点易出现信息交换过于频繁导致处理效率低下以及缺乏容错机制导致系统鲁棒性差。

收稿日期:2013-05-22;**修回日期**:2013-07-18。

作者简介:陈波(1987-),男,浙江慈溪人,硕士研究生,主要研究方向:云计算、计算机网络;　张曦煌(1962-),男,江苏无锡人,教授,博士,主要研究方向:嵌入式系统、计算机网络。

针对文献[7]中提出的混合动态负载均衡策略的不足之处,分析和研究了已有文献,提出了基于分层与容错机制的云计算负载均衡(multi-layer and Fault-tolerant, Layer-Ftolent)算法,对 Hybrid DLB 算法进行改进:1)云服务器组成站点后选择一个站点作为调度服务器,对本站点进行负载管理与任务分配,增加站点选择模块维护邻站点信息,实现站点间分布式负载均衡;2)站点之间通过带宽计算形成有效邻站点,邻站点负载状态通过互相之间交换负载信息来感知,为避免因全网广播引起的网络繁忙,交换信息不独立广播,只在原站点发送任务调度请求时携带站点负载信息至目标站点,且当有新站点加入时,只在站点负载达到阈值时才会更新负载状态表;3)利用云计算中空闲资源对任务进行备份,避免因服务器错误导致任务执行失败,实现系统容错机制。最后通过仿真实验表明本文算法能较好地实现云系统负载均衡,减少请求响应时间,提高资源利用率,增强系统鲁棒性。

1 云计算负载均衡模型

1.1 站点结构

云计算系统由分布在不同区域的物理服务器(P_1,P_2,…,P_n)组成,每个服务器均接入网络。将 n 台服务器分成 m 组(S_1,S_2,…,S_m)组成站点,站点中服务器数量满足条件如下:

$$\begin{cases} 1 < m < \dfrac{n}{2} \\ |S_i| - |S_j| \leqslant |S|_{\min}, i \neq j, \text{且 } i, j \leqslant m \\ |S|_{\min} = \min(|S_1|, |S_2|, \cdots, |S_m|) \end{cases} \tag{1}$$

算法 1 服务器组站算法。伪代码如下所示:

```
Group()
{
  order(P1,P2,…,Pn);
     //式(5)计算每个服务器的负载值,按递增排列,组成列表
  randomGroup();                    //将 n 个点随机分成 m 组
  delete(不满足式(1));
  Si = randomselect();              //随机选择一种分组情况
  while(Si! = NULL)
  {
    select(表头和尾的服务器);        //从服务器列表中选择
    Si = Si - 1;
  }
}
```

上述算法中每个分组从服务器列表的头和尾交替选取服务器,可以避免初始化站点时出现负载过重或过轻的情况。

1.2 负载评价模型

云计算提供的不仅仅是硬件资源,还包含各种服务,任务数量非常庞大且复杂,在实际应用环境中,影响负载的因素有节点 CPU 处理能力 cpu_cap、CPU 利用率 cpu_uti、内存利用率 mem_uti、任务队列长度 $task\_len$、任务规模 $task\_sca$ 等,把这些负载指标以数学的方法设计成量化评价模型,尽可能地减小与实际负载之间的误差,相对全面、准确地衡量云计算中节点的负载情况。

单个服务器 CPU 处理能力计算表达式如式(2):

$$cpu\_cap_i = pe\_num_i \times pe\_mips_i \tag{2}$$

其中:pe_num_i 表示服务器处理器个数,pe_mips_i 表示每个处理器的单字长定点指令平均执行速度(Million Instructions Per Second, MIPS)。

根据式(2)得到预计任务完成时间 res_i 计算表达式[8],如式(3):

$$res_i = \left(\sum_{j=1}^{task\_len} task\_sca_j\right) / cpu\_cap_i \tag{3}$$

其中 $\sum_{j=1}^{task\_len} task\_sca_j$ 表示单个服务器任务总规模。

根据式(3)得到任务规模影响率 $task\_eff$,计算表达式如式(4):

$$task\_eff_i = 1 - \frac{res_i - Aver\_res_i}{res_i + Aver\_res_i} \tag{4}$$

其中 $Aver\_res_i$ 表示服务器 i 历史任务平均执行时间。

服务器负载均衡值 LB_i,计算表达式如式(5):

$$LB_i = w_1 \times cpu\_uti + w_2 \times mem\_uti + w_3 \times task\_eff \tag{5}$$

