软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(8):1907−1924 [doi: 10.13328/j.cnki.jos.004685]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

多用户服务器程序自恢复系统*

史 樾[1], 冯雨声[2], 齐 勇[1], 孙 伟[1]

[1](西安交通大学 电子与信息工程学院,陕西 西安 710049)

[2](北京航天自动控制研究所,北京 100854)

通讯作者: 史樾, E-mail: shiyi@mail.xjtu.edu.cn

摘 要: 服务器系统最无法忍受的就是因为频繁出错甚至崩溃影响正常用户的运行,因此需要系统具有自恢复能力.目前研究应用较多的自恢复策略即回滚检查点策略,并不适用于多用户服务器程序的恢复.针对多用户服务器程序的特点,设计了一种基于虚拟机的自恢复系统 VMSRS(virtual machine monitor-self recovery of service program).VMSRS 的基本思想是以虚拟机监控器为恢复主体,充分利用虚拟机作为第三方底层系统以及硬件资源的管理监控者这些特点所带来的优势,严格保证用户数据一致性、数据元数据操作原子性、恢复数据安全隔离性等;同时应用改进的 SRS(self recovery of service program)思想,在错误发生时不进行回滚,控制错误不让其影响正常用户,并保证正常用户和服务器可以顺利地向前运行,就像没有错误发生一样;并利用系统本身和 VMSRS 的清理机来避免回滚.研究工作设计实现了包括抑制错误、请求恢复、监控、存储管理等模块在内的自恢复系统 VMSRS,主要针对多用户服务器系统中的内存错误来进行恢复.通过对基本功能、基本性能、整体功能的实验分析表明,VMSRS 在不进行回滚、保证性能的前提下,提供了良好的恢复数据安全性以及完善的用户状态数据一致性保证,可以很好地恢复多线程程序,不需要对线程进行任何限制.同时,该研究工作也为在虚拟化技术条件下研究设计自恢复系统进行了很好的实践和探索.

关键词: 虚拟化;自恢复;多线程;一致性

中图法分类号: TP311

中文引用格式: 史樾,冯雨声,齐勇,孙伟.多用户服务器程序自恢复系统.软件学报,2015,26(8):1907−1924. http://www.jos.org.cn/1000-9825/4685.htm

英文引用格式: Shi Y, Feng YS, Qi Y, Sun W. Multi-User server program self-recovery system. Ruan Jian Xue Bao/Journal of Software, 2015,26(8):1907−1924 (in Chinese). http://www.jos.org.cn/1000-9825/4685.htm

Multi-User Server Program Self-Recovery System

SHI Yi[1], FENG Yu-Sheng[2], QI Yong[1], SUN Wei[1]

[1](School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

[2](Beijing Aerospace Automatic Control Institute, Beijing 100854, China)

Abstract: Long running multi-user server system may encounter frequent errors resulting in running disruptions due to its complexity of program, operating environments and user operations. This poses the need of self-recovery of system. Rollback and checkpoint scheme is a popular self-recovery strategy in current research and application, but has no obvious effects in multi-user system. In this paper, a VMM-based self-recovery system named VMSRS (virtual machine monitor-self recovery of service program) is designed according to the characteristics of multi-user server programs. The main idea of VMSRS is regarding VMM as major component of recovery, taking advantage of VM as independent underlying system and hardware resource monitor, and strictly maintaining the consistency and security

* 基金项目: 国家自然科学基金(61272460); 教育部高等学校博士学科点专项科研基金(20120201110010); 国家高技术研究发展计划(863)(2012A A010904); 西安交通大学基本科研业务费自由探索项目(xjj2014046)

收稿时间: 2014-03-03; 修改时间: 2014-07-07; 定稿时间: 2014-07-31; jos 在线出版时间: 2014-11-14

CNKI 网络优先出版: 2014-11-14 12:08, http://www.cnki.net/kcms/doi/10.13328/j.cnki.jos.004685.html

of user data and atomicity of data operation. As an improved SRS (self recovery of service program), VMSRS controls errors to avert affecting normal users in case of system crash instead of committing rollback, allowing users and servers to proceed as if no crash happens. Rollback is avoided by taking advantage of self-cleansing mechanism of system and VMSRS. The issues addressed by VMSRS design include crash suppression module, demand driven restoration module, monitor module, and storage management module. The experiment results from analyzing basic function, basic performance and integral function validate that VMSRS can provide favorable security and consistency of user data while guaranteeing performance and committing no rollback. It recovers multi-thread programs excellently with no limit to threads. Meanwhile, this exploratory study also takes part in current research of self-recovery system utilizing virtualization technology.

Key words: virtualization; self-recovery; multi-thread; consistency

1 研究背景及相关工作

有研究表明,内存错误占整个计算机错误的 30%[1],在很多服务器系统上,比如数据库、网站等存储操作为基础的应用服务器上,这个比例更大.而且这类错误的特点是出错后对系统影响很大,一般会使系统的数据产生错误,影响系统的正常运行,甚至导致崩溃.对于访存量大而复杂的多用户服务器来说,这种影响更为明显,因为多用户之间会互相干扰,特别是存储操作时,出错后必然会导致数据不一致,或者数据错误等问题,直接、间接地影响其他用户的运行.同时,出错后必然会造成内存数据的种种不一致问题,如果用目前流行的回滚检查点恢复策略进行出错后的恢复,有可能带来额外的数据不一致问题,需要设计复杂的恢复策略来避免这些问题.这样又有可能引入未知的问题,并且还会由于多用户访存的复杂性而造成恢复点设计的复杂性成倍地迅速增加.并且还会由于多用户访存的复杂性而造成恢复点设计的复杂性迅速增加.同时,检查点存储效率显著下降[2].

多用户服务器程序的一个特点是,各个用户是独立工作的,每个用户去完成自己想要完成的操作,各个用户之间并没有固定的协作和次序关系.同时,有研究表明,多用户服务器系统的共享内存数量是有限的,并且由于各个用户相对独立地工作,所以出错后内存错误数据的扩散程度也是有限的,并且在操作系统,服务器程序的自身清理机制[3]和其他正常用户的写覆盖等会让错误的内存区域恢复正常.基于以上思想,结合虚拟化技术在服务器系统上的应用,本文设计了一种基于虚拟机的面向多用户服务器程序的新型自恢复系统 VMSRS(virtual machine monitor-self recovery of service program).VMSRS 的基本思想是:

(1) 充分利用虚拟化技术所带来的一些全新的对于研究实现恢复策略很重要的特性,以虚拟机为基础和主体来实现恢复系统.发挥虚拟机在安全隔离性以及作为资源管理者和第三方系统等方面的优势,并将这些优势应用到自恢复系统的研究设计中.

(2) 在错误发生时不进行回滚,控制错误不让其影响正常用户,并保证正常用户和服务器可以顺利地向前运行,就好像没有错误发生,同时利用服务器系统自身的清理机能,避免回滚.

(3) 主要针对多用户服务器系统的内存错误以及所带来的数据错误和用户数据不一致来进行恢复,最重要的原则就是保证正常用户和服务器系统的正常正确运行,使他们不受出错干扰.而对出错用户,系统尽量保证其受限运行到程序结束点,但不保证操作的有效性.

基于虚拟机来研究设计实现这个恢复系统的最主要动机是想利用虚拟机作为操作系统和应用服务器系统之外第三方系统以及硬件资源的管理监控者等特点.对于设计恢复系统特有的优势:一方面可以安全地隔离恢复数据和相关元数据,另一方面利用虚拟化优势,严格保证用户数据一致性、数据操作原子性等.并且如果恢复系统基于操作系统,那么相当于操作系统恢复其本身,必然有些系统错误无法恢复,而虚拟机作为独立系统可以恢复操作系统的自身错误.除此之外,虚拟化是当今服务器上的一个主流技术,是未来的重要发展方向,研究实现一个基于虚拟机的适用于服务器程序的自恢复系统是一种很好的尝试和有意义的探索[4].

这种不进行回滚的恢复方式最重要的应用场合如前所述,就是多用户的服务器系统,因为这种多用户系统的各个用户之间并不是协作去完成某个任务,而是各自在做各自的工作.如果因为一个用户出错而进行回滚恢复,那么在现今服务器系统越来越复杂、用户越来越多的今天,显然是严重影响正常运行且有些得不偿失的方

式.所以本文采用了这种不进行回滚的服务器程序恢复方式.

本文研究设计并实现了基于虚拟机的多用户服务器程序自恢复系统 VMSRS.VMSRS 充分结合了虚拟机和服务器程序自恢复思想 SRS(self recovery of service program)在多用户服务器上各自的优势.利用了虚拟机作为硬件资源管理监控者和相对于服务器系统和操作系统之外第三方系统的优势,研究实现了具有虚拟机优势的改进的 SRS 自恢复技术.VMSRS 解决恢复的安全隔离性以及数据一致性等老式的 SRS 系统未解决的问题,是新型的针对内存错误以及内存错误带来的用户数据不一致、不安全等问题的恢复多线程服务器程序自恢复系统.

