ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.7, July 2009, pp.1986−2004
doi: 10.3724/SP.J.1001.2009.03549

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

增强计算系统可信赖性:融合虚拟化和 SOA*

林 闯, 孔祥震+, 周 寰

(清华大学 计算机科学与技术系,北京 100084)

Enhance the Dependability of Computing Systems: Integration of Virtualization and SOA

LIN Chuang, KONG Xiang-Zhen+, ZHOU Huan

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: E-mail: kxzmail@stu.xjtu.edu.cn, xiangzhen1985@gmail.com, http://www.tsinghua.edu.cn

Lin C, Kong XZ, Zhou H. Enhance the dependability of computing systems: Integration of virtualization and SOA. *Journal of Software*, 2009,20(7):1986−2004. http://www.jos.org.cn/1000-9825/3549.htm

Abstract: With the advance of computer hardware and software techniques and the continuous growth of application requirements, the computing systems, which have computers as their centers, have increasingly broadened the scope of application, while the complexity is also growing quickly. The demand for evaluating and improving the dependability of computing systems is more and more urgent. This paper gives the definition of computing systems' dependability, and a series of quantitative indicators are presented to evaluate the dependability. At the same time, the threats to system dependability are classified and analyzed in details. Since the traditional methods are incapable of dealing with the diverse dependability problem faced by the increasingly complex systems, people are constantly searching for new techniques. In this context, the virtualization technique comes to its renaissance, and is rapidly becoming a major research focus in recent years. In this paper, the existing research results on applying virtualization to enhance the dependability of computing systems are summed up, and the main characteristics and mechanisms of virtualization on enhancing dependability are introduced. But due to the restrictions of the existing computing systems architecture, the superiority of virtualization can not work fully. Service Oriented Architecture (SOA) meets the requirements of virtualization well for the loosely coupled, platform independence characteristics. Therefore, at the last part of this paper, a framework which integrates SOA and virtualization is proposed to enhance the dependability of computing systems, which is called Service Oriented Virtualization (SOV). This paper analyzes how this system framework can enhance the system dependability by mechanisms of virtualization and architecture superiority, when faced with many kinds of dependability threats.

Key words: computing system; dependability; virtualization; SOA (service oriented architecture); service-oriented virtualization

摘 要: 随着计算机软、硬件技术的不断进步和应用需求的日益增长,以计算机为中心的计算系统的应用范围越

* Supported by the National Natural Science Foundation of China under Grant No.60673187 (国家自然科学基金); the National Basic Research Program of China under Grant No.2003CB314804 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z419 (国家高技术研究发展计划(863)); the NOKIA Postgraduate Research Innovation Foundation (诺基亚研究生科研创新基金)

Received 2008-06-09; Accepted 2008-12-29

来越广,其复杂程度也在迅速提高,人们对如何评估和提高计算系统的可信赖性的需求日益迫切.首先给出了计算系统的可信赖性的定义,并系统地定义了一整套量化评价指标;同时,对计算系统面临的各种可信赖性威胁进行了详细的归类分析.传统的方法难以应对复杂系统面临的各种可信赖性问题,人们仍在不断地寻求新的技术.虚拟化技术在这种应用背景下走向复兴,成为一大研究热点.介绍了已有的虚拟化技术在增强系统可信赖性上相关的研究成果,并且总结了虚拟化技术在增强系统可信赖性方面的各种特性和机制.然而由于现有的计算系统体系结构的限制,难以将虚拟化技术在增强系统可信赖性方面的优势充分地发挥出来.面向服务的体系结构(service oriented architecture,简称 SOA)以其松散耦合、平台无关性等特点很好地适应了虚拟化技术的需求.因此,最后将 SOA 和虚拟化技术相结合,提出了一种增强计算系统可信赖性的系统架构,即面向服务的虚拟化 SOV(service oriented virtualization),并且分析了 SOV 系统如何在遭受各种可信赖性威胁时,运用体系结构优势和虚拟化技术的各种机制保证系统可信赖性.

