一种基于准同步检查点的虚拟机卷回恢复算法

张 展 左德承 黄友富 何 辉
(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

摘 要 针对典型的云平台下虚拟化系统的特点,提出了一种结合选择性日志的准同步检查点算法VM_QSC:保持不同虚拟机节点固有的优化检查点周期,通过物理节点Hypervisor选择性地进行虚拟机的消息日志的稳定存储,在全局监控节点维护虚拟机一致线信息,保持全局的一致性。与传统的准同步检查点和同步检查点相比,该算法维持了虚拟机检查点设置的自主性,并显著降低了虚拟化系统的容错开销,可以有效应用于云计算环境下的虚拟资源管理和动态迁移。

关键词 准同步检查点,选择性日志,卷回恢复,虚拟机,云计算
中图法分类号 TP302.8 **文献标识码** A

Rollback Recovery Algorithm for Virtual Machine Based on Quasi-synchronous Checkpointing

ZHANG Zhan ZUO De-cheng HUANG You-fu HE Hui
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

Abstract Considering the characteristic of the virtual machines based on cloud platform, a Quasi-synchronous checkpointing with selective message logging algorithm for virtual machine (for short, VM_QSC) was presented. The algorithm keeps the inherent optimized checkpoint interval of the VM nodes, selectively stores stable optimistic message log, and maintains the consistency of the whole VM systems by hypervisor on the physical machines. Performance evaluation and simulation result show that in contrast with the traditional communication induced checkpointing and coordinated checkpointing, VM_QSC maintains the autonomy checkpointing, and saves communication and storage cost. It adapts for the cloud platform to manage the virtual resource and migrate the virtual machine.

Keywords Quasi-synchronous checkpointing, Selective message logging, Rollback recovery, Virtual machine, Cloud computing

1 引言

虚拟化技术的提出为实现系统资源优化和体系结构透明化提供了优化处理环境,通过虚拟化技术,动态组织多种计算资源,隔离具体的硬件体系结构和软件系统之间的紧密依赖关系,可实现透明的可伸缩计算系统结构。在传统分布式系统中,检查点及日志技术作为软件容错和进程迁移的最主要手段之一,一直以来因对系统资源的高开销而无法广泛应用。在虚拟化系统中,一方面,对系统资源的优化管理为检查点降低开销提供了可能;另一方面,提供操作系统资源的全局视图为系统全局一致性的监控提供了良好的平台。检查点作为容错技术和进程迁移技术的重要策略和手段,在虚拟化系统中的应用正在得到越来越多的重视。

现有虚拟化系统中普遍采用协同检查点进行状态保存,而在大规模的虚拟化分布式系统中,在虚拟机层次上,采用协同检查点[1](Coordinated checkpointing)的方式进行检查点设置将导致如下问题:

(1)同步开销大:协同的检查点协议在消息同步阶段,网络处于阻塞状态,保证虚拟机处理完所有的协同消息来达到同步状态。这期间分布式系统是不可用的,而且对大规模的分布式系统,同步开销会难以容忍。

(2)检查点时间开销:由于虚拟机文件较大,检查点操作的时间较长,检查点期间,分布式系统也是不可用的,导致整个系统的可用性降低。

(3)存储开销大且存在存储竞争:由于虚拟机的检查点是整个虚拟机的镜像,因此对存储空间的需求较大。同时现在大部分分布式系统采用网络存储,在进行同步检查点设置时,对网络存储的竞争也是非常重要的问题。

而在大规模的虚拟化计算环境中,由于计算任务、通信负载、可靠性等影响检查点最优间隔[1]的因素差异较大,不同的虚拟机以不同的间隔自主地作为基本检查点是一种比较优化合理的方式,因此,研究虚拟化环境下准同步的检查点算法在大规模的分布式系统中具有一定的意义。准同步检查点允许进程独立地进行检查点设置,同时通过消息的依赖关系确保形成全局一致的检查点。但传统的准同步检查点通过增加强迫检查点来保证系统的一致性,破坏了结点固有的检查点周

到稿日期:2013-09-17 返修日期:2013-11-21 本文受国家"八六三"高技术研究发展计划重大项目课题(2013AA01A205),国家自然科学基金青年项目(61003047),中央高校基本科研业务费专项资金(HIT. NSRIF. 2014)资助。