2 相关技术与问题

2.1 虚拟化技术

多核已经成为计算机的主流配置,而多核在带来性能提升的同时也对软件的设计和实现提出了全新的机遇和挑战.作为系统硬件资源的管理者,多核操作系统的研究近些年来一直是热点,针对操作系统在多核平台存在的性能瓶颈问题,研究人员做了大量研究工作来改进性能,主要的工作可以概括为以下几个方面:对现有操作系统的改进,重新设计操作系统内核,利用虚拟化技术[5].通过这些技术可以提高虚拟机监控器对硬件资源虚拟化的效率,从而使整个系统的性能接近于无虚拟化的操作系统性能.但是,这些技术仅仅提高了虚拟化后系统的性能,对现有操作系统的性能并没有提升或改进.在现有高并行网络应用场景中,操作系统已经成为这些应用性能进一步提高的瓶颈.

虚拟化技术由来已久,最早的虚拟机定义为有效的、独立的真实机器的副本.它已有 40 年的历史.最早使用虚拟化技术的是 IBM 7044 计算机,首次使用了请求调页和系统管理程序调用.虚拟机必须保证其上可以运行正常的操作系统,在虚拟机内虚拟处理器的指令集的一个子集能够直接在物理处理器上执行.人们常说的虚拟机软件其实是虚拟机监控器 VMM(virtual machine monitor).VMM 就是能够为计算机系统创建这些副本的软件.虚拟机按抽象的层次不同可以分为 5 个级别的虚拟化:硬件抽象级虚拟化;指令集级虚拟化;操作系统级虚拟化;函数库级虚拟化;应用层级虚拟化[6].本文主要针对硬件层级的虚拟化进行研究.

目前个人 PC 和服务器的主流架构都是 X86 架构,未来很长一段时间这种架构都将是主要的计算机体系结构,在 X86 系统上,虚拟化技术的研究已经很具规模和成熟度,主流虚拟化技术有以下几类:全虚拟化技术、半虚拟化技术、硬件支持的虚拟化技术.

2.2 自恢复技术

自恢复技术即为系统在外部不知晓的情况下自行进行错误检测和错误后自行恢复的技术,基本恢复方式的研究已经有 20 多年的历史.自恢复技术最先出现的研究方向是重启恢复,提出了重启整个系统以使应用程序从错误中恢复的方式[6].之后分别有研究工作者提出了软件再生技术和微重启技术,这些技术都试图将系统恢复到遭遇错误之前的一个清洁的状态.重启系统恢复技术需要较长时间的恢复时间,将导致一个明显的系统延时出现.微重启方式相对而言恢复的速度更快,但是它需要系统的一个完全重写,并且在状态同步方面存在问题.重启恢复的最大缺点是需要大量的恢复时间,在恢复过程中无法进行作业,并且当状态被清除后,需要重做很多工作,恢复代价相对于其他恢复方式过于昂贵了.

回滚检查点策略的基本模式是定期或触发式的进行检查点存储和更新,然后在错误发生时,用优秀的回滚策略将系统状态恢复到最近检查点状态.恢复策略一直以来面对几大问题:多线程一致性无法保证;都或多或少需要系统重构设计;定位错误的能力不强;对程序进行不安全的预测;恢复时间过长等.上述局限性是限制回滚检查点策略应用于多用户服务器的根本原因[7],针对多用户服务器程序的应用,对这些局限性进一步分析如下:

首先，在多用户服务器上运行的大部分回滚检查点恢复策略都无法保证多线程或多进程的一致性问题.这是因为回滚检查点恢复技术中基础即为回滚,而回滚往往是针对某个错误线程或进程进行的,这样,当这个进程或线程发生回滚后,其他正常进程或线程如果已经访问这个进程所修改过的内容,就会造成状态的不一致.例

如,著名的 Recovery Domain 恢复方式是在内核中进行恢复,程序处于内核特权级,无法保证用户级多线程的一致性.

其次,有些回滚检查点策略理论上可以保证多线程或进程的一致性,做法是将相关的进程或线程全部回滚,这样做需要很复杂的追踪算法,要设计复杂的检查点策略以及回滚策略,多线程在程序上实现起来是很复杂和低效的,并且这样做的代价非常大,严重影响了正常用户的运行.

最后,回滚检查点策略需要定时或触发式地进行检查点的存储和更新,还需要简单或复杂的运行状态追踪,这样会对服务器程序产生影响,特别是多个用户并发运行,需要检查点设计得更为复杂,需要存储大量数据并且严重影响效率,服务器的响应时间和吞吐率大为下降,而这两点恰恰是服务器程序最重要的性能指标.

2.3 内存崩溃问题

由于多用户服务器程序自身的特点,其内存操作也具有不同于一般程序的特点.简言之,在多用户服务器程序中,多个用户是独立并发运行的,各个用户之间并没有明确固定的交互关系,所以,某个用户发生错误崩溃后,其他用户访存受到影响的程度就是一个值得研究的问题.

研究方法首先通过人为标记 max 数量的崩溃内存地址.之后,通过跟踪和计算,得到最后,也就是 last 时的内存崩溃值.在代表性服务器程序 mysql 上得到的结果如图 1 所示,max 曲线代表最初标记的崩溃内存地址,last 代表最终结束时崩溃内存地址.图 1 可以看到,被标记成崩溃错误的地址仅仅传播一个很小的范围,并且在用户进程结束后数量有显著的下降.这样我们可以得到,如果服务器中有用户出错且我们的目的主要是保证其他用户的内存数据不受影响,那么也许不需要回滚这样复杂耗能的操作,而只需要借助服务器自身的特性,让崩溃的内存自己被清理.造成这种崩溃内存在总量上看不向外传播,反而越来越少的现象的原因有很多,系统原因是服务器自身的一个清理机能.这个机能首先是因为服务器程序和操作系统会有一个垃圾清理的功能,另一个原因是其他正常用户对这些内存也会进行写覆盖,这样错误内存也会消失.而另一重要原因如图 2 所示,共享内存少.

造成图 1 的原因是由于系统的清理机能,即操作系统和服务器程序自身的垃圾功能以及正常用户的写操作,之所以清理机能能够有这么显著的效果是因为各个用户进程之间共享的内存地址并不多,这也可以说明为什么崩溃传播只在一个小范围内进行.有研究工作者已经对这方面问题做过深入的研究和测试.如图 2 所示,mysql 上共享内存的研究,可以清楚地看到,用户之间的共享内存所占比例相当小,并且随着用户请求数量的上升,所占比例越来越小[2].

在其他代表性服务器程序如 cvs,apache,squid 等上做以上两个实验也得到类似的结果.基于这些实验结果可以得出,在多用户服务器程序上,出错后应该可以不进行回滚,只要仔细设计出错后的运行方法就可以达到系统自愈.

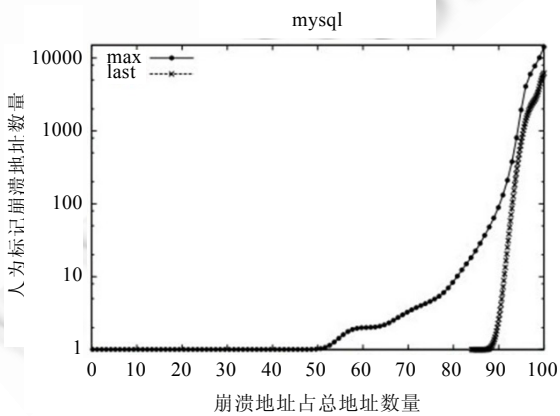


Fig.1 Linear graph of the collapse address

图 1 崩溃地址研究线性图

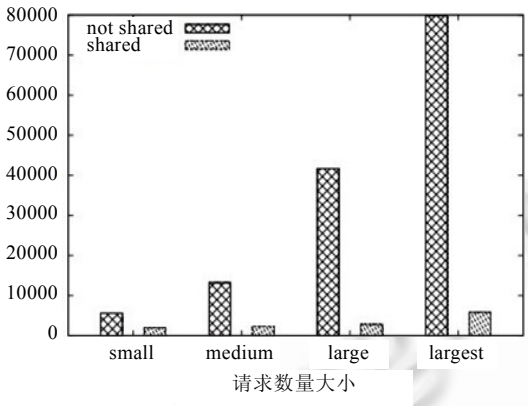


Fig.2 Relationship between the user requests and shared memory

图 2 用户请求与共享内存之间的关系

2.4 多用户服务器程序恢复技术

在多线程已经是基本程序模式的今天,如何保证多线程一致性,一直是恢复技术研究的问题.而如上所述,回滚检查点在这方面有先天弱势[6].于是,近年来,科研工作者针对这种应用需求提出了一种新的恢复模式,服务器程序恢复模式 SRS,SRS 的一个基本应用场合就是多用户服务器程序.它要保证在有用户进程出错的情况下,其他的用户程序不受影响.首先需要保证的是,当用户进程出错后对其他用户进程的影响要尽量小,也就是错误内存的扩散要尽量小,否则,要保证向前运行的代价就太大.而由前一节的内存问题研究证明了这种代价是比较小的,相对于回滚检查点策略的代价而言会小不少.