关键词: 计算系统;可信赖性;虚拟化技术;SOA(service oriented architecture);面向服务的虚拟化

中图法分类号: TP302 **文献标识码**: A

系统可信赖性是衡量系统完成任务能力的重要指标,早在二战前后就开始受到重视.经过几十年来的发展,逐渐成为一门以定性与定量分析、控制、评估和改善系统的可信赖性的学科.如今,不仅在军事领域,机械、电子、汽车等许多行业和部门都对系统可信赖性有着迫切的要求.尤其是随着计算机技术的普及、网络技术的迅猛发展,计算机系统的硬件和软件的规模、复杂性空前地增长.嵌入式计算机技术的发展,更使得计算机无处不在,在许多关键场合发挥着至关重要的作用,如汽车电子、航空电子、工业控制等.我们将由计算机硬件、软件,有时还包括网络互连设施等共同组成的一个整体称为计算系统(文中以下的“系统”如不作特殊说明,一般指计算系统).系统面临的环境越来越复杂,如各种人为和非人为的威胁、嵌入式计算机系统面临的恶劣的工作环境等,都对系统的可信赖性提出更高的要求.因此,计算系统的可信赖性成为一个研究热点.

随着计算系统的复杂性和对可信赖性的要求越来越高,人们不断寻求新方法和新技术以提高系统的可信赖性的脚步一直没有停止.近些年来,一些研究人员开始借助于虚拟化技术.虚拟化技术是在 20 世纪 70 年代为了使多个用户共享昂贵的计算机硬件资源而产生[1,2].随着硬件资源成本的降低,虚拟化技术沉寂了一段时间.但随着人们发现虚拟化技术在增强系统可信赖性方面的优势,虚拟化技术在 20 世纪 90 年代中期重新走向复兴[3].直至目前,虚拟化技术已经成为一大研究热点,国际著名咨询机构 Gartner 公布的未来 IT“十大战略性技术”[4]中,虚拟化名列其中.

如何更好地设计系统架构,也是保证系统的可信赖性的一种手段.传统的 C/S 模式很容易存在单点失效等脆弱性;而分布式架构又存在着构建复杂、难以管理等问题.研究人员也在不断探索新的体系架构风格.于是,面向服务的体系架构 SOA(service-oriented architecture)应运而生.SOA 最初于 1996 年提出,其最初目的是为了改变当时的 C/S 模型,使得系统构架更加容易改变和重新部署.然而,由于当时的技术条件受限等原因,SOA 并没有受到人们的重视.在 2003 年前后,随着 Web Service 的盛行,SOA 重新回到了人们的视野中,并且受到了如 IBM,Intel,Microsoft,HP 和 Sun 等主要 IT 公司的关注,而且也应用于一些政府机构部门的系统,如美国国防部(Department of Defense,简称 DOD)的 C2 系统[5]等.现在,越来越多的计算系统正在或即将应用 SOA 的体系结构.

虚拟化技术通过资源抽象实现了软件与底层硬件平台的解耦合,使得客户操作系统可以在不同的硬件平台之间迁移.这种松散耦合正是 SOA 的一大特征和需求;而 SOA 可以通过服务封装和标准化实现应用服务与操作系统平台的解耦合.因此,本文结合虚拟化和 SOA 提出一种面向服务的虚拟化框架 SOV.通过应用服务、操作系统以及底层硬件平台的松耦合,提供隔离性和灵活性,避免了故障扩散.同时,应用服务和操作系统在平台之间的迁移、替换,提高了系统的容错、容侵的能力.

本文第 1 节论述系统可信赖性的定义,可信赖性是一个综合概念,它包含若干个属性,本文在这一节将分别进行介绍,并且给出它们的量化计算方法.第 2 节主要介绍虚拟化技术并着重分析虚拟化技术提高系统可信赖性的 5 种主要机制.第 3 节介绍虚拟化技术在增强系统可信赖性上的已有研究,并对它们的特点和不足进行分

析.第 4 节对 SOA 的概念、层次模型以及关键特性进行介绍.第 5 节分析 SOA 的可信赖性机制,并对 SOA 与传统体系结构相比在可信赖性上的优势进行论述.第 6 节提出一种新的系统可信赖性解决框架 SOV,并论述应用 SOV 的系统如何保证可信赖性.最后,第 7 节对全文进行总结,并且展望下一步的研究工作.

1 系统可信赖性及其威胁

1.1 可信赖性的定义

20 世纪 50 年代,可信赖性(dependability)的概念被首次提出,可信赖性最初被定义为在一定条件下,系统在一定的时间区间内完成一定功能的能力.可信赖性作为一项综合指标,最初包含可靠性、可用性和可维护性等属性.近几十年间,随着系统功能和性能的不断提升,带来了系统复杂性的剧增,可信赖性的内涵和外延也在不断地被补充和发展.不同组织和机构的标准中对可信赖性的定义也不尽相同,未达成统一的共识.2004 年,Algirdas 等人对可信赖性作了系统的总结[6],认为可信赖性是评价一个系统避免发生严重的服务失效或者过于频繁超出可接受范围的故障的能力,并且对可信赖性包含的属性作了进一步的细分.

