基于 Xen 的虚拟机迁移时内存优化算法

陈廷伟 张 璞 张忠清

(辽宁大学信息学院 沈阳 110036)

摘 要 为了在云计算环境下进行虚拟机迁移,Xen 迁移时采用比较传递页位图和跳过页位图的方式来判断内存页是否重传。针对页位图比较带来多次重传增加网络传送开销的问题,提出基于 AR 模型的内存优化算法,该算法根据所有记录的内存页修改时间间隔来预测内存页的下次修改时间,当下次修改时间大于某个阈值时进行重传。实验结果表明,基于 AR 模型的内存优化算法缩短了虚拟机迁移的时间,减少了虚拟机迁移时的网络开销,保证了同台服务器上其它虚拟机的网络带宽应用。

关键词 云计算,虚拟机,Xen 迁移,AR 模型

中图法分类号 TP317.1 **文献标识码** A

Memory Optimization Algorithm of Migration Based on Xen Virtual Machine

CHEN Ting-wei ZHANG Pu ZHANG Zhong-qing

(College of Information, Liaoning University, Shenyang 110036, China)

Abstract In order to migrate the virtual machine in the cloud computing environment, Xen compares the send-page bitmap with the skip-page bitmap when migrating, and determines whether memory pages are retransmitted or not. Aiming at the problem that multiple retransmissions increase network transmission overhead from the comparison, this paper proposed a memory optimization algorithm based on AR model. According to all recorded intervals in which a memory page is modified, the new algorithm predicts the next modification time of the memory page and retransmits the memory page when the next modification time is greater than a threshold. The results show that the optimization algorithm shortens the migration and network overhead of the virtual machine and ensures the network applications of other virtual machines on the same server.

Keywords Cloud computing, Virtual machine, Xen migration, AR model

1 引言

云计算是一种跨越多组织、多管理域的多机群共享与协同工作的大规模计算[1]。云计算的出现弥补了单一超级计算机不能满足大规模科学计算与工程设计的缺陷。云计算核心技术中包括虚拟化技术,虚拟化技术可以在单台物理机上虚拟出多台虚拟机,提高物理机的资源利用率,该技术常常应用于集群计算、数据中心等场合[2]。虚拟化技术包括经典的 Xen,Xen 将与虚拟化有关的代码集成到虚拟操作系统中,不再重新编译或捕获特权指令,使虚拟操作系统能够发出更有效的指令,凭借高兼容和超稳定的特点,为企业用户提供完善的保障方案。

把云计算环境中的任务通过 Xen 虚拟化技术分配到不同的虚拟机上,提高云计算环境中资源的动态调配和可扩展性,增加服务部署和调度的灵活性,如 IBM 提出的服务。面对云计算中复杂的任务[3],通过在 SaaS 层上软件和策略的设计,可以解决云计算中的一些问题,但不通过底层机制的优化,有些瓶颈问题还是没办法解决,如迁移时间;同时,云计算中服务应用之间的通讯是通过网卡与网卡来实现的,但服务器上的硬件网卡数量有限,不可能给每一台虚拟机加上一块相应的网卡,而是需要将网卡进行虚拟共享,供多台虚拟机同时使用。当虚拟机进行迁移时,网络传送方面的开销所占的比重是非常巨大的,直接影响着这个系统的性能[4],即并行网络任务资源需求的不匹配造成任务低效执行的后果[5]。

本文通过对云计算系统的研究发现,云计算系统中的服务器通过 SaaS 层和 PaaS 层的设计和策略优化能提高服务性能,但没有过多地对 IaaS 层进行研究来更好地解决底层的瓶颈问题。为了解决以上问题,本文研究 Xen 4.2.1[6] 版本的代码,分析 Xen 迁移时的内存算法得出局限性,采用 AR 模型对 Xen 内存迁移算法进行优化,减少内存页面的传递数量,缩短虚拟机迁移时间;页面的减少降低迁移时的网络带宽开销,同时保证了同台服务器上其它虚拟机的网络带宽应用,进而提高了云计算环境下虚拟机的性能,优化了云计算系统。