基于以上的研究结果,在多用户服务器上进行回滚检查点型恢复是不划算的,所以应用于服务器程序的特殊恢复的思想就应运而生了,这种恢复策略与先前的恢复策略有很大的不同,对这种恢复策略的基本思想和依据概括为以下几点:

(1) 根本原则是当服务器上某些用户进程或线程出现错误时,服务器不进行回滚操作而是继续向前运行,并保证正常用户和服务器不受错误用户的影响.同时,尽量让错误程序运行到自己的结束点,正常结束,来使用系统自身所具有的清理回收能力.

(2) 大部分用户访存是独立的、非共享的,并且在崩溃后,崩溃的传播在可控范围内.而无论是操作系统还是很多服务器程序,都有自己的垃圾清理和资源回收机制,利用这个机制,大部分错误不需要设置检查点和回滚.

(3) 多线程进程服务器首先要保证的是整个服务器的运行正常和正确用户访问的正常运行,所以对于出错后所有用户回滚这种方式是无法容忍的.

基于这种思想,这种恢复方式重要的两个设计方面是,首先在发生错误时抑制错误用户进程的运行,让它的崩溃错误不会传播太远,不至于导致服务器崩溃或者其他用户访存不正确.其次是保证其他用户进程访问的内容都是正确的,保证正常用户进程的正确进行.这里就点出了这种思想最重要的两个方面,一个是抑制方面,在出错后,对错误用户进行抑制,保证出错用户不会将错误进一步扩散,另一个是请求恢复方面,在出错后,保证正常用户不受到错误的干扰,如果正常用户需要访问错误用户更改过的内存地址,则启动恢复,将这里的值恢复成之前的正确值.

3 VMSRS 关键问题分析研究

3.1 VMSRS设计问题分析

本文以轻量级虚拟机为恢复主体并与改进的 SRS 恢复策略相结合,研究设计了 VMSRS 恢复系统,充分结合了两种技术思想的优势,针对多用户服务器程序自恢复.VMSRS 主要是对内存错误以及内存出错后造成的用户数据不一致和数据错误等问题进行控制和恢复,保证内存数据错误不扩散,不影响其他用户[8].VMSRS 的基

本设计思想如下:

当多用户服务器上某些用户线程出现错误时,服务器不进行回滚操作,而是继续向前运行,同时保证正常用户和服务器本身不受错误用户的影响.要保证正常用户和服务器本身不受错误的影响,就必须对错误的扩散进行抑制,只有让错误崩溃扩散范围尽可能地小,才有可能保证正常用户和服务器不受影响.同时保证正常用户所访问的数据是正确的,不是被出错进程所影响的,又为了尽量利用最后的系统清理机能,只有当正常用户程序访问到被更改过的共享内存时才进行数据清理恢复.

针对必须回滚的特殊重要程序,未来 VMSRS 会用定制化的内核系统在多核服务器上分出部分核来做进程备份,当进程出错后,马上由其备份进程进行顶替.同时,VMSRS 对错误进程所造成的错误会进行抑制,确保其不会造成其他影响,相当于将错误状态恢复到之前的正确状态,但与一般回滚策略有显著不同.

针对必须重启的问题,若是进程重启则与上面的处理方法一致.若是硬重启,虚拟化技术保证不会出现这种情况,否则,是另一个范畴的问题了.

VMSRS 比之前的回滚检查点等恢复模式更适用于多用户服务器程序,它解决了回滚检查点策略在多用户服务器应用上的缺陷:避免 rollback 策略的昂贵开销;保持了恢复后用户数据之间的一致性;避免检查点复杂的设计实现,同时,在多用户服务器应用中,保证了正常用户的正确运行,不受错误用户的影响,保证数据的一致性.现代服务器和操作系统都有一定的抵御出错的能力,但是出错后所带来的一系列问题,包括用户的状态一致性,因系统错误影响的数据出错以及恢复系统中最基本和重要的恢复数据的安全性等问题都是系统无法感知和抵御的.另外,用于恢复的(数据,元数据)关联对更新的原子性问题一直是错误追踪和老式 SRS 恢复系统无法解决的问题.本文所研究设计实现的 VMSRS 很好地解决了上述的一系列问题,它与以往包括 SRS 在内的恢复技术相比,在保证恢复数据安全,保持用户数据一致性,保持数据和元数据更新的整体原子性,错误定位更加准确且不用更改内核代码和垃圾清理更加主动、及时、有目的性方面有明显的优势.

会有一些应用服务器或者一些服务在发生故障时需要“回滚”甚至重启.在 VMSRS 的恢复思想下,现今和未来可预见的服务程序会越来越多地细分成较小的进程,在多个 CPU 上进行协同运行,为了充分利用多核的性能,必然有越来越多的进程在不同的核上进行相同的处理工作,必须回滚的情况会变少,因为崩溃进程的工作会由其他兄弟进程接手,崩溃进程只需自动消亡即可.VMSRS 设计前向恢复也和这种预期有一定的关系.

3.2 VMSRS恢复状态模型研究

如图 3 所示为 VMSRS 恢复相关数据存储与操作及内存分布图[9],右边的方框表示内存的分布,可以看到客户操作系统有一部分内存是无法看到的,那部分内存是虚拟机专有的,而虚拟机可以看到所有内存.虚拟机专有内存对操作系统是物理不可见的,当客户操作系统启动后,它所读到的物理内存信息就已经被虚拟机所更改,所以无论是发生错误还是恶意攻击,都不可能破坏存储在虚拟机中的数据.

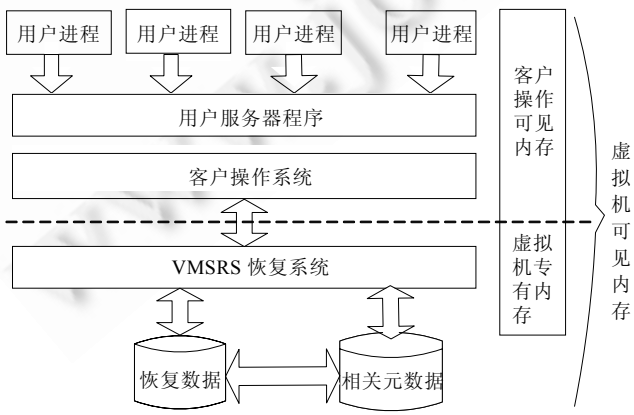


Fig.3 VMSRS recover related data storage and operation and memory-distributed map

图 3 VMSRS 恢复相关数据存储与操作及内存分布图

图 3 还显示了对恢复相关数据操作的原子性,从图中可以看到,服务器程序和客户操作系统对恢复相关数据操作的唯一途径就是陷入虚拟机,然后通过 VMSRS 系统进行相关操作,而虚拟机一旦获得 CPU 的控制权,客户级的系统相对于 CPU 来说就相当于消失了,这样,除非虚拟机完成了所有的操作,然后返回客户操作系统,否则客户级的系统无法进行任何操作.这样就保证了每次陷入虚拟机的操作都是原子的,进而保证了恢复数据和相关元数据的一致性.

VMSRS 与之前的恢复系统相比的优势除了上面提到的安全隔离性以外,还可以保证多线程的数据一致性,图 4 是一个 VMSRS 保证多线程一致性的简单模型,这里只有 3 个线程,*P*1,*P*2,*P*3,其中 *P*3 运行到一定时间后发生错误(aulted).图中菱形、圆形和深色圆形分别代表 3 种基本操作.

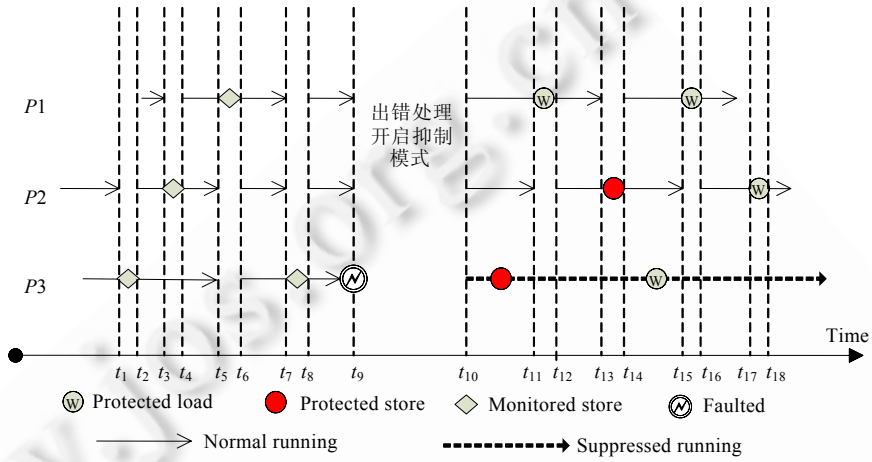


Fig.4 Sequence diagrams of VMSRS multithread execution

图 4 VMSRS 多线程运行时序模型图

图 5 对图 4 中同一数据的状态转换进行了演示.由图 5 中可以清楚地看到数据状态转变的细节.其中,圆圈表用户数据 store 操作的状态,方块代表用户数据 load 操作的状态.图左半部分是出错前的情况,圆圈所代表的用户数据状态随着每一次 store 在改变,而下方的恢复数据的状态也随着每次 monitor store 而保存前一次的旧值,这里的 *P*0 代表在这 3 个进程之前运行的进程,所以最初的恢复数据值因为 *P*3 发生 monitor 写的缘故被存为 *P*0 时的值.而右半部分是错误发生后,进入抑制模式之后的数据状态,首先,当 *P*1 发生 load 时发现现有值是出错进程 *P*3 所写的,所以就进行了恢复,将恢复数据中所存的值写入,接着 *P*2 进行了 store,这时不光对用户数据进行了改动,而且对恢复数据也进行了改动.之后的两次 load 所得的值就都是 *P*2 所 store 的了.

