doi:10.3772/j.issn.1002-0470.2015.03.006

云环境下老化感知的任务调度和运行管理框架①

李 焱②*** 张 鸿** 刘欣然③**

(*中国科学院计算技术研究所 北京 100190)

(**国家计算机网络应急技术处理协调中心 北京 100029)

摘 要 研究了云计算系统现有任务调度和运行管理机制,指出在云计算环境下,基础性支撑软件虚拟机和虚拟机管理器在长期运行过程中均会表现出老化现象,从而降低了其性能,进而影响了任务的调度和执行效果。在一定程度上影响了服务质量(QoS)。针对这种情况,提出了一种老化感知的任务调度和运行管理框架,该框架能在任务调度和运行时感知底层虚拟机和虚拟机管理器的老化状况,并通过及时再生手段恢复虚拟机性能,从而降低老化所带来的负面影响,提升系统可用性。基于 CloudSim 仿真工具,在该框架下实现了 Max-Min 调度算法,并对比说明了该框架的有效性。

关键词 云计算,虚拟机,软件老化,任务调度,资源管理

0 引 言

云计算是一种新型的计算模式,它提供了一种按需访问共享资源池(包含诸如网络、服务器、存储、应用等资源)的方便方法,并能以最小的管理代价实现资源发布。云计算以服务的方式提供资源,缩短了上层应用的开发周期,节省了系统的部署成本,因具有资源共享、弹性配置和按需服务等特点,备受产业界和学术界的关注。由于云计算环境下应用多样、任务极多,因而任务调度和资源分配是提高云平台服务质量的关键。现有云平台任务调度算法大都沿用网格计算领域任务调度方式,专门针对云计算中的任务调度的研究不多。

构建云计算基础架构的一种重要途径是虚拟化,虚拟机和虚拟机管理器是云计算平台的基础性支撑软件。虚拟化使得多个虚拟机能够同时安全地运行在同一台物理机上,屏蔽了物理资源的差异所带来的影响,提高了资源利用率。近年来的研究发现,当软件连续运行较长时间时,软件内部的一些错误条件的积累会导致软件性能衰退甚至停止运行,这种现象称为软件老化(software aging)[1],作为需要长期运行的软件,虚拟机和虚拟机管理器也存在老化现象[2-6]。文献[4]的实验表明,长时间运行的虚拟机和虚拟机管理器软件会挤占业务系统正常运行所需的 CPU 和内存资源,使得业务响应时间由运行之初的 0.25s 增至 20 余秒。虚拟支撑软件系统中的老化现象会影响云平台中任务的运行效率进而导致不能以预先定义的服务等级协议(service-level agreement,SLA)水平提供服务。但现有云平台任务调度及运行管理方法大都未考虑资源的可用性[7-10]或是在考虑资源可用性时并未考虑底层虚拟资源老化所带来的影响[11-14]。因此,本文提出了一种老化感知的任务调度和运行管理框架。在任务调度和运行时,该框架感知底层虚拟机和虚拟机管理器的老化状态,及时再生老化的资源,提高任务执行效率。本研究并通过 CloudSim 模拟工具说明了该框架的有效性。

① 973 计划(2011CB302605)资助项目。

② 男,1984 年生,博士;研究方向:分布式计算,云计算;E-mail:liyan@ncic.ac.cn

③ 通讯作者,E-mail:lxr@cert.org.cn

(收稿日期:2014-11-13)

1　相关工作

1.1　软件老化

软件老化在需要长期运行的软件系统中较为普遍,它降低了系统的性能,增加了系统失效率。操作系统、虚拟机、虚拟机管理器、Java 虚拟机、Web 服务器、Eucalyptus 云架构软件、视频点播系统等软件中都发现存在软件老化现象[2-6,15]。为了消除或减少软件老化所造成的软件失效,人们提出了一种称为软件再生的预防性维护技术[1]。软件再生采用周期性地清除系统内非正常状态(如重启)等方法使系统恢复至预置健壮状态的方法,避免可能出现的潜在故障,使系统能够长期正常运行。

软件老化和软件再生主要研究的问题包括以下三个部分:何时再生,何处再生,如何再生。何时再生主要用以寻找软件的最佳再生时间,太早的话无再生必要,太晚的话不能及时有效地消除系统潜在故障;何处再生主要用以确定再生的对象,宿主机操作系统、虚拟机管理器、虚拟机、虚拟机上操作系统、业务系统等任何一种软件中的老化现象都会造成服务性能的下降甚至崩溃,对非老化的软件系统进行再生操作是无用的;如何再生主要关注再生的具体方法,是通过简单的软件重启还是需要通过热迁移等手段以保证再生过程中业务的不间断运行。