令

$$w_1 + w_2 + w_3 = 1 \tag{6}$$

根据式(5)计算站点平均负载(Site Average Load, SAL)均衡值及服务器负载均衡标准差 Std_LB,如式(7)、(8):

$$SAL = \left(\sum_{i=1}^{m} LB_i\right) / m \tag{7}$$

$$Std\_LB = \sqrt{\frac{\sum_{i=1}^{N} (LB_i - SAL)^2}{N}} \tag{8}$$

由式(8)可知,站点 Std_LB 越小,其负载越均衡。

1.3 站点调度器设计与选择

云服务器组成站点后需选择其中一台服务器作为本站点调度器,本文所述的调度器为运行在服务器上的软件组件。调度器管理本站点内其他服务器以及采用分布式策略调度任务至其他站点,因此基于分层与容错机制的云计算负载均衡策略结合了集中式与分布式负载均衡算法的优点,在站点内集中式管理,站点间分布式调度。

调度器结构如图 1 所示。

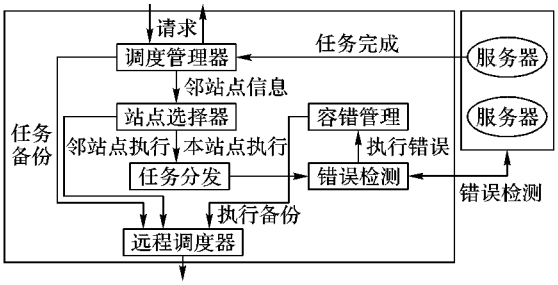

图 1 站点调度器结构

当站点接收任务请求后,调度管理器备份该任务并通过远程调度器将备份发送至指定站点。调度管理器根据任务类型从有效邻站点信息表中选择能够处理该任务的站点,发送该站点信息至站点选择器,选择器决定任务执行站点。若在邻站点执行则将任务发送至远程调度器。若在本站点执行,则通过任务发布模块决定本站点中可以处理任务的服务器。同时错误检测模块对任务执行过程进行监控,若执行出错,容错管理器通过远程调度器发送命令至有该任务备份的站点,执行备份任务。

在本文所述策略中需要设置站点调度服务器,假设某站点中有 k 台服务器(P_1,P_2,…,P_k),当其中一台服务器 P_i 不能找到调度服务器或发现调度服务器出现故障时,首先站内广播负载查询消息收集其他服务器负载值。然后将包括 P_i 在内的各服务器根据负载值大小升序排序,找到负载值处于队列中间位置的服务器 P_j(式(5)、(7))。P_i 发送调度服务器许

可消息至 P_j,若 P_j 运行正常并成功接收许可消息,则以站内广播形式发送调度服务器确认消息至站内其他服务器。若 P_j 未响应,则 P_i 将许可消息发送至负载值与 P_j 最接近的服务器。当其他服务器均没有发送确认消息时,由 P_i 作为当前站点调度服务器并广播调度服务器确认消息。

算法 2　调度服务器选择算法。伪代码如下所示。

输入:站点所有云服务器。

输出:站点调度服务器。

执行环境:当站点内任意一台服务器 P_i 不能找到调度服务器或发现调度服务器出现故障。

```
Scheduler( )
{
  k = 1; i = 1;
  if ( 服务器 P_i 不能找到调度服务器或发现调度服务器出现故
    障)
  {
    collect( );     //发送站内广播消息,收集其他服务器负载值
    list = sort( );          //对包括本服务器在内的所有站内
                             //服务器根据负载值升序排列
    point = length(list)/2;
    find(负载值处于队列中间位置的服务器 P_j);
    while(未收到调度服务器确认消息)
    {
      send "调度服务器许可消息至 P_j";
      if(P_j 运行正常并成功接收许可消息)
      {
        receive "调度服务器确认消息";break;
      }
      else
      {
        if (k==1)
          point = point - i; i ++;k = 0;
        else
          point = point + i; i ++;k = 1;
      }
      if (point <0 or point > length(list))
        broadcast "调度服务器确认消息";
      //P_i 成为调度器
    }
  }
}
```