本文第 2 节介绍相关工作;第 3 节描述虚拟机迁移时内存页拷贝的关键技术;第 4 节介绍基于 AR 模型的内存优化算法;最后给出实验和结果分析,提出下一步的研究工作。

到稿日期:2012-12-23 返修日期:2013-03-01 本文受辽宁省教育科学一般研究项目(L2011004),国家自然科学基金(60903008)资助。

陈廷伟(1976—),男,博士,教授,硕士生导师,CCF 会员,主要研究方向为网格计算、云计算、Web 服务,E-mail:t.w.chen@163.com;张 璞(1987—),男,硕士生,主要研究方向为云计算、虚拟化技术;张忠清(1984—),女,硕士生,主要研究方向为虚拟现实、图像处理。

2 相关工作

随着云计算的快速发展，通过操作系统的虚拟化，提高了云环境中资源的使用率。在集群部署时，通过大规模虚拟机的放置来提高资源利用，这方面有许多研究：有采用带应用服务级目标约束的多目标优化遗传算法[7]，其以对染色体进行编码和译码的方式对解空间内的多个区域同时搜索，以降低服务级目标的违背率，减少虚拟机迁移次数和物理结点的使用数量；有采用自发调节和全局调节之间协作的算法[8]，其处理内存资源充裕和内存资源紧缺状态，优化多虚拟机间的内存虚拟化，提高了服务能力，有较低的性能惩罚。上述策略较好地利用了计算机集群的环境特点，通过大规模集群中虚拟机之间的依赖来优化，实现大型服务器集群的自我维护和管理。然而，它们没有关注虚拟机迁移时的内部实现，没有从虚拟机内部去进行优化和设计。

在云计算部署的同时，云计算环境中有许多开放和成熟的云平台管理系统，它们较多地运行在云计算底层(IaaS)，如GFS(Google File System)是Google推出的，能满足公司快速增长的数据处理，它运行在廉价的硬件设备上，应用于Web-Search；开源的Hadoop可以进行并行运行和实现分布式操作系统，用户若不熟悉底层实现细节来响应用请求，则可自动地将存储数据分割成数据块；弹性计算云(EC2)可以使用户通过网络界面去操作在云平台上运行的实例，它支持对实例关键字的查询，用户根据自己的需要使用计算平台来进行实例付费，使用结束后结束。这些管理平台都是建立在大量廉价的硬件设备之上，通过数据备份和数据分布存储来实现服务，不能很好地利用虚拟化的特点来提高数据的迁移。

同时，云计算集群中使用虚拟化技术增加资源的利用率，虚拟机之间的通信会增加，比如服务器整合、集群运算等，虚拟机之间的通信机制成为瓶颈，尤其在迁移时增加更多的通信量，造成很大的性能开销。在迁移时降低通信量、改善通信机制也是需要亟待解决的问题。

综上所述，云计算集群存在许多有效的策略和算法以及云计算系统，但是没有较多地对IaaS层的虚拟化技术进行研究亦对虚拟机在通讯方面进行优化。为此，提出了基于AR模型的内存优化算法，其相比Xen内存迁移算法缩短了迁移时间，降低了迁移时的网络带宽，提高了虚拟机性能。下面将讨论相关内容。

3 虚拟机迁移时内存页拷贝的关键技术

虚拟机迁移是虚拟化技术中的重要组成部分。

在虚拟机迁移中，最主要和最复杂的部分是内存页拷贝。内存页和影子页表有重要的关系。在虚拟化中，内存虚拟是通过维护影子页表来实现的。Guest OS直接维护虚拟地址VA到机器地址MA的页表，影子页表是对实际页表的一个备份和实时拷贝，如图1所示。Guest OS的实际页表被设为写保护状态，当Guest OS试图修改其页表时，会发生异常。Xen调用相应的异常处理函数，对影子页表的对应表项进行相应的修改，保证影子页表和实际页表的同步，并随实际页表的更新而更新。