软件老化和再生的研究方法可以分为两类:基于模型的方法和基于测量的方法。基于模型的再生通过计算最佳的系统可用性或相关性能度量来得到最佳的再生周期,常用连续时间马尔科夫链、半马尔科夫过程或随机 Petri 网等对系统进行建模;基于测量的再生对实时采集到的系统资源参数进行统计分析,预测关键资源耗尽的预期时间,进而计算得出再生时间[15]。然而,简单的重启再生策略一方面会导致业务的中断,另一方面会清除系统缓存,降低应用缓存命中率,无法使服务响应恢复至再生前的水准。为此,文献[2]提出了一种针对虚拟机管理器的快速热重启方法,重启前用一个虚拟机保存管理器上剩余所有虚拟机内存映像,而后挂起虚拟机,重启后快速加载映像,降低了再生时间消耗并保证了再生后系统性能。文献[5]探讨了虚拟环境下,老化的探测方法,并通过实验比对物理机和虚拟机的性能下降趋势,得出虚拟机的老化速度要快于物理机的老化速度。文献[16]提出了一种不影响虚拟机运行的有效的虚拟机管理器再生方法,将虚拟机热迁移技术引入再生过程以优化资源的利用,引入随机 Petri 网并提出了一种基于时间的再生模型对效果进行了分析。在云计算环境下,虚拟机的广泛使用使得软件老化是一个不可避免的现象[6],已有很多学者对此开展了研究[2-6,16]。

1.2　任务调度

任务调度是计算机领域的经典问题,操作系统、网格计算等在任务调度上已有非常多的研究。云计算环境下任务调度的具体方法大都基于之前的研究成果。Hadoop 平台默认的调度方法基于先来先服务(FIFO)的策略。为避免短小任务的长期等待,FaceBook 提出了一种称之为公平性的调度策略,将任务分成不同的池组,每个任务池保证拥有一定量的执行时间片[17]。

从任务间是否存在关联性来划分,云计算环境下的任务调度研究可分为基于任务间独立假设的调度[7,9,11,14]和考虑任务间依存关系的调度[8,12,13]。如经典的 Min-Min、Max-Min、RASA 等算法都未考虑任务之间的依存关系,适用于 bag of task 型的任务调度[18]。文献[8,12,13]纳入对任务间依存关系的考虑。此类调度算法大多用有向无环图(DAG)对任务进行建模,重点提高关键路径上任务的执行效率。

任务调度已被证明为一个 NP 问题,很难找到一种最优的任务调度算法以满足所有的约束条件[7]。现有任务调度的目标主要有以下两种:用户利益驱动的调度方法和考虑服务提供成本的调度方法。前者主要考虑用户的服务质量,如任务完成时间最短、服务质量最高等;后者在满足用户服务的基础上还需考虑云平台服务提供商的服务成本,如资源负载均衡、系统资源利用率最高、降低云平台能耗等[19,20,10,21]。还有一些调度策略主要考虑调度方法的可扩展性,如多级任务调度策略等[13]。

然而,云计算环境下,现有调度方法对底层虚拟支撑平台的老化问题考虑不足。可用性调度方法主

要通过副本或是为关键任务分配较为可靠的计算资源、网络资源来完成[11,12,14]，但软件老化使得计算能力强的资源在运行过程中也不可避免地会发生性能下降现象甚至崩溃，影响了任务实际运行效果。

2　框架描述

云计算环境下，任务调度根据一定的策略将不同用户所提交的一批任务分配到合适的资源上去执行。调度策略由调度的目标所决定，常见的调度目标是最小化整批任务的完成时间跨度，如 Min-Min、Max-Min 算法等[7]。其他的调度目标还包括负载均衡、系统吞吐量、服务可靠性、服务的成本（如节能）等。

2.1　任务模型

为便于后续描述，本文参照文献[13]利用有向无环图（DAG）描述任务：$\boldsymbol{G}=(\boldsymbol{T},\boldsymbol{E})$。$\boldsymbol{T}=\{\boldsymbol{t}_0,\boldsymbol{t}_1,\cdots,\boldsymbol{t}_{n-1}\}$ 表示需要调度的任务集合，n 表示任务总数；$\boldsymbol{E}$ 为边的集合，每条边表示两个任务间的先序关系，$\boldsymbol{E}$ 为空则表示各任务间是独立的。每个任务 $\boldsymbol{t}$ 可表示为 $\boldsymbol{t}=\{t_{\text{length}},\ t_{\text{status}},\ t_{\text{rres}},\ t_{\text{deadline}},\ t_{\text{input}}\}$，各属性含义如下：