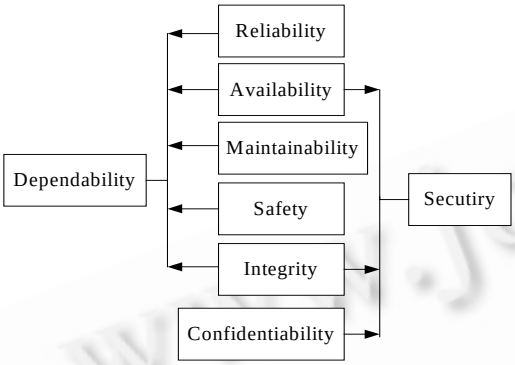


Fig.1 Main attributes of dependability[6]

图 1 系统可信赖性包含的主要属性[6]

关于安全性与可信赖性之间的关系,可信赖性最早被提出来时主要针对硬件系统,并不包含安全性属性.随着计算系统的发展尤其是网络的兴起,安全性受到重视.安全性最初被当作可信赖性的一个属性而引入可信赖性的范畴中.但是,安全性侧重于描述防止非授权的访问和修改的能力,它本身就是一个组合概念,包括传输保密性(confidentiality)、完整性等.而且有些安全性属性并不是系统可信赖性所都要考虑的,例如在考察一辆坦克的火控系统可信赖性时,并不需要考虑传输保密性.因此,在最新的相关研究中都是把安全性从可信赖性范畴中抽出来成为一个和可信赖性并列的范畴[6,7],如图 1 所示.因此,虽然安全性与可信赖性之间有些属性交叉,但并不存在所属关系,两个概念的侧重点不同.

本文主要针对系统可信赖性进行量化分析.可信赖性是一个综合性的概念,包含多种不同的属性,用以描述系统不同侧面的特性[6].包括可靠性(reliability)、可用性(availability)、可维护性(maintainability)、保险性(safety)和完整性(integrity).部分属性指标(如可靠性、可用性)在近几十年的研究中已经有了一些广为接受的量化计算方法[8].在本文的研究中,我们采纳一些经典的结论.另外,对目前尚未统一定义的指标(如可维护性、保险性、完整性)的量化计算方法进行了定义和说明.

1.2 可信赖性的量化计算

可靠性[9−12]:定义为系统在一个特定的时间区间内,能够持续正常工作的概率.设 X 为代表系统正常工作的时间的随机变量,系统的可靠性 $R(t)$,即系统在[0,t]时间区间内正常工作的概率为

$$R(t)=P\{X>t\} \tag{1}$$

可用性[9−11,13]:分可修复系统和不可修复系统两种情况考虑.对于不可修复系统,系统可用性可以认为与系统可靠性相等;对于可修复系统,可用性定义为在规定的条件下,当任务需要时系统处于可使用状态的概率.也就是在一个特定的时间内系统能够正常工作的时间比率.按照时间关系,可用性可以分为瞬时可用性、稳态可用性及固有可用性.瞬时可用性定义为系统在某一时刻处于正常工作状态的概率.设 $Y(t)$为 t 时刻系统的工作状态, S_N 是系统正常工作状态的集合,则瞬时可用性 $A_I(t)$可表示为

$$A_I(t)=P\{Y(t)\in S_N\} \tag{2}$$

考虑稳定状态下系统处于安全状态的概率,即稳态可用度.稳态可用度与瞬时可用度(性)的关系可用下式

来表示:

$$A_S = \lim_{t\to\infty} A_I(t) = \lim_{t\to\infty} \frac{\int_0^t A_I(x)\mathrm{d}x}{t} \tag{3}$$

实际工程应用中,往往更关心系统的固有可用度.固有可用度只考虑系统正常工作时间和故障修复时间,而不考虑其他诸如行政延误时间和预防性维修时间等.因此,固有可用度可以简单用下面的公式计算:

$$A = \frac{MTBF}{MTBF + MTTR} \tag{4}$$

其中,*MTBF* 代表平均故障间隔时间,*MTTR* 是平均故障修复时间.可见,固有可用度完全取决于系统固有的结构性能.