张 展(1978—),男,博士,副教授,主要研究方向为容错计算,E-mail:zz@ftcl.hit.edu.cn;左德承(1972—),男,博士,教授,博士生导师,主要研究方向为容错计算;黄友富(1989—),男,硕士生,主要研究方向为容错计算;何 辉(1978—),男,博士生,主要研究方向为容错计算。

期，增加了额外开销，而由于虚拟化可以提供操作系统资源的全局视图，借助细粒度的日志机制，可以在保持结点固有检查点周期的同时，降低系统容错开销。

本文提出一种结合选择性日志的准同步检查点算法，通过选择性地在 hypervisor 上进行日志记录，来保持虚拟机结点固有的检查点周期，降低容错开销。同时由于监控节点可以掌握全局的信息，因此根据系统一致性依赖跟踪关系和检查点设置情况，有选择地将影响系统一致性的消息记录到稳定存储，这种选择性日志是一种在保持一致性基础上减少开销的有效手段。这样，将容错手段划分为检查点、暂存日志、选择性稳定存储日志 3 个层次，有效降低虚拟化系统的容错开销。

2 相关研究工作

在虚拟化系统中以虚拟机作为计算单元，为了实现虚拟机的迁移以及恢复，需要定期或不定期地进行 GuestOS 的状态保存。在通常的检查点设置算法中，以较高自主性设置检查点的代价往往是卷回距离难以控制。如何做到两者的兼顾就成为一个至关重要的问题。在虚拟化系统中普遍采用协同检查点进行状态保存。文献[2]提供基于 OpenVZ 的虚拟机检查点设置。OpenVZ 是操作系统级虚拟化，每个 Guest OS 被抽象为 Virtual Private Server(VPS)。借助具有检查点功能的 LAM，允许 VPS 进行检查点设置。通过守护进程 Ovzd 则可将检查点复制到多个节点来提高系统整体的可靠性。文献[3]的 Virtual Cluster CheckPointing(VCCP)通过在 Hypervisor 中嵌入检查点功能来实现虚拟机的检查点设置，试图降低上层应用以及客户操作系统容错处理的负担。在一个虚拟集群中，检查点设置由一个作为头节点的虚拟机发起，头节点可协同所有作为计算节点的虚拟机完成检查点设置。文献[4]指出通过 VIOLIN 的虚拟网络可以将一个主机上的多个虚拟机组织在一个虚拟域中(这里的虚拟机平台基于 XEN)，在检查点设置过程中以各个虚拟域为单位，同样采取了协同方式的检查点设置。文献[5,6] 通过 InfiniBand 网络连接基于 Xen 的虚拟主机，通过增强的 ARMCI(Aggregate Remote Memory Copy Interface)在虚拟集群内实现协同的检查点设置。

准同步检查点又称为通信引发的检查点，分为基于模型的准同步检查点和基于索引的准同步检查点。基于模型的准同步检查点通过防止特定的通信及检查点模式的产生，来避免进程之间出现不一致的状态，包括 MRS 协议[7]、BQC 协议[8]和 RDT 协议[9]等。基于索引的检查点算法来源于 Lamport 逻辑时钟的思想，即用一个时间戳(整数索引)来表示进程作检查点时的逻辑时钟，具有相同索引的检查点可以形成全局一致性检查点[10,11]。文献[12]首次比较完整地阐述了基于索引的检查点算法的思想，提出了完整的 QSA(Quasi-Synchronous algorithm)协议，包括准同步检查点算法、回卷算法、垃圾收集以及中途消息的处理。在文献[11,13]中，采用了 Lazy-Index 和 Aftersend 等优化方法。

3 系统模型和基本定义

3.1 系统模型

我们考察一种典型的云平台架构，图 1 为 CloudStack 云平台构建的一种模型，即考虑到单物理机多虚拟机的情况，一台物理主机上构建多个 GuestOS 虚拟机系统(图中的 VM 实例)，物理服务器由统一的 Hypervisor 负责多虚拟机的管理，维护多个虚拟机的消息转发，检查点设置由 Hypervisor 控制，整个系统由 Manager Server Cluster 作为中央控制平台，从整体上掌握虚拟化平台中系统资源的使用情况，并负责系统一致性的维护。虚拟机的检查点镜像信息和必要的消息日志信息通过 Manager Server 的统一管理，定时存储到系统的稳定存储中，这样，通过系统级的 Manager Server、物理机级的 Hypervisor 和虚拟机 GuestOS 三层管理维护系统的容错策略，通过多层次的视图维护全局一致检查点的方法可以减少不必要的强制检查点和消息日志，显著降低容错开销。