Xen对脏页的捕获是通过影子页表进行的。所谓脏页[9]，就是Guest OS中内存发生了变化的页表。

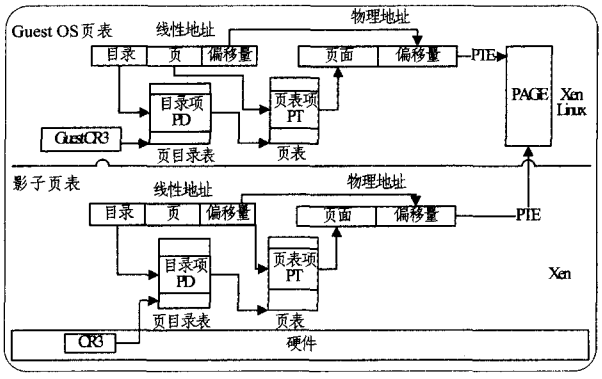


图1 Guest OS页表与影子页表的关系

通过对虚拟机迁移时内存拷贝过程的分析表明，影子页表的表项信息是被动地与Guest OS中的表项进行对应的。根据Xen的设计，当Guest OS内存页发生变化时，需要对Guest OS页表进行修改，这会触发页表错误，将信息传递给Xen，修改影子页表中对应的页表，并将脏页进行记录，形成脏页位图。

虚拟机内存的迁移就是对脏页位图中页表和内存信息的迁移。对虚拟机内存进行迁移需要经过对脏页位图的多次迁移来实现。对Xen源代码进行分析，在tools/libxc/Xc_linux_save.c中存有存储页位图的相关数据表项：to_send，to_skip，to_fix。to_send中存储了由脏页位图和非脏页位图组成的数据结构，记录了所有的页表情况，其产生于前一轮迁移过程中，是这轮迁移需要传送的页位图；to_skip表示在这轮迁移过程中产生的脏页标记到页位图中的页，记录了所有的页情况，是下一轮要传送的to_send；to_fix标记迁移满足终止条件，即最后一轮迁移传送的页面。对to_send和to_skip进行分析，两者在迁移过程中的关系如图2所示。

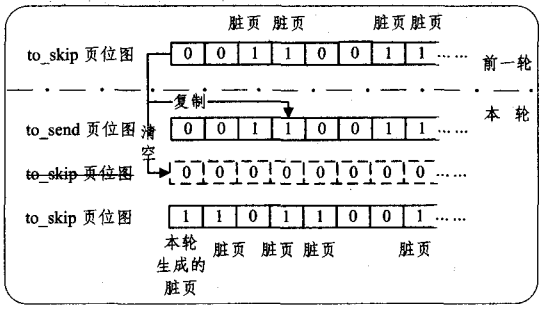


图2 to_send与to_skip的关系

在每轮迭代时，将to_send页位图和本轮动态变化的to_skip页位图进行比较，采用to_send页位图某页为1并且to_skip页位图相应页为0的方式来确定该页是否需要传递。在比较时，选择的是to_skip页位图中某页在迁移时某一刻的状态，并不能判断该页是否是内存修改频繁页，这有可能导致下轮迭代时该页重传，增加内存页传递的页面数，进而增加虚拟机迁移时间和迁移时的服务器网络带宽开销。针对算法局限性，提出基于AR模型的内存优化算法。

4 基于AR模型的内存优化算法

AR模型(Auto-Regressive Model)[10]是一种线性预测，即已知N个数据，可由模型推出第$N+1$个数据，它是研究时间序列的重要方法，它承认事物发展的延续性，运用过去时

间序列的数据进行统计分析就能推测事物的发展趋势。AR模型充分考虑到随机波动的影响，利用历史数据，进行统计分析，并对数据进行适当的处理，进行趋势预测；其优点是简单易行，便于掌握，能够充分运用原时间序列的各项数据，对模型参数有动态确定的能力，精度较好。

在介绍AR模型算法前，首先介绍一下算法的依据：

(1)Xen内存迁移算法与to_skip页位图中某一刻的状态进行对比存在随机性，从而导致页面多次重传；AR模型采用时间间隔预测，较准的预测时间间隔降低了页面多次重传；