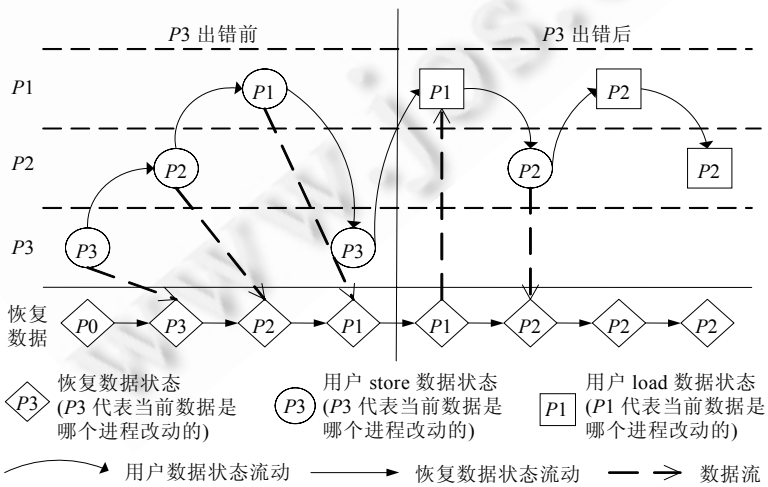


Fig.5 Data state transition diagrams of VMSRS multithread

图 5 VMSRS 多线程数据状态转换图

3.3 VMSRS恢复时间模型研究

图 6 所示为自恢复系统状态转换时间模型图,S0,SP,SF,SR 分别代表正常运行状态、亚稳定状态、出错状态以及恢复状态.t_{XY}代表从 X 状态到 Y 状态转化需要的时间,t_X表示 X 状态的运行时间.

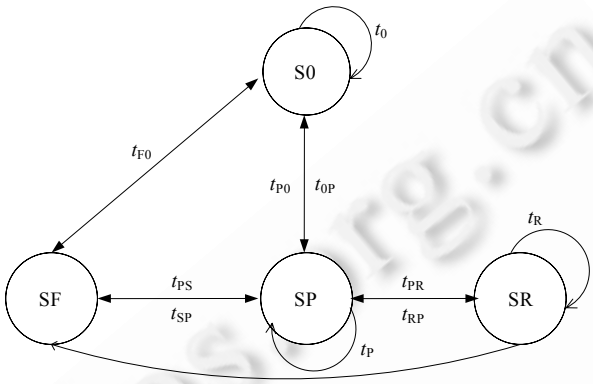


Fig.6 Model diagram of the restore system state transform time

图 6 自恢复系统状态转换时间模型图

根据图 6,分析回滚检查点恢复系统和 VMSRS 恢复系统的运行时间可得出,当 N 达到一定值后,VMSRS 系统的运行时间比回滚检查点系统的运行时间要小,而且随着 N 的增加,这个差距会越来越大.这就从理论上说明了在时间性能方面,VMSRS 恢复模型也要比回滚检查点策略模型更适合于多用户的服务器程序.

4 VMSRS 恢复系统的设计

针对现今多核系统的特点,充分利用现有操作系统已经发展成熟的资源管理系统,本文设计并实现了一个应用硬件虚拟化技术,借鉴半虚拟化思想的多核系统上的虚拟机系统.基本设计思路是将虚拟机监控器作为操作系统与硬件之间的恢复主体,操作系统是与硬件直接交互,直接管理硬件,而虚拟机监控器作为一个监视控制单元,负责整体硬件资源管理和对操作系统的硬件行为进行监控拦截.这里采用硬件虚拟化和半虚拟化相结合的技术,虚拟机监控器需要多个模块的协同工作才能实现这样的功能.

4.1 VMSRS总体设计

VMSRS 恢复系统发挥了虚拟机的优势,保证可以顺利、正确地运行多线程程序.同时对 SRS 恢复策略进行很多改进,并且加入函数库的方式来进行用户级的错误监控和运行监控,提高恢复的准确性,避免修改内核代码,同时也加入简单的自身垃圾回收功能,对仅仅依靠系统自身清理的老式 SRS 技术是一个很好的补充.基于以上设计理念,本文设计的基于虚拟机的前向恢复系统的体系结构如图 7 所示.主体部分都在虚拟机内设计实现,外部提供函数库作为接口,下面就 VMSRS 进行简要说明.

VMSRS 恢复系统主要由虚拟机内实现的内存管理和存储模块、SVM 监控模块、错误用户抑制模块、正常用户请求恢复模块、辅助功能模块组成,辅以在 guest 级实现的面向对象式的交互监控函数库模块,各个模块相互协作,实现了一个新型的恢复系统.

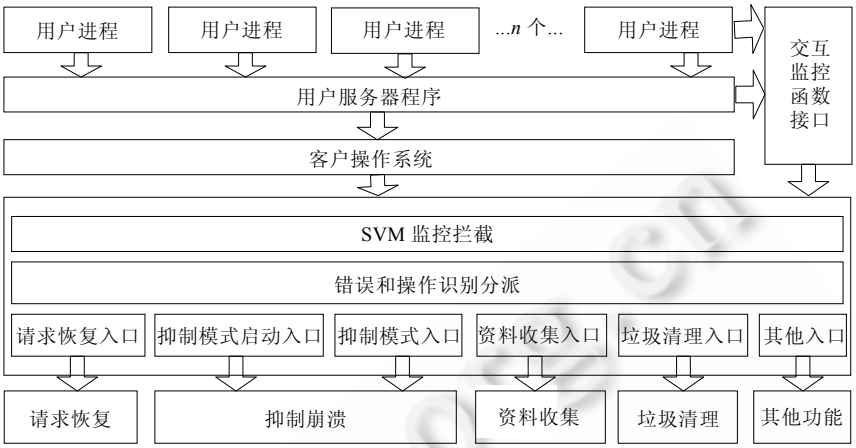


Fig.7 VMSRS system structure

图 7 VMSRS 体系结构

4.2 SVM监控模块

整个虚拟机内部由 SVM 监控模块负责所有错误监控和必须的操作监控以及判断处理等工作,所有的陷入虚拟机的操作都是通过 SVM 监控模块进行的.本文的 SVM 监控模块分为对上、对下两层结构,分别为其上的客户操作系统和用户服务程序以及为其下的自恢复系统处理模块提供接口.对上来说,SVM 监控模块进行的监控拦截工作直接拦截客户操作系统的错误,并通过交互监控函数库接口来拦截用户服务器程序,用户进程和用户服务器程序通过两种途径被 SVM 监控模块监控和控制.一种途径是传统的直接通过 VMM 虚拟机监控客户操作系统来实现,所有用户操作和错误最终都会反映在操作系统上,VMM 监控无法到达用户级,对用户服务器程序和用户进程的监控管理是通过监控管理操作系统实现的.另一种途径是通过被系统提供的交互监控函数接口来进行与用户服务器程序的直接交互,这种途径使得 VMM 可以直接监控到用户层级的错误和操作,对错误的识别更加准确,而且可以在用户触发操作系统错误之前使其陷入操作系统,这样做的优势很明显,不但使 VMM 越过了复杂的操作系统直接监控自己需要监控的操作,而且避免了 VM 处理操作系统错误的复杂性和不确定性.对下来说,SVM 监控模块所做的工作是把拦截错误和操作进行识别、分类、分派,然后将识别好的信息通过其中所包含的各个模块传递给恢复系统的下层各个处理模块来进行处理.这里,SVM 监控模块相当于很多体系结构中的代理.只有通过 SVM 监控模块的识别才能确定操作和错误到底是哪类的,应该给哪个模块去处理等等.

该模块的实现是基于虚拟机 SVM 模块的,所以自恢复系统的 SVM 监控模块的主体部分都在虚拟机 SVM 模块的主要处理函数 svm_exit_handler()中实现,所有客户操作系统被拦截的中断操作都进入自己处理函数入口,利用 SVM 硬件虚拟化技术提供的特殊中断 VMMCALL 来实现和客户操作系统以及用户级服务器程序和用户程序的交互.另一数据结构是 vm_frame,该结构作用是作为一个堆栈空间保存进入虚拟机和弹出虚拟机时客户操作系统所需要压栈保存的值,其中,结构 generl_regs 保存寄存器的值.