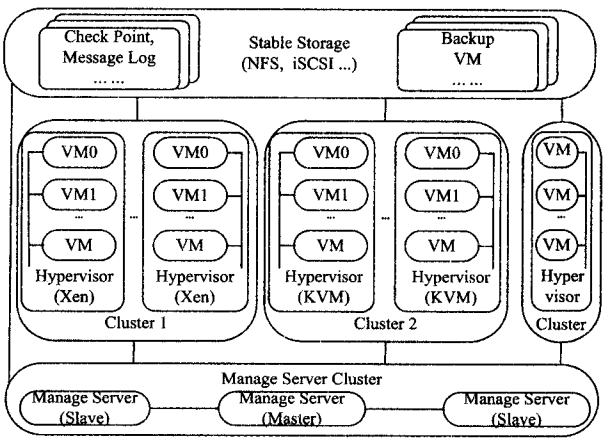

图 1 虚拟机平台架构

3.2 相关定义

在虚拟化环境下的分布式计算由在虚拟机上的并发进程组成，假设一台虚拟机上执行一个进程。计算模式是事件驱动的，消息的发送和接收均被称为事件。H 是在分布式计算中事件的集合，计算模式可以由 $\hat{H}=(H,\xrightarrow{hb})$ 表示，其中 $\xrightarrow{hb}$ 表示 Lamport 所定义的先于(happen before)[14]关系。系统的计算模式满足分段确定性模型[14](Piecewise Deterministic，PWD)：任何一个进程的执行都可以看作是一组状态间隔(State Interval)的集合，由不确定事件计算产生进程的状态转换。每个进程周期性存储局部检查点。每一个检查点被分配唯一的序号。进程 P_i 的第 x 个检查点被分配序号 x，并且表示为 $C_{i,x}$，对应于局部状态 $\sigma_{i,s}$($x\leqslant s$)。$C_{i,x}$ 是由 $C_{i,x-1}$ 与 $C_{i,x}$ 之间的事件序列作用在 $C_{i,x-1}$ 对应的状态而产生的。

分布式系统检查点中的一致性系统状态(consistent system state)[1]是指对于每个被接收到的消息，其发送进程的状态表明该消息已经被发出了。

定义 1 一个全局检查点 $GC=\{C_{0,m_0},C_{1,m_1},C_{2,m_2},\cdots,C_{n,m_n}\}$ 称为一个全局一致性检查点仅当对任意消息 m，有：$receive(m)\in C_{i,m_i}\rightarrow send(m)\in C_{j,m_j}$，$C_{i,m_i}\in GC$，$C_{j,m_j}\in GC$，其中，$0\leqslant i\leqslant n-1$，$0\leqslant j\leqslant n-1$。

全局一致性状态就是保证没有孤儿消息产生，孤儿消息对应于发送者还没有发出相应的消息而接收者就已经接到该消息，这在实际情况中是不可能发生的。任何卷回恢复协议的基本目标就是当系统因失效而导致系统状态不一致时，使系统返回到一个一致性状态。一致的全局检查点也称为恢复线[1](recovery line)，基于检查点的卷回恢复协议首先应确保在发生错误时系统能确定恢复线。

4 结合选择性消息日志的准同步检查点算法

根据图 1 所示云平台架构下虚拟化环境的特点，设计了

结合选择性日志的准同步检查点算法 VM _QSC，不同类型的 VM 按照自身检查点周期的最优值设置检查点，由物理机上的 Hypervisor 层暂存各 GuestOS 的检查点，并延迟存储到稳定存储(选择网络负载低的时机)，同时暂存转发的消息日志，保证在某个 VM 失效时可以在不影响其它 VM 的情况下实现一致的卷回，同时选择性地在稳定存储中记录部分消息日志，保证在物理机失效的情况下，该节点的多个虚拟机可以实现一致的卷回恢复。在云平台的 Manager 节点维护整个系统的全局一致信息，监控系统状态。