(2)AR模型的计算时间虽然比Xen内存迁移算法的时间长，但是传递的页面数量可足够弥补之(本算法主要考虑中等以上内存使用情况下的场景)。

4.1 算法相关数据结构

为了有效地使用AR模型预测内存页的下次修改时间，AR模型算法定义了相应的数据结构，用来保存迁移过程中的内存页信息，包括所有的页面Link_All、更改页面的链表Link_Change、未更改页面的链表Link_NoChange、更改页面在本轮迁移过程中的所有更改时间间隔Link_IntervaLink_Change。

```
typedef struct Link_All{/*所有的页面*/
  struct Link_Change *lc;/*存储更改页面的头指针*/
  struct Link_NoChagnge *lnc;/*存储未更改页面的头指针*/
  int ct;/*本次修改的时间,用来计算时间间隔*/
}
typedef struct Link_Change{/*更改的页面链表*/
  LongInt pageNumber;/*更改的页面编号*/
  Intervale *it;/*更改页面的所有间隔时间*/
  listChangeNode *next;/*指向下一个更改的结点的指针*/
}listChangeNode;
typedef struct Link_NoChange{/*未更改的页面链表*/
  LongInt pageNumber;/*未更改的页面编号*/
listNoChangeNode *next;/*指向下一个未更改的结点的指针*/
}listNoChangeNode;
typedef struct Link_IntervaLink_Change{
  int time;/*两次页面更改的间隔时间*/
  intervaLink_Change * next;/*指向下一次更改的时间*/
}intervaLink_Change;
```

4.2 算法思想和实现

针对Xen 4.2.1代码在内存页拷贝过程中存在的缺陷，对to_send和to_skip页位图的数据结构进行修改，Xc_linux_save.c中采用新建的数据结构：Link_All，Link_Change，Link_NoChange，Link_IntervaLink_Change。

在任何迭代过程中，所有修改和未修改的页信息都存在Link_All链表中。在一轮迭代之前，所有的页面信息存于Link_NoChange中；迭代开始以后，从Link_NoChange中取出修改的页存入Link_Change，在Link_Change中加入更改的页面编号pageNumber、更改的时间ct，第一次更改时，不记入Link_IntervaLink_Change；等大于一次更改后，根据更改的时间和ct的值，计算间隔时间，然后写入与pageNumber对应的Link_IntervaLink_Change链表中，并且同时更新ct为更改的时间。

时间链表Link_Change针对不同的修改页存储的时间间隔链表Link_IntervaLink_Change的结点数量是不一样的；有的页面有可能还没有间隔时间，在一轮迭代过程中，一直存于Link_NoChange链表中。脏页迁移的判断不通过比较to_send和to_skip来实现，改为采用AR模型算法根据本轮迁移中Link_Change链表中相应的pageNumber的相隔时间值Link_IntervaLink_Change链表来预测pageNumber页面下次的间隔时间。

系统中页面变化不同带来Link_Change链表中存储的对应pageNumber的Link_IntervaLink_Change大小是不同的、发生变化的。设某页pageNumber的时间间隔值Link_IntervaLink_Change链表中time构成的时间间隔序列是$\{x_t\}$，$t=1,2,\cdots,N$，N为Link_IntervalLink_Change链表中的time结点和，其构成的AR(n)模型为

$$\begin{cases} x_t=\varphi_1 x_{t-1}+\varphi_2 x_{t-2}+\cdots+\varphi_n x_{t-n}+a_t \\ \phi_n\neq 0 \\ E(a_t)=0, Var(a_t)=\sigma_a^2, E(a_s a_t)=0, s\neq t \\ Ex_s a_t=0, \forall\, s<t \end{cases} \tag{1}$$

通过$AR(n)$模型可计算出x_t的值，即为对应的pageNumber内存页下次更改的时间间隔预测值。计算时，需要按照AR模型中的相关方法对$\phi_1,\phi_2,\cdots,\phi_n,\sigma_a^2$这$n+1$个参数进行估计。因为有

$$a_t=x_t-\phi_1 x_{t-1}-\phi_2 x_{t-2}-\cdots-\phi_n x_{t-n} \tag{2}$$

$$\sigma_a^2=\frac{1}{N-n}\sum_{t=n+1}^{N}(x_t-\sum_{i=1}^{n}\phi_i x_{t-i})^2 \tag{3}$$