步骤如图 2 所示。

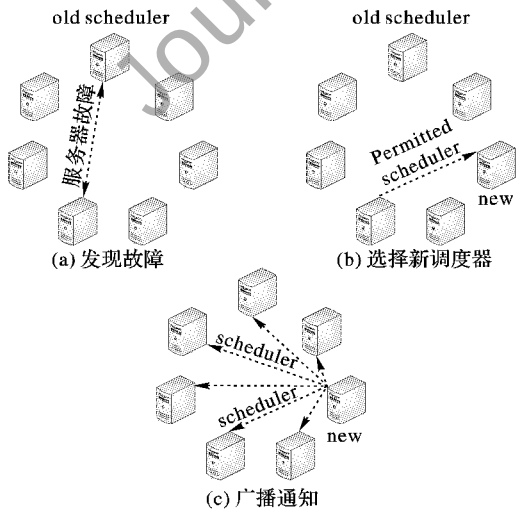

图 2　站点调度服务器选择

当选择负载值较大的服务器作为调度器时,因其执行效率较低,会影响站点的处理能力;当选择负载较轻的服务器作为调度器时,会减少适合执行任务的节点,导致站点负载不均衡。因此本文采用的策略是所选择的调度服务器负载处于负载列表中间位置,在不影响站点处理能力的前提下更好地实现站点负载均衡。

通过上述站点初始化及调度器选择算法,将云计算系统分为 3 层结构,依次为:站点组成的任务调度层、物理服务器组成的资源管理层和虚拟机组成的任务执行层。

2　负载均衡与容错

本文所述基于分层与容错机制的云计算负载均衡策略在站点内采用调度器集中管理,站点间采用分布式任务调度的方式实现云系统负载均衡,并通过容错机制有效利用空闲资源,提高任务执行成功率。

2.1　邻站点及负载信息

本文算法需要将云节点服务器在逻辑上组成邻站点,邻站点通过计算站点间信息传输质量形成。站点间信息传输质量 TQ 计算表达式如式(9):

$$TQ = TR/DW \tag{9}$$

其中:TR 表示站点信息传输速率,DW 表示站点间网络带宽。

根据式(9) 计算站点之间距离系数如式(10):

$$\partial = TQ_{ik}/TQ_{\min} \tag{10}$$

其中:TQ_{ik} 表示站点 S_i 与 S_k 之间的信息传输质量,$TQ_{\min}$ 表示与站点 S_i 距离最近站点之间的信息传输质量。根据式(10),当站点 S_i 与 S_k 之间的 ∂ 值在一定范围内时,认为 S_k 是 S_i 的邻站点,且站点信息加入 S_i 调度器的邻站点列表中,当 S_k 负载小于 S_i 时,则认为该站点是有效邻站点。根据实验及文献[8],当距离系数 $\partial = 1.375$ 时可以得到较好结果,因此当两站点间 $\partial \leqslant 1.375$ 时,则认为是邻站点,结构如图 3 所示。

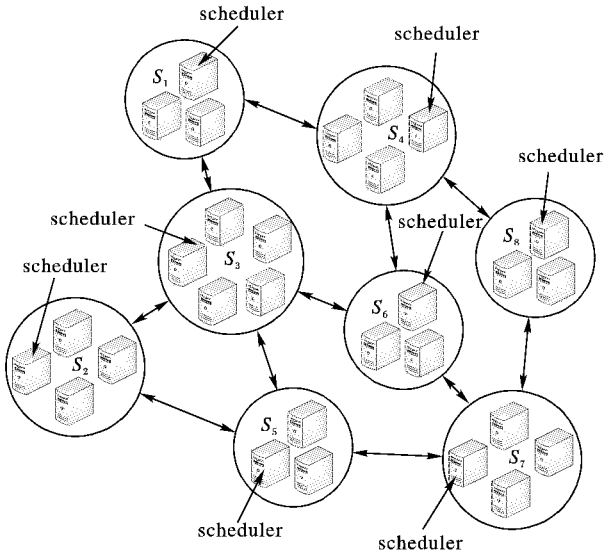

图 3　云计算分层结构

每个站点调度器维护其邻站点状态信息表 State。调度器可以在任意时间根据状态信息表及式(8)评估其邻站点的负载,状态信息表主要包含负载值和更新时间两个参数:(load,time)。云计算环境中分布有大量站点,频繁地读取其他站点的信息会产生多余的网络及系统开销,需要在保证信息及时更新的前提下尽可能减小开销,本文采用原站点发送任务调度请求时携带本站点及其邻站点负载信息至目标站点的策略。当站点 S_i 负载过重需要将任务调度至邻站点时,调