4.1 基于时钟的准同步检查点周期设置

将虚拟机的检查点间隔根据节点通信频繁程度不同设定为两种：基准周期型节点和多周期节点，对于基准周期型节点最小检查点间隔，在检查点计时器溢出时，同步设置检查点；而多周期节点检查点周期长，限定该类型节点的检查点周期为基准周期的整数倍。将检查点间隔为基准检查点的节点定义为 SI 型(Single Interval)节点，将检查点间隔为多周期的节点定义为 MI_i(Multi Interval)型节点，下标 i 表示基准检查点间隔的周期数。

VM_QSC 采用基于时钟的准同步检查点：HV 和虚拟机节点 S 按照基准时间设置检查点，SI 型节点设计基本检查点，MI_i 型节点按照基准时间的 i 倍设置检查点。在 HV 上维护各节点的一致时间线，HV 对各节点基准间隔进行计数，作为当前索引的参考值，使索引保持同步前进。当到达检查点设置基准设置时间点时，在所有需要设置检查点的进程完成检查点设置之前，系统的一致恢复线不变；当 HV 检测到所有需要设置检查点的虚拟机节点完成检查点设置时，更新一致线，在该位置设置检查点的进程以最新的检查点设置一致线，没有设置检查点的进程仍按照原检查点设置一致线。

如图 2 所示，VM_1 到 VM_4 为虚拟机 GuestOS 实例，VM_2，VM_3 为 SI 型节点，VM_1 为 MI_2 型节点，VM_4 为 MI_3 型节点。T_0 为检查点设置基准时间，则在 T_0 附近，当 VM_2，VM_3 均完成检查点设置后，各节点一致线为($C_{1,0}$，$C_{2,1}$，$C_{3,1}$，$C_{4,0}$)，在 $2T_0$ 附近，由于时钟的偏移，消息 m 在 VM_2 设置完成检查点 $C_{2,2}$ 后发出，而到达 VM_3 时 $C_{3,2}$ 还没有设置，则 VM_3 将该检查点提前设置后接收 m，保持一致性。在该位置，当所有 2 号检查点设置完成前，系统维护的一致线仍为($C_{1,0}$，$C_{2,1}$，$C_{3,1}$，$C_{4,0}$)，在 manager 监测到检查点 $C_{1,2}$，$C_{2,2}$，$C_{3,2}$ 设置完后，一致线前移到($C_{1,2}$，$C_{2,2}$，$C_{3,2}$，$C_{4,0}$)，当某个虚拟机 GuestOS 失效时，系统由物理机 Hypervisor 暂存的检查点和日志信息完成虚拟机恢复，如果某个物理机节点失效，为保持系统一致性，各相关节点将卷回到一致线。

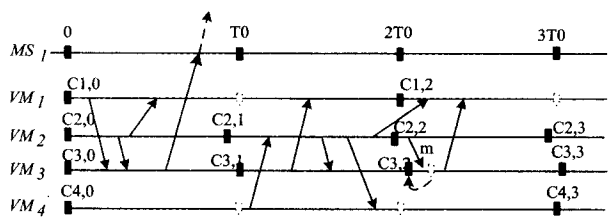

图 2 准同步检查点

由于系统中不存在公共时钟，各个节点的时钟存在着时钟差异，时钟漂移远小于检查点周期。时钟漂移造成的检查点的不一致可以通过下文的选择性消息日志方法予以解决。

4.2 选择性消息日志算法

仅有周期性的准同步检查点策略，并不能保证系统的一致性，需要结合日志的方法消除由于某些虚拟机节点失效所产生的孤儿消息，物理节点的 HV 在内存暂存节点的接收消息日志，同时采用选择性消息日志在稳定存储中记录 MI 型节点的部分接收消息，而 SI 型节点不需要任何消息日志的稳定存储，这样在维护各节点检查点的固有周期的同时减少了稳定消息日志的数据量。