所以，一旦估计出ϕ_i，就可以按照式(3)估计出σ_a^2，因此$AR(n)$模型的参数估计重点是对$\phi_1,\phi_2,\cdots,\phi_n(t=1,2,\cdots,n)$这$n$个参数的估计。

参数的估计采用AR模型的常用参数估计方法：最小二乘法，该算法比较简单，参数估计无偏，精度高，可表示为方程组$Y=X\phi+\alpha$，式中，

$$\begin{aligned} Y&=[x_{n+1}\quad x_{n+2}\quad\cdots\quad x_N]^{\mathrm{T}} \\ \phi&=[\phi_1\quad\phi_2\quad\cdots\quad\phi_n]^{\mathrm{T}} \\ a&=[a_{n+1}\quad a_{n+2}\quad\cdots\quad a_N]^{\mathrm{T}} \end{aligned} \tag{4}$$

$$X=\begin{bmatrix} x_n & x_{n-1} & \cdots & x_1 \\ x_{n+1} & x_n & \cdots & x_2 \\ \cdots & \cdots & \cdots & \cdots \\ x_{N-1} & x_{N-2} & \cdots & x_{N-n} \end{bmatrix}$$

若使误差平方和

$$\begin{aligned} Q(\phi)&=Q(\phi_1,\phi_2,\cdots,\phi_n) \\ &=\sum_{i=1}^{N}(x_t-\varphi_1 x_{t-1}-\varphi_2 x_{t-2}-\cdots-\varphi_n x_{t-n})^2 \\ &=\sum_{i=1}^{N}a_i^2 \end{aligned} \tag{5}$$

达到最小，则此时的$\hat{\phi}=(\hat{\phi}_1,\hat{\phi}_2,\cdots,\hat{\phi}_n)^{\mathrm{T}}$成为参数$\phi$的最小二乘(Least Square)估计。则$\phi$的最小二乘法估计为

$$\phi=(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y \tag{6}$$

通过Link_Change链表中相应pageNumber页的Link_IntervaLink_Change建立AR(n)模型，从1到N运行最小二乘法进行计算，得到σ_a^2。因为AR(n)模型随着阶次的升高，σ_a^2的总体趋势是下降的，在n较小时，下降很快，随着n的增加，下降趋势变慢，直到下降为0。将σ_a^2下降到0时的n作为AR(n)的正确阶数，通过最小二乘法计算出模型参数，然后代入AR(n)模型中得到正确率高的x_t值作为pageNumber内存页预测的下次更改时间。

当预测的时间间隔大于固定的阈值时，说明该页不再发

生变化，满足迁移条件，进行迁移。对本轮需要迁移的所有页面进行统计，记入 send_this_iter；然后将 Link_Change 链表置空，所有页重置入 Link_NoChange，进入下一轮迭代后将 send_last_iter 置为 0，把 send_this_iter 拷贝到 send_last_iter 中，send_this_iter 置为 0。

在进入下一轮迭代之前，需要判断拷贝是否结束，结束的判断条件代码如下：

```
if(((send_this_iter>sent_last_iter)
  &&RATE_IS_MAX()) ||
  (iter>=max_iters) ||
  (send_this_iter+skip_this_iter<50))
  last_iter=1;
```

send_this_iter>sent_last_iter&&RATE_IS_MAX()表示本轮发送总页数大于上一轮发送页数，并且网络负担达到最大；iter>=max_iters 表示迭代次数达到最大的迭代次数；end_this_iter+skip_this_iter<50 表示本轮发送的内存页数和跳过的内存页数总和小于 50。达到这些条件之一时，last_iter 设置为 1，即进入最后一轮迭代；否则，进行下次迭代。

5 实验和结果分析

5.1 实验设置与环境

本节将对 Xen 内存迁移算法与基于 AR 模型的内存优化算法(以下简称 AR 算法)进行对比实验。实验中使用 Whetstone 基准测试程序(WP)，Python 矩阵计算(PM)，FTP 服务器上传、下载(FS)等多种类型的应用程序作为不同虚拟机运行的任务(满足普遍性)，在不同内存下进行迁移，然后依据迁移时间(MT)、宕机时间(DT)、网络带宽(NB)3 个指标进行评价对比。在相同应用和相同内存的情况下，每个实验运行 20 次，统计 20 次实验结果各指标的平均值。