（1）t_{length} 表示任务的长度，表明任务的计算量，以预估任务在特定资源上的期望执行时间；

（2）t_{status} 表示任务的状态，可能的状态包括就绪（READY）、执行（EXEC）、挂起（SUSPEND）、唤醒（RESUMED）、撤销（CANCEL）、完成（FINISH）和失败（FALSE）等，具体的和研究背景有关；

（3）t_{rres} 表示任务的资源需求，该项可根据实际情况进行深入刻画，如可细分为对计算能力、存储能力、带宽能力或是所在地域等，亦可对各项需求加上可用性的要求；

（4）t_{deadline} 表示任务的最晚完成时间，该项是任务调度成功与否的判断标准；

（5）t_{input} 表示任务执行所需的关键输入资源，如数据等，当结合数据的存放地点进行任务调度时，需用到该项值。

2.2　资源模型

本文主要考虑的资源为云计算环境下的虚拟机。$\boldsymbol{V}=\{v_0,v_1,\cdots,v_{k-1}\}$ 表示系统中的虚拟机集合，k 表示虚拟机的个数。虚拟机可进一步刻画为 $\boldsymbol{v}_i=\{v_{rm},v_{ai},v_{st},v_{task}\}$，$v_{rm}$ 表示该虚拟机所依托的虚拟机管理器编号（为简便起见，该编号取虚拟机管理器集合中的下标值），v_{ai} 表示虚拟机的老化指数（取值为 0 到 1 间的数值，值越小表示老化程度越高）。v_{st} 表示虚拟机的状态，可能的状态范围包括：就绪健康态（RH）、执行健康态（EH，表示正在有任务执行且处于健康状态）、就绪老化态（RA，表示资源处于就绪状态但老化指数已超过一定的阈值）、执行老化态（EA，表示正在有任务执行但资源的老化指数已超过一定的阈值）、再生状态（SR，表示正对资源执行再生操作）、失效状态（F，表示资源处于失效状态）。v_{task} 表示虚拟机正在执行的任务。引入老化及再生之后的虚拟机状态转移图如图 1 所示。

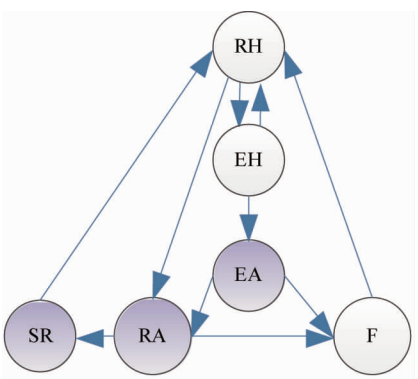

图 1　虚拟机状态转移图

虚拟机管理器（virtual machine monitor，VMM）的集合用 $VMM=\{vmm_0,vmm_1,\cdots,vmm_{m-1}\}$ 表示，m 表示虚拟机管理器的个数。通常，一个虚拟机管理器可管理多个虚拟机资源。虚拟机管理器可进一步刻画为 $vmm_i=\{vmm_{rm},\ vmm_{ai},\ vmm_{st}\}$，$vmm_{rm}$ 表示该虚拟机管理器所管理的虚拟机编号的集合，vmm_{ai} 表示虚拟机管理器的老化指数（取值同虚拟机的老化指数），vmm_{st} 表示虚拟机管理器的状态。因为虚拟机管理器不执行用户的任务，故可能的状态范围包括健康状态（H）、老化状态（A）、再生状态（R）、失效状态（F），状态转移图如图 2 所示。

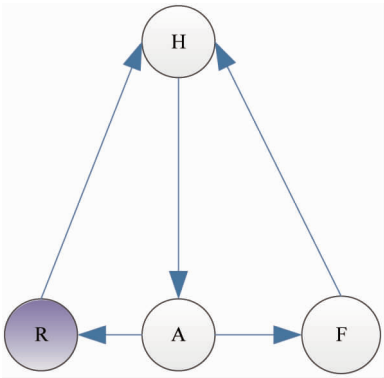

图 2　虚拟机管理器状态转移图

2.3　框架描述

本文基于上述任务和资源模型提出的任务调度和运行管理框架如图 3 所示,主要模块包括任务调度器、调度策略及资源信息库、老化探测、软件再生等,各模块功能描述如下:

(1) 老化探测(AgingDetector)。该模块对虚拟机和虚拟机管理器老化状态进行监测,并实时更新虚拟机和虚拟机管理器的老化指数 v_{ai}、vmm_{ai}。确定软件老化状态的方法主要有两种,即基于模型的方法和基于测量的方法[3-6],本框架用基于测量的

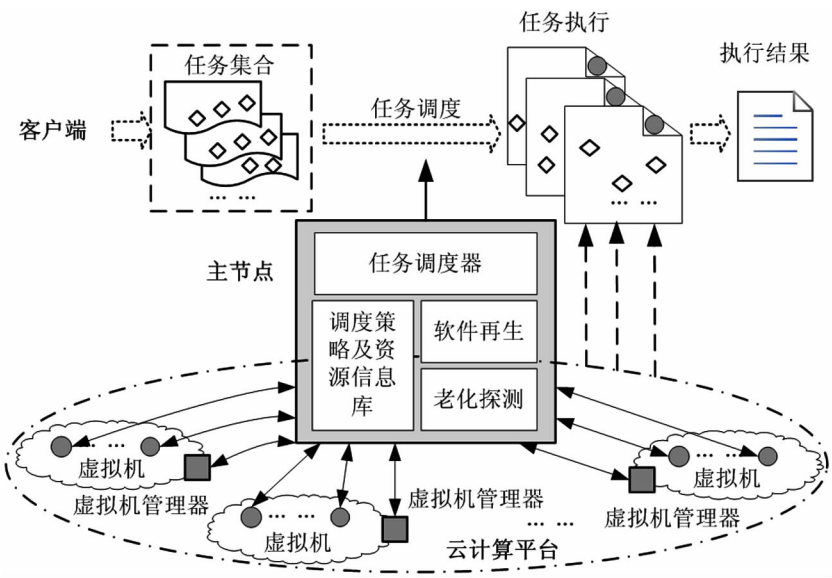

图 3　老化感知的任务调度和运行管理框架

方法实现老化探测。虚拟机上运行采集代理,定时收集内存使用率、CPU 使用率等指标以反映虚拟机老化状态,并向老化探测模块反馈相关指标数据;已有研究表明,可用内存是反映软件老化状态的一项重要指标,为简单起见本框架用该指标的变化情况反映虚拟机、虚拟机管理器老化指数,具体算法流程如图 4 所示。为满足老化探测需求,亦可在此框架下定制特有的老化探测方法。

(2) 调度策略及资源信息库(PolicyandResInfoDB)。该模块主要用以存储可供用户选择的任务调度策略及资源信息。调度策略是具体调度算法的实现,如 Min-Min、Max-Min 或遗传算法等;资源信息除了包括虚拟机、虚拟机管理器的编号、所在物理位置等基本信息外,还包括虚拟机和虚拟机管理器的老化指数、状态等信息。老化探测模块会根据探测结果及时更新资源信息。资源信息是根据调度策略实施任务调度及软件再生的重要依据。

(3) 软件再生(SoftRej)。软件再生模块执行运行管理的功能,当探测到虚拟机或虚拟机管理器老化指数达到一定阈值时,软件再生模块负责根据资源信息采取一定策略对虚拟机或虚拟机管理器执行重启操作,以将资源能力恢复到初始状态。实际再生时,可根据任务是否可中断采取不同再生策略。对于不可中断任务,可采取虚拟机迁移等手段将任务移至健康的虚拟机上执行,而后对原虚拟机执行再生操作。具体算法流程如图 5。

```
AgingDetect(i){//周期性探测老化状态,t 为探测时间
    PolicyandResInfoDB pdb;//信息库
    for(v_i in pdb.vList()){//针对每个虚拟机
        double resp_t= respond(v_i);//用探测程序探测虚拟机响应.也可用模型建模方式得出 t 时刻虚拟机的性能参数
        if(resp_t > resp_t-1 +f)//超过上一测试点响应一定值,断定存在老化现象
        if(v_i.v_st=RH)=RA;//状态改为就绪老化态
        if(v_i.v_st=EH)=EA;//状态改为运行老化态
        v_i.v_ai= v_i.v_ai • resp_t-1/resp_t;//更新老化指数
        if(v_i.v_ai <=pdb.threshold4vm)//老化到一定度
            SoftRej(v_i);//再生该虚拟机}
    for(vmm_i in pdb.vmmList()){//针对每个 vmm_i
        double avm_t= avmemory(vmm_i);//探测系统可用内存指标.也可用建模方式得出 t 时刻管理器性能
        if(avm_t+f < avm_t-1)//可用内存降低一定阈值
        vmm_i.vmm_st=A;//状态改为老化态
        vmm_i.vmm_ai= vmm_i.vmm_ai• avm_t/avm_t-1;//更新老化指数
        if(vmm_i.vmm_ai <=pdb.threshold4vmm)//老化到一定度
            SoftRej(vmm_i);//再生虚拟机管理器}
```