如图 3 所示，其假设与图 2 相同。在 t_{T0} 时刻，VM_2 和 VM_3 设置检查点，则在 HV 中暂存的这两个节点的区内接收消息日志可以删除，而由于 VM_1、VM_4 在 t_{T0} 不进行检查点设置，因此在 0 到 T_0 周期的接收消息日志需要在 t_{T0} 时刻记录到稳定存储中，如图上消息 m_3；对于跨区消息 m_4，需要在发送方的 HV_1 和接收方的 HV_2 上同时保存消息的暂存日志，保证在某一小区 HV 失效时该消息可以重现。在 t_{2T0} 时刻，由于 VM_1、VM_2、VM_3 均要设置检查点，因此这 3 个节点之前的暂存接收消息日志均不用保留，但要回收保存在稳定存储的 VM_1 的消息日志中，如图上 m_5 不保存到稳定存储；而 VM_4 节点仍需要在 t_{2T0} 保存其暂存日志到 HV 的稳定存储，如 m_6 需要保存到稳定存储。在 t_{2T0} 到 t_{3T0} 周期，VM_1 接收到 VM_2 由前一周期发送的消息 m_7，需要保存到稳定存储；而由于时钟漂移，VM_2 发出的 $CSN_2=3$ 的消息 m_8 到达 VM_3 时，VM_3 还没有设置 $CSN_3=3$，则 VM_3 为保证一致性，提前设置 $CSN_3=3$ 的检查点后接收消息 m_8。

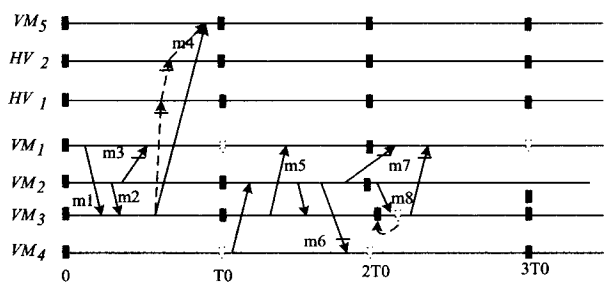

图 3 选择性日志

4.2.1 卷回恢复算法

通过在 Manager Server 采用多模的容错机制，可以认为 MS 在虚拟化环境中是可靠的，因此 MS 的失效和卷回恢复不在此讨论。

当只有虚拟机 VM 失效时，恢复所需要的消息日志能够从其所属物理节点的 HV 得到，卷回恢复算法如下：当 HV 收到 VM_i 卷回恢复的请求时，将 VM_i 存储在稳定存储中的检查点传递到 VM_i 作为卷回恢复的起点，并将存储在 HV 内存中的接收消息按照失效前接收的消息按顺序重新发给该进程，使 VM_i 完成一致的卷回恢复。

由于一致恢复线信息和消息的确定因子在控制台和稳定存储中，物理节点的失效将通过状态重构完成，各虚拟机可实现一致的卷回恢复；由于通过控制台节点和稳定存储对一致恢复信息的管理，孤儿消息不会产生，系统可以保持一致性。

5 性能评价

对 VM_QSC 算法进行性能评价，讨论选择性记录在稳定存储中的消息日志占转发消息的比例。对系统进行简化[15]，设定由 SI 型和 MI_2 型两种类型节点组成的系统中，SI 型节点的检查点间隔为 T_0，MI_2 型节点的检查点间隔为 $2T_0$，做如下假设：

(1)SI 型节点向 SI 型节点发送消息符合速率为 $\lambda_{\alpha 1}$ 的泊松分布。

(2)SI 型节点向 MI_2 型节点发送消息符合速率为 $\lambda_{\alpha 2}$ 的

泊松分布。

(3)MI_2 型节点向 SI 型节点发送消息符合速率为 $\lambda_{\beta 1}$ 的泊松分布。

(4)MI_2 型节点向 MI_2 型节点发送消息符合速率为 $\lambda_{\beta 2}$ 的泊松分布。

(5)不考虑消息发送的延迟,即接收者总能无延迟地收到发送者发送的消息。

(6)假设两种类型节点数量之比为 1∶1。

为了计算需要记录在稳定存储中的消息日志占所有传递消息的比例,首先分别计算在一个检查点间隔时间段内,两类进程所接收到的消息数的平均值。

SI 型节点接收到来自 SI 型节点的消息符合速率为 $\lambda_{\alpha 1}$ 的泊松分布,即:

$P\{N_{\alpha 1}(t)=m\}=\frac{(\lambda_{\alpha 1}t)^m}{m!}e^{-\lambda_{\alpha 1}t}$,其中 $t>0$,所以在一个检查点周期 T_0 中,SI 型节点接收到来自 SI 型节点的消息数目的平均值为:

$$\overline{N_{\alpha 1}}=E(N_{\alpha 1}(T_0))=\sum_{m=0}^{\infty}m\frac{(\lambda_{\alpha 1}T_0)^m}{m!}e^{\lambda_{\alpha 1}T_0}$$
$$=\lambda_{\alpha 1}T_0e^{-\lambda_{\alpha 1}T_0}\times e^{\lambda_{\alpha 1}T_0}=\lambda_{\alpha 1}T_0$$

MI_2 型节点接收到来自 SI 型节点的消息符合速率为 $\lambda_{\alpha 2}$ 的泊松分布,在 MI_2 型的一个检查点周期 $2T_0$ 中,$\overline{N_{\alpha 2}}=2\lambda_{\alpha 2}T_0$,同理,$\overline{N_{\beta 2}}=2\lambda_{\beta 2}T_0$,$\overline{N_{\beta 1}}=\lambda_{\beta 1}T_0$。

计算 SI 型节点在 T_0 周期所接收的消息总数,根据泊松过程的性质,$N_{\alpha 1}(t)$、$N_{\beta 1}(t)$ 为互相独立且强度分别为 $\lambda_{\alpha 1}$、$\lambda_{\beta 1}$ 的泊松过程,则其叠加 $N_{P_{T0}}(t)=N_{\alpha 1}(t)+N_{\beta 1}(t)$ 仍为泊松过程,并且

$$P\{Np_{T0}(t)=m\}=P\{N_{\alpha 1}(t)+N_{\beta 1}(t)=m\}$$
$$=\sum_{k=0}^{m}P\{N_{\alpha 1}(t)=k+N_{\beta 1}(t)=m-k\}$$
$$=\sum_{k=0}^{m}P(N_{\alpha 1}(t)=k)P(N_{\beta 1}(t)=m-k)$$
$$=\sum_{k=0}^{m}\frac{[\lambda_{\alpha 1}(t)]^k e^{-\lambda_{\alpha 1}t}}{k!}\cdot\frac{[\lambda_{\beta 1}(t)]^{m-k}e^{-\lambda_{\beta 1}t}}{(m-k)!}$$
$$=e^{-(\lambda_{\alpha 1}+\lambda_{\beta 1})t}\sum_{k=0}^{m}\frac{m!}{k!(m-k)!}\cdot\lambda_{\alpha 1}^k\cdot\lambda_{\beta 1}^{m-k}\cdot\frac{1}{m!}$$
$$=\frac{[(\lambda_{\alpha 1}+\lambda_{\beta 1})t]^m}{m!}e^{-(\lambda_{\alpha 1}+\lambda_{\beta 1})t},m=0,1,2,\cdots$$

即 $N_{P_{T0}}(t)$ 满足强度为 $\lambda_{\alpha 1}+\lambda_{\beta 1}$ 的泊松过程,可得,在检查点周期 T_0 中,SI 型节点接收的平均消息数 $\overline{N_{P_{T0}}(T_0)}=\overline{N_{\alpha 1}(T_0)}+\overline{N_{\beta 1}(T_0)}=(\lambda_{\alpha 1}+\lambda_{\beta 1})T_0$,同理,$MI_2$ 型节点在其检查点周期 $2T_0$ 中接收的平均消息数 $\overline{N_{P_{2T0}}(2T_0)}=\overline{N_{\alpha 2}(2T_0)}+\overline{N_{\beta 2}(2T_0)}=2(\lambda_{\alpha 2}+\lambda_{\beta 2})T_0$,需要记录到稳定存储的消息的平均数为 MI_2 型节点接收消息的一半,即需要稳定存储的消息占总消息的比例为:

$$\eta=\frac{(\lambda_{\alpha 2}+\lambda_{\beta 2})T_0}{2(\lambda_{\alpha 1}+\lambda_{\beta 1})T_0+2(\lambda_{\alpha 2}+\lambda_{\beta 2})T_0}$$
$$=\frac{(\lambda_{\alpha 2}+\lambda_{\beta 2})}{2(\lambda_{\alpha 1}+\lambda_{\beta 1}+\lambda_{\alpha 2}+\lambda_{\beta 2})}$$