另外需要补充的是,这里所指的可用状态是指能够满足系统功能和性能规格需求的状态.对于有些容错系统,在部分构件发生故障后可能会提供降级的服务,只要能够满足系统规格需求,就认为处于可用状态(此时的系统时间归入 *MTBF*),反之,认为系统处于不可用状态(系统时间归入 *MTTR*).

可维护性:描述系统调整、修复和容错的能力[6].可以简单地用平均故障修复时间 *MTTR* 来衡量,也可以用系统服务发生失效后在时间间隔 t 内被修复的概率来定义,设 Z 为表示系统处于服务失效状态的时间,则可维护性 $M(t)$为

$$M(t)=P\{Z\le t\} \tag{5}$$

保险性:定义为系统在运行生命周期内,不对用户和环境造成灾难性后果[6,14,15].我们可以用概率表示,设 S_F 表示系统运行状态中出现安全事故的状态集合,则保险性 S 可表示为

$$S = \sum_{i\notin S_F} \pi_i \tag{6}$$

其中,π_i 为系统处于状态 i 时的稳态概率.

完整性:定义为抵御不适当的系统修改的能力[6].如果可以识别不适当系统修改时系统所处的状态,就可以对系统完整性的量化进行计算.当然,很多时候对系统完整性遭到破坏的所有状态的识别是困难的.假设系统完整性遭到破坏的状态集合为 S_I,则我们可以用与保险性类似的方法来计算完整性 I:

$$I = \sum_{i\notin S_I} \pi_i \tag{7}$$

从求解方法的角度来看,上面定义的可信赖性属性指标大致可以分为 3 类:第 1 类是稳态可信赖性指标,它们的求解需要结合系统的稳态概率分布,如保险性和完整性;第 2 类是瞬态可信赖性指标,其求解需要结合系统的瞬态概率,如可靠性(因为可靠性衡量的是持续正常工作的能力,$t\to\infty$时,可靠性趋向于 0,考虑稳态没有意义);第 3 类是混合可信赖性指标,如可维护性和可用性(存在瞬态和稳态下的两种定义).当然,所有指标都还可以通过对系统进行仿真模拟来求得数值解,仿真的数值解可以用来和模型求解结果相互验证.图 2 给出了可信赖性指标计算方法之间的关系.

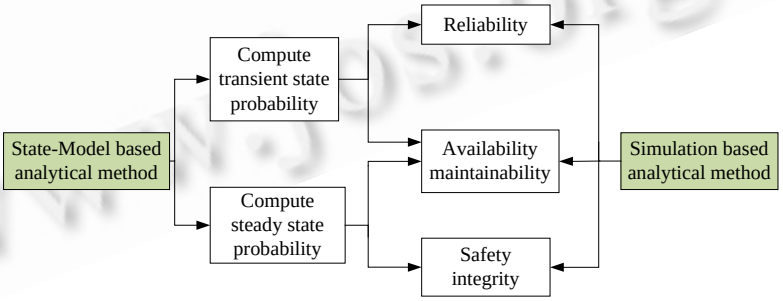


Fig.2 Computing methods of dependability indexes

图 2 系统可信赖性指标计算方法

1.3 系统可信赖性面临的威胁

系统可信赖性威胁可能由硬件问题、软件 bug 等非人为因素引发,还可能是由于人为误操作甚至是恶意的攻击造成.另外,有些嵌入式计算机系统所处的恶劣的环境也给系统的可信赖性带来隐患.为了系统地研究这些可信赖性的威胁,文献[16]根据可信赖性各种威胁之间的相互关系和对系统的影响程度,将这些可信赖性的威胁进行分类.文献[6]又对它们的分类和相互转换关系作了进一步的补充和论述.本文依据已有的研究成果,将系统的可信赖性威胁分为缺陷(fault)、错误(error)和故障(failure).

缺陷:是指在系统的设计、开发、使用各个阶段中存在的不足和隐患,以及各种可能引发错误的外在因素(包括外部实体的恶意攻击).缺陷可以进一步细分为开发缺陷(development fault)、物理缺陷(physical fault)和交互缺陷(interaction fault)[6,16].开发缺陷包括所有在系统开发过程中引入的系统错误隐患;物理缺陷包括可能影响到系统硬件的所有系统内部缺陷;交互缺陷包括所有可能引发系统错误的外在因素和隐患.