4.3 虚拟机内存管理和存储模块

内存管理模块主要依附虚拟机内存管理系统,这里的内存管理模块主要设计用于恢复数据内存区域的存储,包括恢复数据、元数据等的存储和管理,包括一系列基础函数和基础数据结构.本文设计的 VMSRS 主要针对服务器软件系统中的内存错误以及内存出错后带来的数据不安全、不一致等问题进行恢复.

恢复系统的内存管理模块是基于 VMM 的内存管理模块来设计实现的,这要和内存区域存储模块和进程状态信息存储模块相结合,为请求恢复模块、资料收集模块和垃圾清理模块提供基础的操作和数据存储方式.利用以上两类基础函数,内存管理模块完成了与其他模块的交互操作.根据自恢复系统的特点还设计了一些只应

用于自恢复系统的特殊的内存基础操作,比如内存块的复制传递、元数据数据的更新复制操作等利用以上两类基础函数,内存管理模块完成了与其他模块的交互操作.

4.4 抑制崩溃用户模块

MSRS 的重要功能就是抑制错误的用户线程,使其在出错后继续运行,直到正常的程序结束点,而且在其继续运行期间,抑制其所有访存操作,保证错误用户线程的继续运行不会影响到服务器系统或者正常用户线程的运行.本文基于虚拟机自恢复系统所实现的抑制功能模块是在虚拟机监控器的基础上,利用 SVM 监控模块的监控功能.在有用户进程或线程发生错误时,最快地拦截错误,使错误在扩散之前就陷入虚拟机,然后开启抑制模式,进行初始化工作,然后返回客户操作系统.

VMSRS 的基本理论特点就是在某用户进程发生错误时,对该错误用户进程进行抑制,所谓的抑制是指对错误用户进程的运行进行监控,一旦发现有访存操作,则将该操作进行处理,目的是使错误用户进程不再将自己的错误数据扩散到更大的范围、更多的物理地址去,这样可以最大限度地保证正常用户不受错误影响,可以继续运行下去.VMSRS 系统中,错误用户进程在发生错误后不进行回滚或者直接终止,而是继续有条件地向下运行,直到一个正常的程序结束点,这一切都需要错误用户抑制模块进行控制和操作.

4.5 正常用户请求恢复模块

请求恢复的基本思想是,只有当正常用户读访问共享内存时才进行恢复.本文设计的 VMSRS 基本要求是保证正常用户不受到错误的干扰.在我们所针对的访存应用中,这种干扰来自于被出错用户进程污染的数据和内存区域,所以要保证正常用户的访存操作请求能够得到正确应答,就必须保证正常用户访问被污染的内存区域之前,该区域已经被清理恢复,这些功能都是由请求恢复模块来实现的.同时,请求恢复模块还有更新恢复数据的功能,该功能是指,当某被污染的地址被正常用户写过后,这个地址的值就干净了,这时请求恢复模块会把虚拟机中存储的数据即相关元数据都进行更新.

正常用户请求恢复的基本设计思想是在保证所有错误共享内存都被恢复的前提下,最低限度进行恢复操作,并同时保证正确的共享内存数据不被错误地覆盖.VMSRS 基本要求是保证正常用户不受到错误的干扰,在我们所针对的访存应用中,这种干扰就来自于被出错用户进程污染过的数据和内存区域,所以要保证正常用户的访存操作请求能够得到正确的应答,就必须保证正常用户访问被污染的内存区域之前,该区域已经被清理恢复,这些功能都是由请求恢复模块来实现的.同时请求恢复模块还有更新恢复数据的功能,这个功能是指,当某个被污染的地址被正常用户写过后,这个地址的值就干净了,这时,请求恢复模块会把虚拟机中存储的数据即相关元数据都进行更新,比如,一个正常的用户进程对某被污染的内存地址进行了写操作,可是相关恢复内存区域存储的恢复数据和元数据没有及时更新,导致下一个正常用户读这个区域时,误以为还是污染的,用旧值进行恢复,导致访存数据不一致.这个功能还使得将来的操作受影响较小.

4.6 辅助功能模块

辅助模块主要有垃圾回收和资料收集两个功能.在错误程序结束时对已知需要恢复的内容进行一次性恢复处理,这样做比单纯垃圾清理功能更加准确、高效、有目的性.资料收集功能在所有用户进程开始前运行,主要目的是收集需要监控和共享的内存.本自恢复系统的垃圾清理模块设计在虚拟机的内部,作为一个功能较为单一的模块存在的,该垃圾清理功能被触发的时机是某个出错用户进程结束时,前面已经提到,当每个进程结束时,会陷入一次虚拟机.

VMSRS 的原则就是让错误程序继续运行到程序的正常结束点,然后利用操作系统和服务器程序自身的垃圾清理机制来进行脏数据的清理工作,如前文所提到的,这样做有一定的盲目性,因为系统程序本身并不知道某个用户进程是否出错,而垃圾清理也不是在某个程序一结束之后立即进行,所以,清理效果无法保证.VMSRS 设计了一个简单的垃圾清理机制,利用了虚拟机的信息优势,在每个错误进程结束时进行一次垃圾清理工作,较好地保证了脏内存区域及时被恢复,不影响其他正常用户使用.

在所有的用户进程运行之前,VMSRS 要做两件事,一件是系统的初始化,比如存储区内存的分配、各个重要

数据的清零等工作,之后需要对服务器程序进行初始资料收集工作,主要是收集系统的一些基本信息和服务器系统中需要监控的地址,本自恢复系统对于以上功能,在交互监控函数接口模块提供两个接口供用户使用,资料收集模块目前只需要用户给出所需监控的内存的虚拟地址即可,其他的由本系统完成.系统的初始化工作由Init_profile 函数完成,该函数负责对所有的系统状态、特殊参变量,以及与客户操作系统进行交互的数据项进行清除和初始化工作,并对存储空间进行清空.资料收集工作由 profile 函数进行,该函数接收用户传来的虚拟地址等内容,进行共享内存区域的存储和初始化和相关元数据的存储及初始化等工作.

4.7 交互监控面向对象式函数库

客户级(guest-level)面向对象式的函数库,用于服务器系统和用户线程直接与虚拟机监控器交互,以及便于VMM 直接监控拦截用户级的错误和操作,而不必通过操作系统.本系统选择函数库接口的方式来进行虚拟机与用户服务器系统的直接交互.这样,不但错误定位准确,而且避免了很多修改内核的复杂问题,同时,用户编程方便,只要接口不变,函数库是封装的,用户需要升级恢复系统版本时只需更新函数库即可.本文所设计的函数库基本有以下类型:

第 1 类是基本访存操作的封装,比如对于读写操作,赋值操作等.第 2 类是对基本的访存函数的封装,比如printf 等函数的封装.第 3 类是对基本内存操作函数的封装,比如 malloc,memset 等函数封装,保证其操作都在恢复系统的监控下进行.第 4 类是对重要的内存相关操作的封装,比如指针增减操作等,检查指针操作是否非法.第5 类函数是为以上这些函数提供服务的函数以及虚拟机通信的相关函数,如 crash_suppression_os 等.

本恢复系统所设计的函数接口提供给服务器系统设计者,程序员们只要使用本系统提供的函数库就不仅可以实现用户级的服务器程序和用户程序和虚拟机的良好通信,并且可以使虚拟机的监控直接植入用户级,使得虚拟机对于错误和操作的监控和拦截更加简单、高效,避免不必要的性能损耗和各种内核问题.

5 测试与结果分析

本文所设计实现的 VMSRS 恢复系统可以运行在真实的多核 X86 架构的服务器上,在以下服务器上进行测试,见表 1.

Table 1 Test environment

表 1 测试环境

	DELL T680	SUN X3800-m2
CPU	AMD Opteron(tm) Processor 2350	AMD Opteron EE Processor 2377
体系架构	ccNUMA	ccNUMA
核心数	8	32
内存	16G	256G
内核版本	Linux 2.6.24 以上	Linux 2.6.24 以上
操作系统版本	Ubuntu,debian	Ubuntu,debian

5.1 VMSRS基本功能测试与评价

在基本功能测试中,选取本地多线程服务器程序作为测试对象,其基本测试环境如下:

测试数据集:公司职员的花名册表,用户可以对其中的一些数据进行随意的改动.

测试错误集:越界访问,主要是指针越界,或因 mmap 引起的内存空洞等.

基本测试场景:多个用户线程并发对数据集进行更改读取等操作,其中某用户出错.以下的每个基本功能测试都在此场景的基础上加入一些额外的因素,以针对性地测试某个功能的情况.

5.1.1 恢复效果测试与评价

不采用本恢复系统的一般应用程序运行如图 8 所示,当某个用户写入数据并打印后,发生指针越界错误,这时服务器程序就会发生段错误,而用户程序也会终止.即使发生错误较小不会使系统直接终止,但出错程序所写入的数据已经不可用,后续用户如果读取这些数据,那么读到的也是错误数据.