由于 MI_2 型节点消息传递较少,$\lambda_{\alpha 2}$,$\lambda_{\beta 2}$ 均为较小的值,例如:$\lambda_{\alpha 1}/\lambda_{\alpha 2}=2∶1$,$\lambda_{\beta 1}/\lambda_{\beta 2}=2∶1$,$\lambda_{\alpha 2}/\lambda_{\beta 2}=2∶1$,因此可以得到 $\eta=1/6$,即仅仅 1/6 的消息需要存储到稳定存储中。可见,选择性记录在稳定存储中的消息日志占总消息的比例很低,算法能够保证系统的低开销。

结束语 本文针对云平台下虚拟化系统的特点,提出了面向虚拟机的结合选择性日志的准同步检查点及卷回恢复算法 VM_QSC,其保持 VM 按照固有周期设置检查点,通过在 HV 上记录转发的 VM 间消息暂存日志和选择性的稳定存储日志保证系统的一致性。通过检查点、暂存日志、选择性稳定存储日志 3 个层次增强 VM 设置检查点的自主性,同时容错开销也显著降低。

下一步的工作是结合云计算环境下全局视图的特点,优化 VM_QSC,在保证一致性的基础上,进一步减少容错算法和虚拟机动态迁移的开销,同时构建小型云平台,在真实环境中测试该算法的性能。

参 考 文 献

[1] Elnozahy E N, Alvisi L, Wang Y M, et al. A Survey of Rollback-Recovery Protocols in Message-Passing Systems[J]. ACM Computing Surveys, 2002, 34(3): 375-408

[2] Walters, Paul J, Chaudhary, et al. A fault-tolerant strategy for virtualized[J]. HPC clusters Journal of Supercomputing, 2009, 50(3): 209-239

[3] Ong H, Sarago N, Chanchio K, et al. VCCP: A Transparent, Coordinated Checkpointing System for Virtualization-based Cluster Computing[C]// Proceedings of IEEE Int. Conf. Cluster Comput. 2009

[4] Kangarlou A, Xu D, Ruth P, et al. Taking Snapshots of Virtual Networked Environments[C]// Proceedings of Int. Workshop Virtualization Technology Distrib. Computer. 2007

[5] Scarpazza D P, Mullaney P, Villa O, et al. Transparent System-level Migration of PGAS Applications using Xen on InfiniBand [C]// Proceedings of IEEE Int. Conf. Cluster Computer. 2007: 74-83

[6] Vallee G, et al. Checkpoint/Restart of Virtual Machines Based on Xen[C]// Proceedings of the High Availability and Performance Computing Workshop. 2006

[7] Acharya A, Badrinath B R. Checkpointing distributed application on mobile computers[C]// Proc. of the 3rd Int'l Conf on Parallel and Distributed Information System. 1994: 73-80

[8] Baldoni J R, Fand Quag-lia, Cliciani B. A VP-accordant checkpointing Protocol preventing useless checkpoints[C]// Proc. of the Symposium on Reliable Distributed Systems. 1998: 61-67

[9] Tsai Ji-chiang. Systematic Comparisons of RDT communication-Induced checkpointing protocols[C]// Proceedings of Pacific Rim International Symposium on Dependable Computing. 2004: 66-75

[10] 罗元盛,闵应骅,张大方. 一种基于索引的准同步检查点协议[J]. 计算机学报,2005,28(10):1620-1625

[11] Tsai Ji-chiang. Applying the fully-informed checkpointing protocol to the lazy indexing strategy[J]. Journal of Information Science and Engineering, 2007, 23(5): 1611-1621

[12] Manivannan D, Singhal M. A Low-overhead Recovery Technique Using Quasi-synchronous Checkpointing [C]// Proceedings of the 16th International Conference on Distributed Computing Systems. 1996: 100-107

[13] Baldoni R, Quaglia F, Fornara P. An index-based checkpointing algorithm for autonomous distributed systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10 (2): 181-192

[14] Lamport L. Time, clocks and the ordering of events in a distributed systems[J]. Comm. ACM, 1978, 21(7): 558-565

[15] Mandal, Sarathi P, Mukhopadhyaya, et al. Performance analysis of different checkpointing and recovery schemes using stochastic model[J]. Journal of Parallel and Distributed Computing, 2006, 66(1): 99-107