错误:是指系统背离正确的状态,这种背离可能导致系统发生故障,但也可能由于系统的容错、纠错机制而避免引起故障[6,16].当一个错误发生时,会激活错误消息和错误信号,称这种错误为已检测错误(detected error);如果一个错误发生但没有被检测到,称其为潜伏错误(latent error).

故障:是指系统的某一项功能无法按规格说明的要求完成,或偏离正常的状态并引发非预期的后果.故障反映的是系统功能无法实现和性能不可接受地下降的一种严重结果.故障可以分为开发故障(development failure)、服务故障(service failure)和可信赖性故障(dependability failure)[6].开发故障是指在系统开发过程中,系统的某些构件或功能模块发生问题,往往导致开发过程被迫回卷(roll back)甚至终止;服务故障是指最后交付的系统的某些服务背离了预期的正确服务;可信赖性故障是指在系统的使用阶段,由于严重的服务故障频繁发生,导致系统发生不可接受的性能降级或功能背离,无法满足系统预定的要求.

缺陷、错误、故障相互之间的因果和转换关系如图 3 所示.

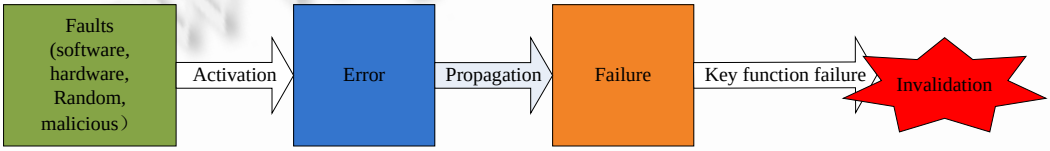


Fig.3 Transition process of dependability threats

图 3 可信赖性威胁转换过程

由系统软、硬件或者各种人为和非人为因素造成的系统缺陷,由于某些因素(如模块调用、恶意攻击等),有些缺陷被激活,从而诱发产生错误.有些错误可能并不会波及系统对外提供正常的服务,但一旦错误传播扩散,使得系统服务的关键部分产生错误,就会引起系统某些功能模块的故障.系统故障若没有及时修复和排除,往往会引起新的缺陷,这个过程也就是文献[6,16]中所描述的可信赖性威胁转换链.

但是,一旦系统的主要功能模块产生故障,特别是对于不可修复的系统,这将使得系统无法完成关键任务和功能,导致了系统的失效(invalidation),迭代终止.

1.4 增强系统可信赖性的现有手段及其不足

近几十年来,人们在不断探索增强系统可信赖性的方法.已有的手段和技术主要在错误避免、查错、纠错、容错等几个方面.但是,已有的增强系统可信赖性的手段,不论是在错误避免、查错、纠错还是容错等方面采取的措施都存在着一些不足.例如:查错、纠错技术准确率难以保证;过多地依赖操作人员的参与,自动化程度低;理论模型和形式化模型的错误避免手段往往代价非常高,而且难以保证模型的完备性;容错机制中,物理的冗余备份手段带来了高昂的硬件资源开销,并且调度和管理复杂.这些不足促使研究人员不断地去寻找新的技术和机制来增强系统可信赖性.20 世纪 90 年代中期,人们发现了虚拟化技术在增强系统可信赖性和降低成本上的优势,使得虚拟化技术在经过十几年的沉寂之后,由于人们对增强系统可信赖性的迫切需要而重新走向复兴.

2 应用虚拟化技术增强系统可信赖性

2.1 虚拟化技术概述

仅在计算机科学与技术领域,“虚拟”这个词就有广泛的应用,例如虚拟现实、虚拟网络、虚拟存储等等.本文研究的主要对象是计算机系统虚拟化(system virtualization,简称虚拟化).