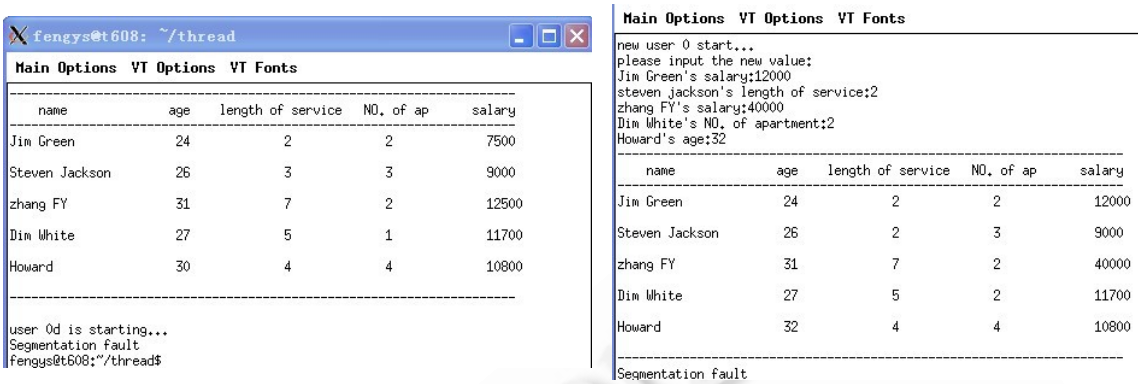
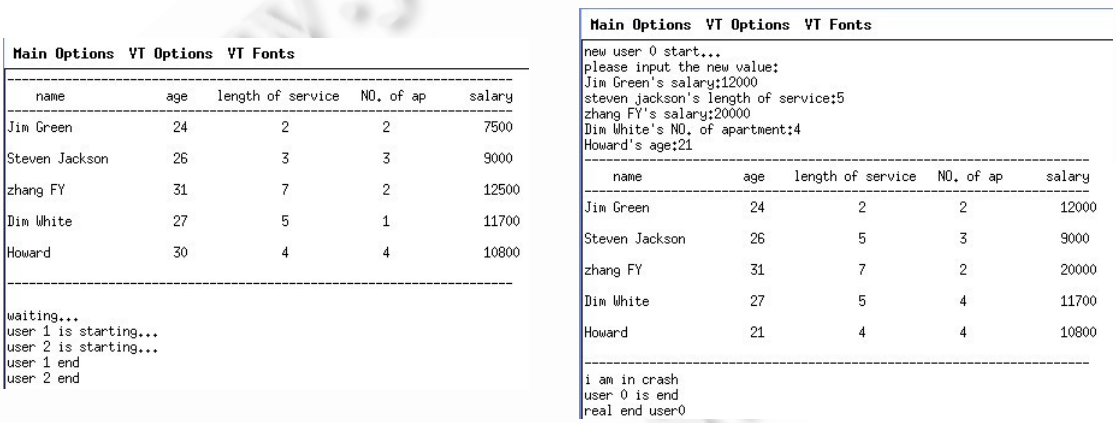


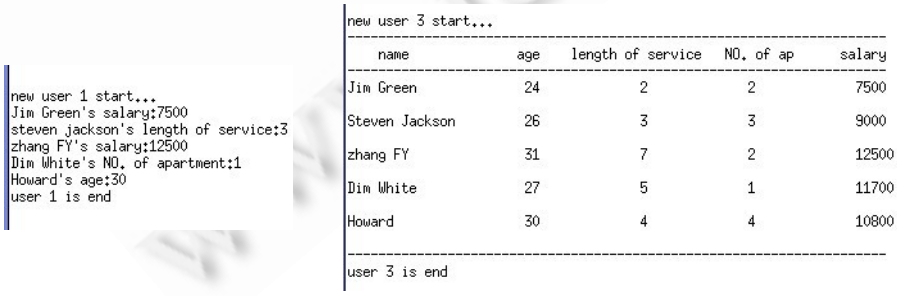
Fig.8 General server program after the user error

图 8 一般服务器程序用户出错后情况

有 VMSRS 的系统测试结果如图 9 所示.错误发生后,服务器程序继续运行,正常用户同样正常运行,而错误进程虽然出错,但是不崩溃,只是受限制运行到程序的结束点,然后结束.服务器程序和正常用户不会感觉到有错误发生,而错误用户会知道自己的执行出错.



(a) 服务器段 (b) 出错用户



(c) 正常用户

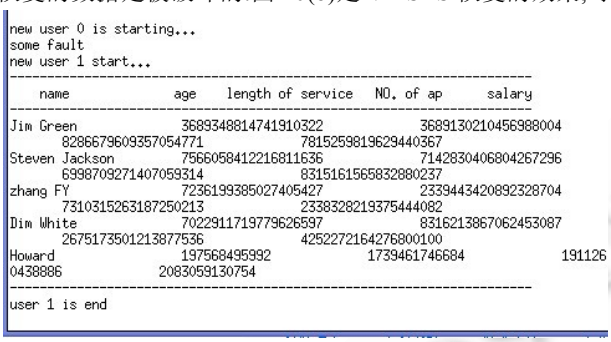
Fig.9 VMSRS recovery after the system error

图 9 系统出错后的 VMSRS 恢复情况

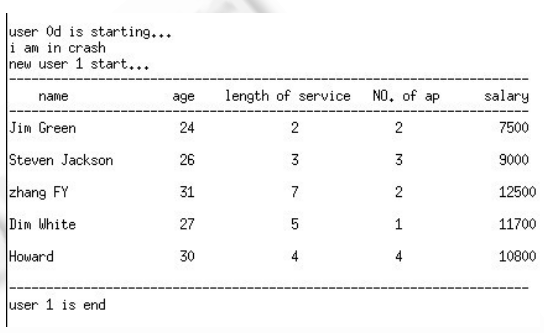
5.1.2 安全隔离性测试与评价

测试场景:某用户出错,并故意或无意破坏了用于恢复而存储的数据(常见场景是用户程序 BUG 或被病毒破坏).

测试效果:如图 10(a)所示,普通的恢复系统在某用户出错后,启动恢复策略,将存储的恢复数据恢复出来.可恢复的数据是被破坏的.图 10(b)是 VMSRS 恢复的效果,可以正确地将原先的数据全部恢复出来.



(a) 普通恢复



(b) VMSRS 恢复

Fig.10 Contrast figure about whether recovery area is changed

图 10 恢复区是否被更改对比图

测试分析:基于虚拟机实现的恢复系统相对于其他恢复系统有一个很重要的优势,即良好的隔离性,可以保证恢复数据和相关元数据的安全性,这一点是其他非基于虚拟机的恢复系统很难达到的.通过图 10(a)和图 10(b)可以清楚地看到,VMSRS 在保证恢复数据安全方面具有先天优势,这种优势的保证是通过最初的物理级别的改动实现的,虚拟机的内存区域是用户和操作系统所访问不到的,可以保证恢复数据区的安全.普通恢复系统在用户出错后,是不可能 100%地保证恢复数据安全的,特别是在用户恶意攻击的情况下.

5.1.3 一致性测试和评价

一致性测试是测试的重点,VMSRS 是保证多线程服务器程序可以正常运行而不需要进行线程顺序等的限制,所以用户数据的一致性是最基本的要求.一致性的情况有很多种,这里设计 3 个不同的测试场景:

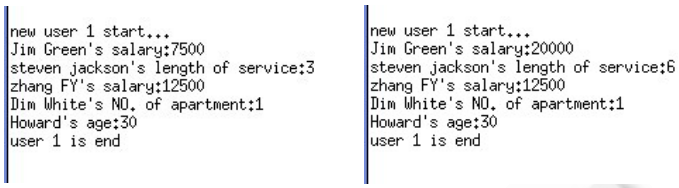
测试基本场景:设定最初服务器数据的值以及出错用户 0 更改后的值.

(1) 测试场景 1:用户 0 出错后,需要开启抑制模式,进行一定的初始化工作,这时另一个线程占用了 CPU,对出错的数据进行读取.注意,此时的抑制模式还没完全开启,无法控制发生的情况.

测试目的:抑制模式初始化其他线程获得 CPU 时造成抑制模式开启不完全所带来的一致性问题.

测试效果:如图 11 所示.图 11(a)表示 VMSRS 的恢复效果,图 11(a)的结果与出错用户 0 更改前的结果,也就是最初服务器数据的值相同,表明错误得到了恢复.图 11(b)是没有虚拟机的 SRS 恢复方式,图 11(b)的值前两项已被错误用户 0 更改,而后几项在未被更改时,错误用户 0 已出错,造成正常用户读取的一部分是错误数据而另一部分是正确数据,导致数据不一致现象.