图 4　老化探测流程

```
SoftRej(vmm){//虚拟机管理器再生
    vmm.vmm_st =R;
    for(i in vmm.vmm_rm){//对承载的每个虚拟机
        lock(v_i);//锁定，不能再分配任务
        if((v_i.v_st=EH)||(v_i.v_st =EA)){//虚拟机忙碌
            suspend(v_i.v_task);//挂起正在执行的任务，若不可挂起可采用虚拟机迁移的方式完成
            if(v_i.v_st =EH) v_i.v_st =RH;
            else v_i.v_st =RA;}
        reset(vmm)//重启虚拟机管理器
        vmm. vmm_st =H;
        resume(vmm);//恢复管辖范围内所有虚拟机中任务}
SoftRej(v){//虚拟机再生
    if((v.v_st)=RA){//处于就绪老化态
    v.v_st =SR;//处于再生状态
    reset(v);//重启该虚拟机
    v.v_st =RH;  //恢复至初始健康态,老化指数为 1}
    if((v.v_st)=EA){//处于执行老化态
    suspend(v.v_task);  //挂起正执行任务,若为不可暂停任务可通过虚拟机迁移方式完成;v 的状态会变为 RA
    v.v_st =SR;
    reset(v);//重启该虚拟机
    v.v_st =RH;  //恢复至初始健康态
    resume(v);//恢复 v 上任务的运行}}
```

图 5　再生流程

（4）任务调度器（Scheduler）。任务调度器是该框架的核心功能模块，它根据任务到达情况，并结合用户的实际需求选择合适的调度策略进行任务调度。用户需求和资源状态是调度器调度的主要依据。例如，为保证服务质量，对于高优先级用户提交的任务或是运行时间较短且任务执行截止期限较近的任务，调度器会优先选择健康的虚拟机执行这些任务。在任务执行时，当探测到虚拟机老化到一定程度时，会启动再生模块及时恢复虚拟机服务能力，以提高任务的执行效率。任务执行流程见图 6。

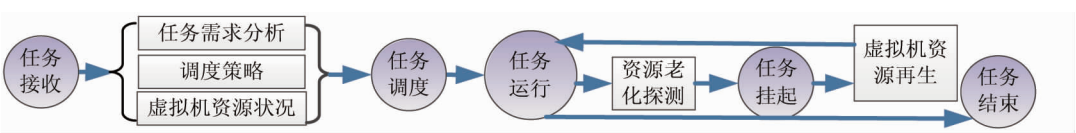

图 6　任务执行流程

3　仿真研究与结果分析

为表明框架的有效性，本文将 Max-Min 任务调度方法纳入该框架，并使用澳大利亚墨尔本大学推出的 CloudSim 进行仿真实验。Max-Min 算法是一种经典的任务调度方法，主要思想是优先调度执行时间长的任务，并将该任务放置于预计能够最早完成该任务的资源上执行，以最小化批量任务的执行时间跨度，伪代码如下：

Max-Min 任务调度方法

```
Max-MinSch(tList){
  for(task in tList){//找出任务执行时长最长的任务,直至 tList 为空
    maxT = tList.maxTask();
    for(vm in V){//
      minVm = V.minExeCTime();//找出预计最早完成该任务的资源
        maxT run(minVm); }
        tList.remove(maxT);//去除该任务
          minVm.execTime += maxT.runtime(minVm);//更新该虚拟机能够执行任务的时间}}
```

3.1　实验设计

本文用 CloudSim[22] 模拟出一个数据中心中的若干个虚拟机资源。虚拟机的配置为:单核 1000mips 计算能力、512MB 内存、映像大小 10GB。为分析虚拟机数量 *vmNum* 的影响,进行了 5 次分组实验,每次分组实验所执行的目标虚拟机数量分别为 3、4、5、6、7,并考虑不同的再生阈值,模拟得出各种情况下批量任务的执行跨度,并和原始 Max-Min 算法的执行效果进行比较。