度器从有效邻站点中选择最优站点并把任务发送至该目标站点 S_k。在该请求中附加本站点及其所有邻站点的负载信息，即站点 S_i 调度器中的邻站点状态信息表。目标站点 S_k 接收该负载信息表后与其状态信息表对比，若包含相同站点，则根据时间戳更新负载信息。同样在目标站点 S_k 发送确认帧至 S_i 时附加本站点及其邻站点信息表，S_i 以同样的方式更新状态信息表。随着云计算任务的增加，任务调度频率也随之提高，站点负载信息更新在不增额外开销的前提下能得到及时更新，实现云计算系统服务器负载均衡[9-11]。

2.2 站点内负载均衡策略

站点是一个小型集群系统，而集中式负载均衡策略在小型系统中更能体现优越性：容易维护，安全性高，没有过多地暴露子节点。本文采用轮询与最小优先算法结合的策略，根据调度器收集的本站点所有服务器的负载值 LB_i 按从小到大排序形成列表，当有任务到达时按列表顺序从上到下轮询选择服务器进行任务分配。负载值根据周期 T 进行更新，以准确、实时评价服务器当前负载，而频繁的更新会增加系统开销，一般情况下把周期设置为 5 s 到 10 s[12]。

当某个时间点在负载表更新的同时有新任务到达，需要根据负载表进行资源分配，此时会出现数据读写不一致的问题。为避免这个问题，本文算法设置两张负载表，当有任务到达时将其中一张表用来选择定位服务器，另一张表进行下一个周期的负载信息更新，当本周期完成时跳转到另一张表进行任务分配定位，交替使用，有效避免读写不一致问题。

2.3 站点间负载均衡策略

站点间负载均衡采用邻站点分布式动态均衡策略。当有任务请求到达站点时，任务的分配要根据站点效率动态调整。当出现以下情况是需要对任务重新调度：1）当站点调度器接收任务请求消息时，站点无法提供任务所需全部资源；2）当任务在某站点执行时其实际执行时间超过预期；3）站点过载。当调度器探测到情况 1）和 3）时，需要从邻站点中选择有效站点进行任务调度；当探测到情况 2）时，对站点有效性进行评估，若有效则重新评估任务执行时间，若无效则选择有效站点进行任务调度。

动态调度策略目标是通过任务调度减小负载重与负载轻的站点间的负载差，实现系统负载均衡。由于站点负载信息是附加在任务调度请求消息中的，随着云计算系统任务调度的增加，站点调度器中邻站点负载信息能够较好地反映当前负载状态，增加了负载均衡算法的可信度。当站点有任务要调度至邻站点中负载最小的站点时，为了避免负载较小站点接收过多任务导致负载增大，需要引入控制机制。这里引入变量 *Threshold*，*Load_diff*，计算公式如式（11）、（12）：

$$Threshold = secondLightestLoad - lightestLoad \tag{11}$$

$$Load\_diff = sourceLoad - lightestLoad \tag{12}$$

其中：*lightestLoad*、*secondLightestLoad* 表示邻站点中负载值最小的两个站点负载值，*sourceLoad* 表示本站点负载值。控制条件为 $Load\_diff \geqslant Threshold$。

算法 3　负载控制算法。伪代码如下所示：

```
BEGIN
获取站点 S_i 负载值;
邻站点负载值从小到大排序;
WHILE 站点任务队列 Jobqueue(S_i).size > 0
{
  获取 lightestLoad, secondLightestLoad 值;
  计算 Threshold, Load_diff;
  WHILE Load_diff ≥ Threshold
  {
    SubmitJob(Load_diff, lightestLoadSite);
    获取站点 S_i 负载值;
    更新 Threshold, Load_diff;
  }
}
END
```

经过一段时间的调度，负载大的站点负载逐渐减小，而负载小的点不会增加过重的任务负担，使整个云计算系统趋于负载均衡状态。

2.4 容错模型

任务在执行时可能会因虚拟机错误、硬件错误等而导致执行失败，为了解决任务执行错误问题，本文在负载均衡中引入分布式容错机制，保证用户请求能及时完成，提高云计算系统的鲁棒性。通过上述负载均衡策略，当云计算各服务器达到负载均衡状态时，未利用的资源比例会相对增加，因此可以合理地利用这部分空闲资源用来实现容错机制。分布式容错基于点对点容错机制，即每一个站点都是另外一个对应站点的备份。当任务调度至某站点后，调度器对该任务进行独立备份，远程调度器将该备份任务调度至能够提供相应资源的邻站点。