测试分析:没有使用虚拟机 SRS 恢复方式,当某线程出错时,需要进行抑制模式的开启初始化等工作,但硬件并不知道这些情况,恰好在抑制模式的处理还未结束时时间片用完,这就可能造成正常用户读取一部分错误数据(初始处理完成前)而另一部分是正确数据(初始处理完成后).而在 VMSRS 恢复中,虽然多个用户运行,这时,用户 0 发生错误,陷入虚拟机,进行操作,其他用户无法进行任何操作,从虚拟机返回后,其他用户才可继续操作,因为虚拟机和操作系统不在一个层级上,所以陷入虚拟机后相当于操作系统及其上的服务器程序都是不存在的.这就很好地保证了恢复系统可以安全充分进行抑制模式开启工作.



(a) VMSRS (b) 未使用虚拟机的恢复

Fig.11 Based on the virtual machine recovery way's consistency compared with other error recovery mode's consistency

图 11 基于的虚拟机恢复方式与其他恢复方式出错后一致性对比

(2) 测试场景 2:用户出错后,在回滚检查点恢复中,回滚后重新执行之前的操作,但由于各线程运行时间并不一定与之前完全一致,造成重新运行的几个线程的先后顺序发生变化.

测试目的:测试由于回滚操作使得用户操作顺序不可预知的情况下所带来的一致性问题.

测试效果:回滚操作后,用户 3 和用户 4 读写顺序发生颠倒,所以用户 3 读出的是老值而不是用户 4 写入的新值.VMSRS 恢复效果如图 12 所示,用户 4 在抑制监控模式开启后在运行中对一些数据进行了更改,在它之后读取这些数据的用户 3 就可以读出用户 4 写入的数据.

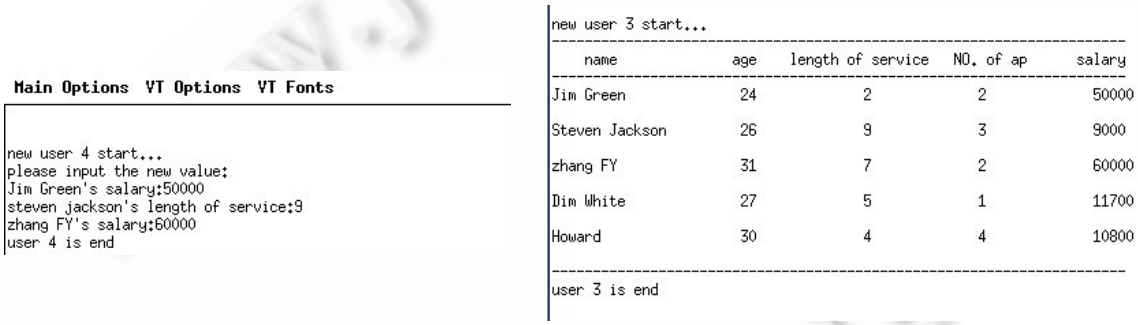


Fig.12 VMSRS recovery to ensure data's consistency

图 12 VMSRS 恢复方式保证数据一致性

测试分析:造成用户不一致的原因是用户执行某步顺序的颠倒在回滚中很常见,因为即使加读写锁,也只能控制同时对某个数据的读写,而不能保证两个用户的顺序谁先谁后,这都是在执行中确定回滚后,由于各种原因,很有可能导致本来先写后读或先读后写的几个用户顺序发生颠倒,这是加锁无法解决的,回滚检查点策略在这种情况下很难保证数据一致性.而 VMSRS 恢复方式,由于不进行回滚操作,所以保证了不会发生读写操作颠倒等回滚可能带来的一致性问题.

(3) 测试场景 3:一个用户对某数据进行修改操作,这需要进行相关元数据更改;另一用户抢占 CPU 的控制权,并且对这个刚刚修改的数据进行访问.

测试目的:测试由于数据元数据非原子更新所带来的一致性问题.

测试效果:在老式 SRS 中,正常用户 4 虽然写入了新的数据,但用户 3 读到的还是老数据.VMSRS 的恢复效果同样如图 13 所示,用户 3 读出的值是用户 4 写入的、逻辑正确的值.

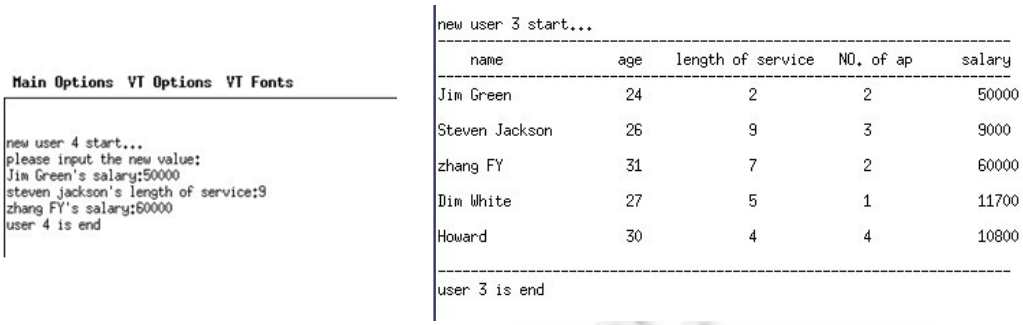


Fig.13 VMSRS recovery to ensure consistent data metadata

图 13 VMSRS 恢复方式保证数据元数据一致

测试分析:老式 SRS 无法很好地解决数据元数据一致性问题,用户 4 写入数据时用户 3 发生了读操作,这时刚好造成了用户 3 读到的数据虽然是正确的,可是由于系统还没来得及更新用户 4 写入数据的元数据,所以造成了恢复系统判断用户 3 读到的值是脏数据,所以继续用旧值恢复该数据,这样的结果是,不但用户 3 读出了老数据,而且用户数据区的数据也被错误地重新恢复,用户 4 写入的新数据被老数据重新覆盖.这种情况很容易发生在更新用户进程时间片刚好更新完,数据没来得及更新元数据时发生.而对于 VMSRS 来说,凡是与恢复有关的操作都会陷入虚拟机中,对数据和元数据的更新操作也不例外,这样,当虚拟机内部对恢复数据和相关元数据进行更新时,其他用户是处于停滞状态的,只有切换回客户操作系统,才会继续按多线程调度顺序运行,这就从根本上保证了更新操作的原子性.

关于一致性测试整体分析评价:VMSRS 恢复系统相对于之前的恢复系统的主要优势就是保证用户和数据状态的一致性.传统的回滚检查点恢复系统在用户一致性问题上或多或少存在缺陷,而新型的服务器程序恢复方式 SRS 则存在着数据元数据的不一致问题,以及错误发生后处理期间的用户数据不一致问题,这些问题都可以通过虚拟机的方式解决.这是因为当某用户陷入虚拟机处理时,CPU 的执行权力暂时就交给虚拟机了,这时操作系统和其上运行的程序都无法运行,这就保证了在虚拟机中处理期间的操作是原子的、不分割的.由以上实验可以看出,基于虚拟机的 VMSRS 系统与以前的系统相比,保证了数据用户数据的一致性,并且同时保证了数据元数据更新操作的原子性,在解决一致性问题上表现得比较优秀.

5.2 VMSRS基本性能测试与评价

测试数据集:选择一些正常的程序以及一些读写操作密集的程序.

测试结果和评价:如图 14 所示,在正常时刻,即没有用户线程出错,或者出错线程已经结束后,这种情况下,本恢复系统对用户的性能影响在 5%左右.影响之所以小是因为在正常情况下,恢复系统并不做特殊工作,只判断是否处在正常情况下,所以对整个系统的影响很小.对于赋值操作,大约有 10%左右的影响,这是因为赋值操作需要对读写两端进行比较.因为本系统提供函数库操作,所以可以由程序员掌握是否需要使用恢复函数来代替基本操作,而一个服务器程序的大部分读写赋值操作是不需要监控的,所以性能可以进一步提升,精心按照系统优化的程序甚至可以将正常情况下的影响降低到 3%.图 14 还显示了在出错后,其中读密集程序和写密集程序两条曲线是表示出错后在抑制监控模式下,恢复系统对赋值密集型应用和读写密集型应用的影响,其中因为写操作需要检查写入端是否为共享内存,如果是,就需要进行恢复存储数据和元数据的更新操作,这种情况比读操作影响更大.而赋值操作因为需要读写两端的数据判断,所以影响要大许多.不过值得注意的是,抑制监控模式运行时间只是占整个系统运行时间的很小一部分,而且为了保证出错后正常用户数据不受影响,这些性能影响是可以容忍的.

测试结果分析:与老式 SRS 相比,在正常运行时性能略低,老式 SRS 正常运行的 overhead 大约是 5%左右[10],但是在出错恢复时的 Overhead 相当,都达到 2 倍左右.这是由于本恢复系统主要是基于虚拟机恢复的,所以虚拟

机的开销是一个很明显的影响性能指标.本文虚拟机设计采用的 SVM 硬件虚拟化模式能够最大限度地减少虚拟机的开销.另一方面,本文应用层监控工作是交给函数库实现的,这势必带来一定的开销,这可以通过其他方式避免,由于本文主要是研究实现在虚拟化技术下的多用户服务器程序恢复系统,充分发挥虚拟化的安全可靠性和操作原子隔离性等优势,所以在性能方面并未做过多优化和设计.与回滚检查点策略相比,高度优化的回滚系统也有 12%的 overhead,本系统正常运行时的性能影响不到 10%.