实验结点是 15 台 CentOS 5.8 的 Linux 操作系统，CPU 是 Intel Xeon E7520 1.866GHz；内存是 16G；硬盘为 4TB；网络连接是 1000Mbps 以太网。10 个结点内核上需要安装和编译 Xen 4.2.1 源码，在这 10 个结点上可以同时运行多台虚拟机；另外 5 个结点不需要安装和编译 Xen 4.2.1 源码，而是将其作为 NFS 服务器来存储虚拟机的 img 镜像。虚拟机 img 镜像采用的是 CentOS 5.6 的 Linux 系统，其内存大小在启动时可动态调整。

5.2 实验结果和分析

Xen 内存迁移算法和 AR 算法在 MT、DT、NB 这 3 个指标上进行实验，得到的对比结果数据如图 3－图 5 所示。

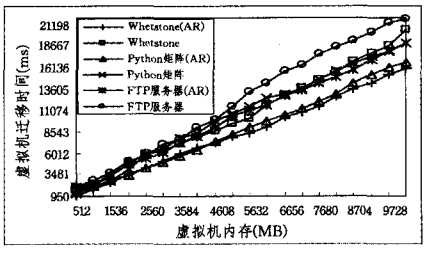


图 3 Xen 内存迁移算法和基于 AR 模型算法的 MT 指标对比

如图 3 所示，随着内存大小从 0.5G 增加到 10G，两种算法的 MT 评价指标在 WP、PM、FS 应用上都呈现增加的趋势，但是 AR 算法在 MT 指标方面一直优于 Xen 内存迁移算法，这通过同种应用的不同算法得到的数据分布差可以看到。因为 AR 算法的 FS 内存使用量大，所以其在应用上表现明显。在内存是 4G 时，AR 算法的 MT 指标比 Xen 内存迁移算法低 7.38%；在内存是 8G 时，低 8.35%；在内存是 10G 时，低 9.21%。这说明 AR 算法较 Xen 内存迁移算法在迁移时取得了较好的结果。

如图 4 所示，随着内存大小的增加，两种算法的 DT 评价指标在 WP、PM、FS 应用上也都呈现增加的趋势，并且 AR 算法在 DT 指标方面一直优于 Xen 内存迁移算法，内存达到 4G 以后表现明显，这通过同种应用下不同算法得到的数据分布差可以分析出来。在内存是 4G 时，AR 算法的 DT 指标比 Xen 内存迁移算法低 8.58%；在内存是 8G 时，低 8.71%；在内存是 10G 时，低 8.69%。这说明 AR 算法较 Xen 内存迁移算法在迁移时取得了较好的结果。

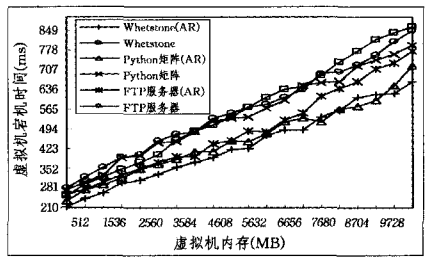


图 4 Xen 内存迁移算法和基于 AR 模型算法的 DT 指标对比

如图 5 所示，记录虚拟机迁移前后和迁移时的网络带宽，网络带宽采用 bing 算法进行估测[11]，两种算法在 NB 指标上都呈现上下波动的趋势，但整体趋势为中间高两边低，上下波动是由虚拟机运行应用中的小任务的开始和结束导致的。在数据分布中，FS 的数据分布明显高于 WP 和 PM，因为 FS 的上传下载占用较大的网络带宽；WP 高于 PM 是因为 PM 基本上不需要使用网络的信息传输。10s 前和 30s 后为非迁移时间，10s 到 30s 为虚拟机迁移时间。在非迁移时间段，两种算法的 WP、PM、FS 3 种应用各自相互交错波动，NB 指标值很接近，因为不需要考虑两种算法；在迁移时间段，AR 算法在 NB 指标方面就优于 Xen 内存迁移算法，网络带宽小于 Xen 内存迁移算法的。在迁移时间段的波动比较明显，这是因为内存迁移过程中的多次迭代和单次迭代后的短暂停留。