当任务在站点中的一个服务器中执行时，调度器容错检测模块对执行过程进行检测，当任务被中断而执行失败时，容错管理模块将错误反馈至远程调度器，远程调度器发送执行备份任务命令至备份站点。为了避免站点接收过多备份任务而增加负载，降低容错效率，每个站点只能接收一个备份请求，当主任务执行成功时，主站点发送备份释放消息至备份站点，释放资源以接收其他站点的备份。当备份站点未收到备份执行消息和资源释放消息时，这里引入时间控制方法。当任务实际执行时间大于预计时间（即认为平均最小任务错误时间大于最大任务执行时间）时，备份站点认为主任务执行失败，立即执行备份任务，并释放资源以接收其他站点的备份[13]。

3 实验及结果分析

本文选用云计算仿真平台 CloudSim 对算法进行实验分析和评估，主要的评价指标为任务在云计算系统中的分配次数、完成响应时间。

实验首先对距离系数∂进行验证，通过系统平均响应时间测得较理想的∂系数值。实验中架设 200 台服务器的云系统，对∂值从 1.25～1.675 进行测试，对应的系统响应时间如图 4 所示。

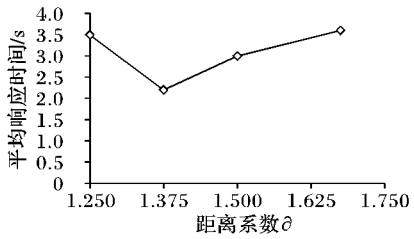

图 4　距离系数与平均响应时间关系

从实验结果得出当距离系数 $\partial = 1.375$ 时系统获得最短平均响应时间，因此当两个站点间 $\partial \leqslant 1.375$ 时，认为是邻站点较合适。

本文算法 Layer-Ftolent 与 Hybrid DLB 算法使用相同的服

务器、网络环境,且站点分组相同,其他具体参数值如表 1 所示。

表 1　对比实验参数

参数变量	参数值
服务器数量	100,200,300,400,500
任务个数	1 000
CPU 总需求	2 500 GHz
内存总需求	30 000 MB

为了更好地体现对比实验效果,这里假设每个任务都能提供对 CPU、内存等资源要求量,任务之间都是互相独立,且当任务执行时间超过 $(T + kT)/pe\_mips_i$,则认为任务超时,执行失败[14-15]。

实验所测试的数据为任务平均调度次数以及任务平均完成时间,包括任务调度时间和任务执行时间,实验结果如图 5 ~6 所示。

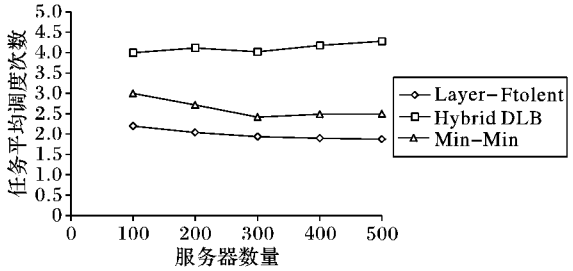

图 5　任务的平均调度次数对比

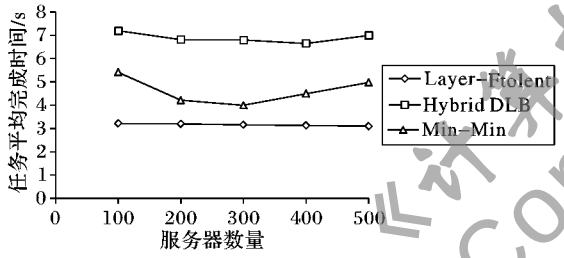

图 6　任务平均完成时间对比

其中 Min-Min 算法曲线是文献[8]中与 Hybrid DLB 对比所得出的实验结果。比较 Hybrid DLB 与改进的 Layer-Ftolent 云计算负载均衡策略,由图 5 可以看出,随着云计算服务器的增加,任务平均调度次数减小,说明能更快、更容易地找到能提供相应资源的服务器。对比两种算法,在相同数量的服务器时,Layer-Ftolent 算法平均调度次数减少 25% 左右,说明该算法在更短的时间内高效地找到了有效服务器,有效减少了因任务频繁调度使网络开销增大,服务器运行效率低下等方面的不利影响。