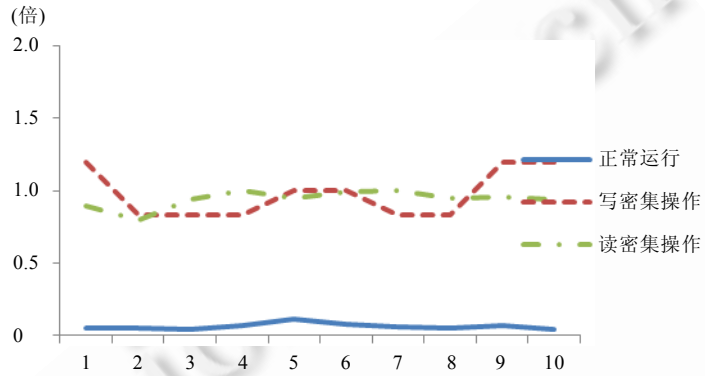


Fig.14 VMSRS recovery system overhead

图 14 VMSRS 恢复系统 overhead 图

5.3 VMSRS实际功能测试与评价

我们使用简单的网络购书系统模式来进行系统实际功能的测试.本测试主要是功能性的测试[10],测试程序较简单,测试数据集选择较为简单、易控制的,这是为了让测试更加精确且易于理解,绝不会影响到对真实功能的测试.这里为了对系统的功能测试更为准确,将原先手动的选择书名、输入购买册数等功能都编程自动化,这样做的原因是自动化后能够更好地测试多线程程序的并发工作情况,这样能够更好地测试多用户同时操作时的数据一致性和可靠性问题[11],更好地显示 VMSRS 在多线程程序上的优势.

测试数据集为:20 个用户.一个书目表,用户可以对其中书目的数量进行改动,也就是买书操作.

设计测试场景:20 个用户随机连接到服务器段,其中有些用户选择只是读取书目表浏览,有些用户则进行浏览并买书操作.测试开始后,其中 1,7 用户的服务器段程序出错,并且在出错时已经进行了买书操作,对数据进行了修改,但是还未完全结束修改就发生了崩溃.3,9,12,18 用户是正常用户,将进行买书操作,分别对数据进行修改.其他用户为正常的,只进行读数据操作的用户.

测试结果:经过测试,1,7 这两个用户出错后,其后紧接的正常用户所读的内容为旧的、正确的值,而 3,9,12,18 这 4 个用户用户进行数据更改后,后续用户所读的是他们修改过的值.没有出现数据不一致或数据恢复不完全(因为数据元数据不一致引起的)以及恢复数据出错等问题.在整个系统运行期间,没有感觉到出错后对性能的严重印象,同时各个用户获得的数据都是正确的,符合整个逻辑操作流程.

测试评价: 进行一次全局性的测试,目的是测试本系统可以很好地保证出错后所有用户的数据正确性,是否真正适合多用户多线程并发的真实服务器程序,测试结果良好,没有出现任何数据错误或不一致情况,并且性能影响较小.之前的理论分析和设计实现已经详细论证了 VMSRS 在多用户程序上可以保证恢复数据安全、用户数据一致性,保证多线程在出错后的顺利运行,这里只是对其进行了一个有效的验证.

5.4 VMSRS总体对比评价

表 2 给出了应用于多用户服务器系统上各种恢复模式的对比情况.

Table 2 Recovery system overall comparative evaluation

表 2 恢复系统总体对比评价

	VMSRS	Checkpoint/rollback	Rx	Assure	SRS	Recovery domain
恢复效果	良好	良好	一般	良好	良好	良好
恢复数据安全性	好	一般	一般	一般	一般	一般
对正常用户影响	小	/	较大	小	小	一般
适用于多线程并发	适用	/	否	一般	否	一般
数据一致性	保证	/	否	保证	否	否
操作原子性	保证	/	/	/	否	/
对性能影响	小	一般	小	一般	小	较小
用户数增加对性能影响	小	大	一般	较大	小	较大

由表 2 的对比可以看出,本文所设计的 VMSRS 适用于多用户服务器程序这种应用,同时,在综合表现方面比之前的一些恢复策略都要好.这是因为 VMSRS 是针对多用户服务器程序设计的,并且由于使用了虚拟机作为恢复系统的主体,从而获得了其他恢复系统所没有的一些优势.所以,综合这些原因,VMSRS 目前较适用于多用户服务器程序恢复系统.

6 结 论

随着多用户服务器系统在社会生活中的作用日趋重要,多用保证户服务器的稳定正确运行就成了一项很重要的工作,这就需要自恢复系统.但当今的服务器自恢复系统大多是应用回滚检查点恢复技术的,这种恢复在多用户应用中有其自身的缺陷.针对以上问题,本文结合虚拟化技术以及改进的多用户服务器恢复技术,以恢复多用户服务器内存错误、解决出错后所带来的一系列问题为目标,设计实现了基于虚拟机的多用户服务器恢复系统 VMSRS.可以得出,在虚拟机上进行应用于多用户的服务器程序恢复系统研究设计实现是很有意义的.不但可以充分发挥虚拟机和恢复系统相结合的优势,而且虚拟机越来越受到系统设计者和容错计算机系统研究者的重视,本文的研究工作也为将来的系统恢复技术提供了很好的参考和探索.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是解丰涛、谈博、钟云和孙伟同学表示感谢.

References:

[1] Rinard M. What to do when things go wrong: Recovery in complex (computer) systems. In: Proc. of the 11th Annual Int'l Conf. on Aspect-Oriented Software Development Companion. New York: ACM, 2012. 1–2. [doi: 10.1145/2162110.2162112]

[2] Nagarajan V, Jeffrey D, Gupta R. Self-Recovery in server programs. In: Proc. of the ISMM. New York: ACM, 2009. 49–58. [doi: 10.1145/1542431.1542439]

[3] Lenharth A, Adve V, King ST. Recovery domains: An organizing principle for recoverable operating systems. In: Proc. of the ASPLOS. New York: ACM, 2009. 49–60. [doi: 10.1145/1508244.1508251]

[4] Sidiroglou S, Laadan O, Perez CR, Viennot N, Nieh J, Keromytis AD. ASSURE: Automatic software self-healing using rescue points. In: Proc. of the ASPLOS. New York: ACM, 2009. 37–48. [doi: 10.1145/1508244.1508250]

[5] Wang AJ, Iyer M, Dutta R, Rouskas RN, Baldine I. Network virtualization: Technologies, perspectives, and frontiers. Journal of Lightwave Technology, 2013,31(4):523–537. [doi: 10.1109/JLT.2012.2213796]

[6] Bessho N, Dohi T. Comparing checkpoint and rollback recovery schemes in a cluster system. In: Proc. of the the 12th Int'l Conf. on Algorithms and Architectures for Parallel Processing (ICA3PP 2012), Volume Part I. LNCS 7439, Berlin, Heidelberg: Springer-Verlag, 2012. 531–545. [doi: 10.1007/978-3-642-33078-0_38]

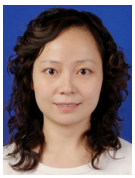
[7] Azimi R, Tam DK, Soares L, Stumm M. Enhancing operating system support for multicore processors by using hardware performance monitoring. ACM SIGOPS Operating Systems Review, 2009,43(2):56–65. [doi: 10.1145/1531793.1531803]

[8] Kapil D, Pilli ES, Joshi RC. Live virtual machine migration techniques: Survey and research challenges. In: Proc. of 2013 the 3rd IEEE Int'l Advance Computing Conf. (IACC). Washington: IEEE, 2013. 963–969. [doi: 10.1109/IAdCC.2013.6514357]

[9] Ye KJ, Jiang XH, Huang DW, Chen JH, Wang B. Live migration of multiple virtual machines with resource reservation in cloud computing environments. In: Proc. of 2011 IEEE the 4th Int'l Conf. on Cloud Computing. Washington: IEEE, 2011. 267–274. [doi: 10.1109/CLOUD.2011.69]

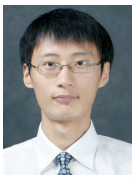
[10] Tomic O, Luciano G, Nilsen A, Hyldig G, Lorensen K, Næs T. Analysing sensory panel performance in a proficiency test using the PanelCheck software. European Food Research and Technology, 2009,230(3):497–511. [doi: 10.1007/s00217-009-1185-y]

[11] Betta G, Capriglione D, Pietrosanto A. A statistical approach for improving the performance of a testing methodology for measurement software. IEEE Trans. on Instrumentation and Measurement, 2008,57(6):1118–1126. [doi: 10.1109/TIM.2007.915143]



史楠(1975－),女,河南郑州人,博士,副教授,CCF 会员,主要研究领域为系统软件,系统安全,虚拟化.

齐勇(1957－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为分布式系统,操作系统,云计算.



冯雨声(1985－),男,工程师,主要研究领域为系统软件,航天控制,单机设计.

孙伟(1988－),女,硕士生,主要研究领域为系统软件,虚拟化.