(1)任务样例。待调度的任务有 25 个,执行时间在 0 到 200min 内随机分布,如图 7 所示。

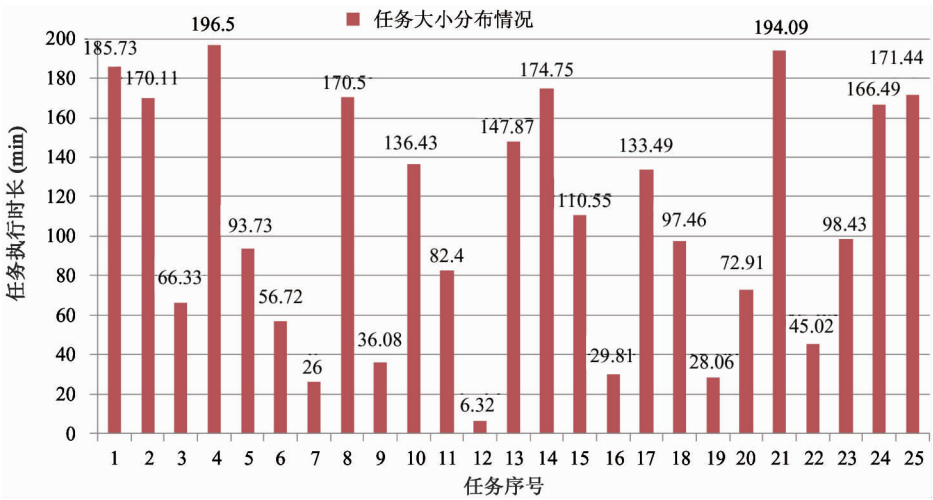

图 7　调度的任务列表

(2) 老化模拟及再生阈值设置。文献[4]实际测试标明,虚拟机老化导致的处理 HTTP 请求响应时间近似呈线性增长趋势;文献[23]也同样表明 Web 服务器响应时间呈线性增长趋势。因此,本实验假设虚拟机任务响应时间消耗也呈线性增长趋势,归一化常数值后取 $r = 0.002t + 1$(r 为响应时间,t 为运行时间长度,单位为 min)。虚拟机的性能和响应时间成反比,即 $1/r$ 能够反映虚拟机的性能。因此,本实验用 $1/(0.002t + 1)$ 模拟虚拟机性能变化趋势。虚拟机运行时间越长,性能变得越低。虚拟机完全健康状态性能为 1,再生阈值分别取 0.9、0.8、0.7、0.6、0.5、0.4、0.3 等 7 个值。阈值 0.8 表示探测到虚拟机的性能衰退至 0.8 时需进行再生操作,之后虚拟机恢复至完全健康状态,即性能恢复成 1。设 *t-continue* 表示虚拟机至少需连续运行的时间使得性能衰退至某阈值,进而触发再生操作。据此,算出 *t-continue* 值见表 1。

表 1　不同阈值对应的虚拟机再生时间点

阈值	0.9	0.8	0.7	0.6	0.5	0.4	0.3
t-continue (min)	56	125	214	333	500	750	1167

(3) 虚拟机再生时间消耗。虚拟机重启时间消耗依赖于虚拟机内存大小,文献[2]表明内存大小为 4GB 的虚拟机,重启时间在 2min 左右。故本实

验设置虚拟机再生时间消耗为 2min。

3.2 实验结果与分析

5 次分组实验结果如图 8 所示,横坐标表示目标虚拟机资源的个数,纵坐标表示上述 25 个任务在不同策略下的执行时间跨度。很明显,虚拟机资源越少,同一批任务执行时间跨度就越大。当有 7 个虚拟机资源时,上述任务至多需要 463.98 分钟执行完成,但只有 3 个虚拟机参与执行时,同一批任务需 2200 余分钟才能完成执行。

各次分组实验结果中最右侧的值表示用原始 Max-Min 算法执行的时间跨度,因为未考虑虚拟机老化所带来的性能损耗,导致完成同批次任务所需的时间最长。左侧为在本文所述框架下的 Max-Min 算法执行结果,模拟时虚拟机再生阈值不同导致执行结果有所差别。最左侧为阈值取 0.9 的实验结果,即监测到虚拟机性能降低到初始性能的 90% 以下时,运行管理框架会待该虚拟机空闲时再生该虚拟机资源,提高了任务执行效率。阈值越大,框架对虚拟机老化程度越敏感,再生手段使得虚拟机在健康状态工作的时间越长,批量任务执行的效率也就越高。但任务执行总时间的长短会影响框架的效率提升空间。当虚拟机个数为 3 时,每个虚拟机需运行较长时间才能完成任务,而虚拟机连续运行时间越长性能下降越明显,框架对于任务执行效率的提升越明显,当阈值为 0.9 时,能将任务执行时间跨度减少 52.5%,即使在阈值取 0.3 时(即监测到虚拟机性能下降 70% 时才采取再生措施),也能将任务执行跨度减少 25%。当虚拟机个数为 7 时,为完成同批次任务每个虚拟机运行时间并不长,虚拟机性能衰退并不会很明显,但在阈值取 0.9 时,框架仍能将任务执行时间跨度减少 16.5%。上述实验说明该框架是有效的。