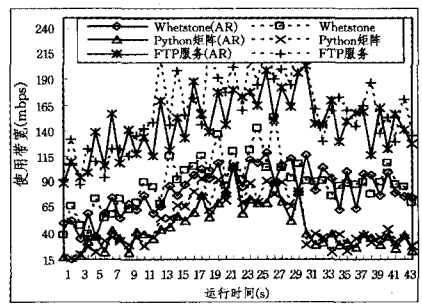


图 5 Xen 内存迁移算法和基于 AR 模型算法的 NB 对比

综上所述，虽然两种算法在非迁移时间 NB 指标值很接近，但是 AR 算法在 MT、DT、除非迁移时间以外的 NB 这 3 个指标上均好于 Xen 内存迁移算法，并且通过 3 个指标的对比，AR 算法取得很大的优势。这说明 AR 算法能够有效地缩短虚拟机的迁移时间，降低迁移时的网络带宽。

结束语 本文选择成熟的 Xen 并通过对 Xen 4.2.1[9] 版本的代码进行研究，详细分析 Xen 迁移时内存算法的局限

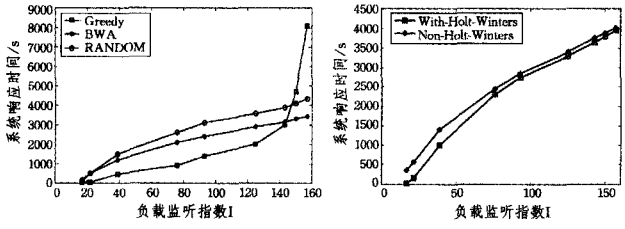
(下转第 88 页)

务的长度相同),统计在不同 I 值情况下系统的响应时间。

b)以 a)中生成的任务请求数为依据,以周为一个时间周期,利用本文提出的 Holt-Winters 季节指数平滑模型对下一周的任务请求数做出预测,并提前为每天预先部署相应数量的虚拟机。记录并统计不同任务请求强度下系统的响应时间。

4)实验结论

实验结果如图 4 所示,在图 4(a)中,对 3 种算法在不同负载监听指数强度下的系统响应时间取 9 个点进行描述。本文可以得出结论,即在没有暴发式任务请求的环境下,Greedy 算法有着更佳的表现,但是随着系统瞬时任务请求量的不断增加,本文提出的 BWA 模型可有效地降低暴发式负载对于系统响应时间的影响。通过观察图 4(b)本文可以得出以下结论:Holt-Winters 模型可有效预测系统下一个时间周期的任务请求数量,在低负载率(非暴发式任务请求)的环境下,可更好地加快云计算系统的资源部署过程。



(a) 3 种算法在不同负载监听指数强度下的系统响应时间

(b) 添加跟踪预测算法前后系统响应时间比较

图 4 BWA 模型对系统响应速率的影响

结束语 本文针对云计算系统会遇到瞬时大量任务请求的问题,设计了 BWA 模型以应对暴发式的任务请求。其中主要的工作从以下两点展开:1)设计了负载监听指数 I,用以实时监控云计算系统任务请求变化量,并通过监测器算法动态地判断暴发式任务请求的始末;2)增加了 Holt-Winters 季节指数平滑模型,以预测云计算系统小规模的符合季节性变化规律的任务请求量,从而加快云计算系统的响应速度。最后通过实验仿真证明本文提出的模型可以有效地提高云计算系统在暴发式任务请求量下的响应速度,提高用户体验。

参 考 文 献

[1] 谭一鸣,曾国荪,王伟. 随机任务在云计算平台能耗的优化管理方法[J]. 软件学报,2012,23(2):266-278

[2] Cui V,Zhou T,Chen J,et al. 中国云计算发展之道 [R]. 北京:IDC 中国,2010

[3] 杨星,马自堂,孙磊. 云环境下基于改进蚁群算法的虚拟机批量部署研究[J]. 计算机科学,2012,39(9):33-37

[4] Tai Jiang-zhe,Meleis W,Zhang Jue-min,et al. ARA:Adaptive Resource Allocation for Cloud Computing Environments under Bursty Workloads[R]. 978-1-4673. Northeastern University,Boston,USA,2011