图 6 可以说明服务器数量对任务平均完成时间的影响。本文算法将任务的执行与调度,负载检测消息均控制在站点内和邻站点之间,从图 6 中可以看出,服务器的数量对任务完成效率影响较小,系统具有较好的可伸缩性,有效解决了 Hybrid DLB 由于服务器数量的增加使负载检测消息频繁地全系统广播导致系统开销增加、效率低下的问题。通过图 6 的对比,Layer-Ftolent 算法在任务平均完成时间上减小了 27% 左右,相应的任务执行效率提高了 27% 左右,说明算法能比较稳定地达到云计算负载均衡,使每个服务器在负载比较好的情况下运行,也反映出算法的容错机制有效避免了任务因服务器错误而长时间得不到响应的问题。

通过上述两个仿真实验,Layer-Ftolent 负载均衡策略较 Hybrid DLB 策略在云计算负载均衡、任务响应时间及任务调度次数方面有了大幅度改善。

4　结语

本文研究了云计算环境下基于分层与容错机制的负载均衡策略,对文献[7]中的混合动态负载均衡算法进行了改进。比较全面地对服务器实时负载进行数学评价,通过实验测定较理想的距离系数,从实验数据可以得出,将负载信息交换控制在邻站点之间,并通过任务调度请求进行同步以及利用服务器空闲资源进行容错等方法可以使云计算系统在较短时间内达到负载均衡,减小请求响应时间。当然,本文方法未考虑云计算中服务器间实际距离带来的网络延迟,这是下一步将要解决的问题。

参考文献:

[1] MONDAL B, DASGUPTA K, DUTTA P. Load balancing in cloud computing using stochastic hill climbing - a soft computing [C]// C3IT-2012: Proceedings of the 2nd International Conference on Computer, Communication, Control and Information Technology. New York: Elsevier, 2012: 783 - 789.

[2] BABU L D, KRISHNA P V. Honey bee behavior inspired load balancing of tasks in cloud computing environments [J]. Applied Soft Computing, 2013, 13(5): 2292 - 2303.

[3] 姚婧,何聚厚. 基于自适应蜂群算法的云计算负载平衡机制[J]. 计算机应用,2012,32(9):2448 - 2450.

[4] MUKHERJEE K, SAHOO G. Mathematical model of cloud computing framework using fuzzy bee colony optimization technique [C] // Proceedings of the 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies. Washington, DC: IEEE Computer Society: 2009: 664 - 668.

[5] RAHMEH O A, JOHNSON P, TALEB-BENDIAB A. A dynamic biased random sampling scheme for scalable and reliable grid networks [EB/OL]. [2013-04-25]. http://www.cms.livjm.ac.uk/taleb/Publications/08/Infocomp'08.pdf.

[6] 余郭福,李鸿健,唐红. 基于反馈机制的动态负载平衡算法研究[J]. 计算机应用研究,2012,29(2):527 - 529.

[7] MEHTA M A, JINWALA D C. Novel algorithms for load balancing using hybrid approach in distributed systems [C] // Proceedings of the 2nd IEEE International Conference on Parallel, Distributed and Grid Computing. Washington, DC: IEEE Computer Society, 2012: 27 - 32.

[8] BALASANGAMESHWARA J, RAJU N. A hybrid policy for fault tolerant load balancing in grid computing environments [J]. Journal of Network and Computer Applications, 2012, 35(1): 412 - 422.

[9] KHANLI L M, RAZZAGHZADEH S, ZARGARI S V. A new step toward load balancing based on competency rank and transitional phases in grid networks [J]. Future Generation Computer Systems, 2012, 28(4): 682 - 688.

[10] 杜垚,郭涛,陈俊杰. 云环境下机群弹性负载均衡机制[J]. 计算机应用,2013,33(3):830 - 833.

[11] 王少峰,周忠,吴威. 一种面向分布式虚拟环境的分层迭代负载均衡算法[J]. 软件学报,2008,19(9):2471 - 2482.

[12] FATTEBERT J L, RICHARDS D F, GLOSLI J N. Dynamic load balancing algorithm for molecular dynamics based on Voronoicells domain decompositions [J]. Computer Physics Communications, 2012, 183(12): 2608 - 2615.

[13] SHARMA M, SHARMA P. Performance evaluation of adaptive virtual machine load balancing algorithm [J]. International Journal of Advanced Computer Science and Applications, 2012, 3(2): 86 - 88.

[14] 陈一骄,卢锡城,时向泉,等. 一种面向回话的自适应负载均衡算法[J]. 软件学报,2008,19(7):1828 - 1836.

[15] 冯小靖,潘郁. 云计算环境下的 DPSO 资源负载均衡算法[J]. 计算机工程与应用,2013,49(6):105 - 108.