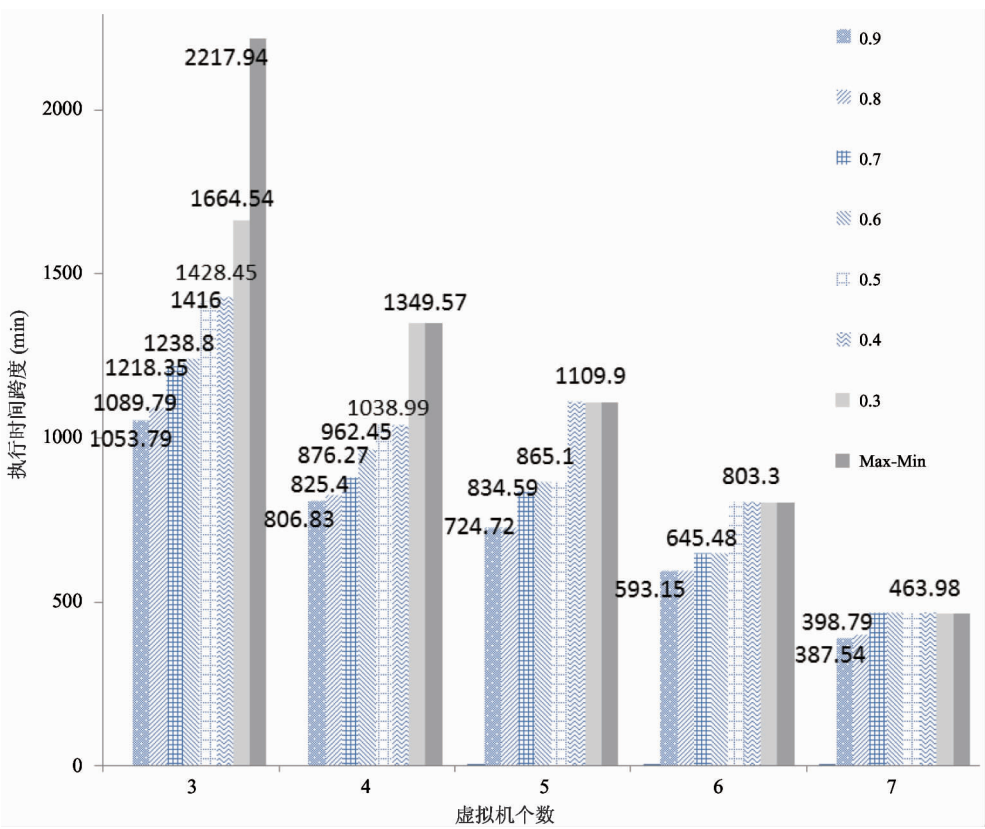

图 8 不同调度和管理策略下批量任务执行时间跨度

4 结 论

云计算环境下,虚拟机软件需长时间运行,但老化现象会导致虚拟机性能下降进而影响任务的调度和执行效果。针对这种情况,本文提出了一种老化感知的任务调度及运行管理框架,利用这一框架可

在任务调度和运行时及时感知虚拟机软件老化状态,通过再生手段恢复虚拟机性能来提升任务运行效率。本研究通过 CloudSim 模拟证明了该框架的有效性。下一步将重点对虚拟机软件的老化预测方法进行研究,以提升虚拟机再生的效率。

参考文献

[1] Huang Y, Kintala C, Kolettis N, et al. Software rejuvenation: Analysis, module and applications. In: Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing, IEEE, Pasadena, USA, 1995. 381-390

[2] Kourai K, Chiba S. A fast rejuvenation technique for server consolidation with virtual machines. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Edinburgh, UK, 2007. 245-255

[3] Araujo J, Matos R, Maciel P, et al. Experimental evaluation of software aging effects on the eucalyptus cloud computing infrastructure. In: Proceedings of the Middleware 2011 Industry Track Workshop. ACM, Lisbon, Portugal, 2011. 4