[5] 罗军舟,金嘉晖,宋爱波,等. 云计算:体系架构与关键技术[J]. 通信学报,2009,32(7):1337-1348

[6] 杨际祥,谭国真,王荣生. 并行与分布式计算动态负载均衡策略综述[J]. 电子学报,2010,38(5):1122-1130

[7] Gusella R. Characterizing the Variability of Arrival Processes with Indices of Dispersion[R]. TR-90-051. International Computer Science Institute,USA,1990

[8] Fang Mi-ning, Giuliano C, Cherkasova L, et al. Burstiness in Multi-Tier Applications: Symptoms, Causes, and New Models, CNS-0720699[R]. CCF-0811417. College of William and Mary, Williamsburg,USA,2008

[9] Lassnig M,Fahringer T. Identification,modeling and prediction of non-periodic bursts in workloads[C]// 2010 10th IEEE/ACM International Conference on Cluster,Cloud and Grid Computing. Austria,IEEE DOI /CCGRID,2010:485-494

[10] Goodwin P. The Holt-Winters Approach to Exponential Smoothing:50 years Old and Going Strong[J]. FORESIGHT,2010,Fall:30-33

[11] Prajakta S K. Time series Forecasting using Holt-Winters Exponential Smoothing[D]. Kanwal Rekhi School of Information Technology,2004

[12] Rodrigo N C,Ranjan R,Beloglazov A,et al. CloudSim:A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms[R]. Cloud Computing and Distributed Systems (CLOUDS) Laboratory,Australia,2010

(上接第 67 页)

性,采用 AR 模型优化内存算法,减少内存页面的传递数量,缩短虚拟机迁移时间,降低迁移时的网络带宽开销,保证了同台服务器上其它虚拟机的网络带宽应用,提高了云计算环境下虚拟机的性能,优化了云计算系统。如果考虑 AR 模型的普遍性,有些场景是不能胜任的,比如低内存服务环境下的云计算,因为 AR 模型的计算时间反而提高迁移时间。下一步工作就是研究低内存场景。

参 考 文 献

[1] 文明波,丁治明. 适用于云计算的面向查询数据库数据分布策略[J]. 计算机科学,2010,37(9)

[2] Li Bo,Li Jian-xin,Huai Jin-peng,et al. Enacloud:An energy-saving application live placement approach for cloud computing environments[C]//Proceedings of the International Conference on Cloud Computing. Bangalore,2009:17-24

[3] 陈廷伟,周山杰,秦明达. 面向云计算的任务分类方法[J]. 计算机应用,2012,32(10):2719-2723,2727

[4] Menon A,Santos R J,Turner Y,et al. Diagnosing Performance Overheads in the Xen Virtual Machine Environment[C]// Proc of the 1st ACM/USENIX International Conference on Virtual Execution Environments. Chicago,USA:[s. n.],2005:13-23

[5] 刘伯成,陈庆奎. 云计算中的集群资源模糊聚类划分模型[J]. 计算机科学,2011,34(12)

[6] Xen [EB/OL]. http://xen. org

[7] 李强,郝沁汾,肖利民,等. 云计算中虚拟机放置的自适应管理与多目标优化[J]. 计算机学报,2011,34(12)

[8] 张伟哲,张宏莉,张迪,等. 云计算平台中多虚拟机内存协同优化策略研究[J]. 计算机学报,2011,34(12)

[9] Han H,Jung H,Kang S,et al. Performance evaluation of a remote memory system with commodity hardware for large-memory data processing[J]. Cluster Computing,2011,14(4):325-344

[10] Yan Li-ren,Huang Wei. An IC yield enhancement approach by ARMA modeling and dynamic process control[J]. The International Journal of Advanced Manufacturing Technology,2009,42(7/8):749-756

[11] 张文杰,钱德沛,栾钟治,等. bing 算法估测网络带宽的研究与实现[J]. 小型微型计算机系统,2004,25(5)