[4] Matos R, Araujo J, Alves V, et al. Experimental evaluation of software aging effects in the eucalyptus elastic block storage. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Seoul, Korea, 2012. 1103-1108

[5] Cui L, Li B, Li J, et al. Software aging in virtualized environments: Detection and prediction. In: Proceedings of the 2012 IEEE 18th International Conference on Parallel and Distributed Systems. IEEE Computer Society, Singapore, 2012. 718-719

[6] HemadeviM N M, Dhivya M M, Manikandan M K. A survey on software aging and rejuvenation in server virtualized system. *International Journal of Science, Engineering and Technology Research*, 2013,2(11): 2069-2073

[7] Elzeki O M, Reshad M Z, Elsoud M A. Improved max-min algorithm in cloud computing. *International Journal of Computer Applications*, 2012, 50(12): 22-27

[8] 刘少伟, 孔令梅, 任开军等. 云环境下优化科学工作流执行性能的两阶段数据放置与任务调度策略. 计算机学报, 2011, 34(11): 2121-2130

[9] 史少锋, 刘宴兵. 基于动态规划的云计算任务调度研究. 重庆邮电大学学报: 自然科学版, 2012,24(6)

[10] 肖艳文, 王金宝, 李亚平等. 云计算系统中能量有效的数据摆放算法和节点调度策略. 计算机研究与展, 2013, 50: 342-351

[11] Fu S. Failure-aware resource management for high-availability computing clusters with distributed virtual machines. *Journal of Parallel and Distributed Computing*, 2010, 70(4): 384-393

[12] 曹洁, 曾国荪, 钮俊等. 云环境下可用性感知的并行任务调度方法. 计算机研究与发展, 2013, 50(7): 1563-1572

[13] 李文娟, 张启飞, 平玲娣等. 基于模糊聚类的云任务调度算法. 通信学报, 2012, 33(3): 146-154

[14] Qin X, Xie T. An availability-aware task scheduling strategy for heterogeneous systems. *Computers, IEEE Transactions on*, 2008, 57(2): 188-199

[15] 杜小智, 齐勇, 鲁慧民等. 视频点播系统的软件老化估计和预测. 计算机研究与发展, 2012, 48(11): 2139-2146

[16] Paing A M M, Thein N L. High availability solution: Resource usage management in virtualized software aging. *International Journal of Computer Science & Information Technology*, 2012, 4(3): 85-99

[17] Zaharia M. Job Scheduling with the Fair and Capacity Schedulers. Hadoop Summit, 2009, 9

[18] Legrand A, Touati C. Non-cooperative scheduling of multiple bag-of-task applications. In: Proceedings of the 26th IEEE International Conference on Computer Communications, Anchorage, USA, 2007. 427-435

[19] 左利云, 曹志波. 云计算中调度问题研究综述. 计算机应用研究, 2012, 29(11): 4023-4027

[20] 林伟伟, 齐德呈. 云计算资源调度研究综述. 计算机科学, 2012, 39(10): 1-6

[21] Zhang W, He H, Chen G, et al. Multiple virtual machines resource scheduling for cloud computing. *Appl. Math*, 2013, 7(5): 2089-2096

[22] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 2011, 41(1): 23-50

[23] Li L, Vaidyanathan K, Trivedi K S. An approach for es-

timation of software aging in a web server. In: Proceedings of the 2002 International Symposium on Empirical Software Engineering, IEEE, Nara, Japan, 2002. 91-100

An aging-aware task scheduling and executing framework for cloud computing systems

Li Yan[*,**], Zhang Hong[**], Liu Xinran[**]

([*]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]National Computer Network Emergency Response Technical Team, Beijing 100029)

Abstract

The study analyzed the current task scheduling and management mechanism of cloud computing systems, and pointed that in cloud environments, aging of the basic software of virtual machines and virtual machine monitors occurs in their long time working, leading to their performance degradation, further, to the decline of the task scheduling effect, and finally the quality of services (QoS). In view of this problem, an aging-aware task scheduling and executing framework for cloud computing was proposed. The framework can sense the aging status of virtual machines and virtual machine monitors during task scheduling and executing, and reduce the negative effect of software aging by rejuvenating the performance of virtual machines to improve the availability of cloud computing systems. The task scheduling algorithm of Max-Min was realized under the framework based on the simulation tool of CloudSim to verify the effectiveness of the framework.

Key words: cloud computing, virtual machine, software aging, task scheduling, resource management