虚拟化发展到今天,各种不同的技术不断涌现[17−19],同时也出现了一批虚拟化的产品.按照对底层硬件的虚拟化的程度可以将系统虚拟化技术分成硬件仿真器(hardware emulator)、全虚拟化(full virtualization)、半虚拟化(para-virtualization)以及操作系统级虚拟化(OS-level virtualization).硬件仿真器是基于仿真技术通过完全模拟底层硬件来实现虚拟化,代表的产品有 Bochs[20],QEMU[21]等;全虚拟化使用 VMM(又称为 Hypervisor)实现客户操作系统和原始硬件的解耦,代表产品有著名的 VMware 公司的技术产品[22]和 IBM 推出的 z/VM[23]等;半虚拟化技术是为了解决全虚拟化技术性能下降的缺陷而提出来的一种新的虚拟化技术[24],在半虚拟化系统中,客户操作系统可以感知到底层虚拟硬件,典型的半虚拟化系统有 Denali[24],Xen[25]等;操作系统级的虚拟化在操作系统之上虚拟多个服务器,支持在单个操作系统上简单隔离每一个虚拟服务器.严格来讲,操作系统级虚拟化并不真正虚拟底层硬件,而只是一种进程的模拟技术,典型代表是 Linux-Vserver[26],OpenVZ,Zones[27].

2.2 虚拟化技术增强系统可信赖性的主要机制

虚拟化技术之所以能够增强系统可信赖性,我们总结主要有以下几种机制,分别是服务器整合(server consolidation)、隔离(isolation)、动态迁移(live migration)、多层备份机制(multi-layer backup)以及检查点/重启机制(checkpoint/restart).

(1) 服务器整合:将原来独立的服务器通过 VMM 合并到同一个物理服务器上,如图 5 所示.首先,服务器整合增加了功能部件的复用度,降低了硬件的成本,从而提高了系统的可负担性;其次,可以提高服务器的利用率.目前,服务器功能强大但利用率很低,通常只有 10%~15%.利用虚拟化技术的服务器整合机制可以更充分地利用服务器中的闲置资源,提高服务器的利用率;最后,通过服务器整合有利于实现负载均衡[28].

(2) 隔离机制:隔离有两层含义:一方面,由于 VMM 层的引入,实现了 OS 和底层硬件的解耦;另一方面,实现了各 Guest OS 之间的隔离,每一个应用服务运行在自己的 Guest OS 之上,从而实现各个应用服务之间的独立,如图 5 所示.首先,由于各个应用服务之间的隔离,使得一种应用服务在遭受攻击或者发生故障时不会波及到其他应用服务,提高了可靠性和可用性;其次,上层 Guest OS 和应用服务与底层硬件的解耦,使在不影响用户服务的前提下,实现应用服务在不同硬件平台之间迁移成为可能,同时也提高了可维护性.另外,每个应用服务运行在单独的 Guest OS 上,就可以针对每一个应用服务定制专用的操作系统,从而提高效率和节省软件开销.

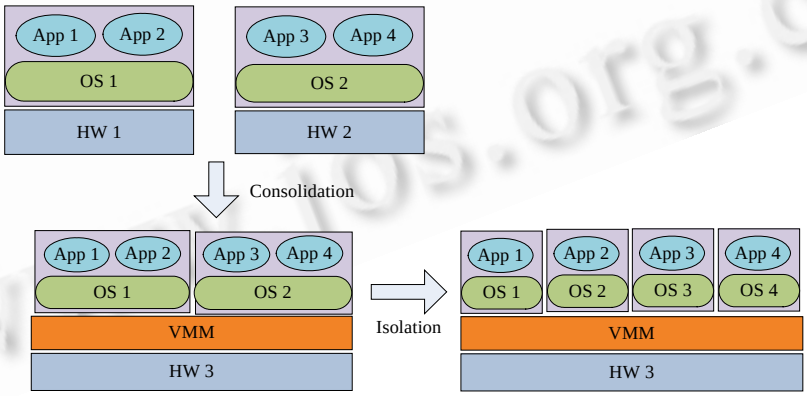


Fig.5 Server consolidation and isolation

图 5 服务器整合和隔离

(3) 动态迁移机制:或者称为热迁移(hot migration)机制,是指在不影响用户接受服务的情况下,实现应用服

务或整个 Guest OS 从一台硬件机器转移到另外一台机器上运行.这项技术使得系统硬件的维修或者替换不会影响用户的服务,极大地提高了系统的可用性和可维护性.典型的实现方式是记录 Guest OS 的执行状态,利用高速网络将虚拟机文件从原硬件机器传输到目的机器上,再恢复 Guest OS 的执行状态.由于传输迁移的过程在很短的时间内完成,另外采取一些手段可以使得这种迁移过程用户根本感觉不到,从而提高了系统的可用性.目前,动态迁移已经得到很多虚拟机软件的支持,如 VMware 的 VMotion 技术[29](如图 6 所示)和 Xen 上的 Live Migration 技术[30].

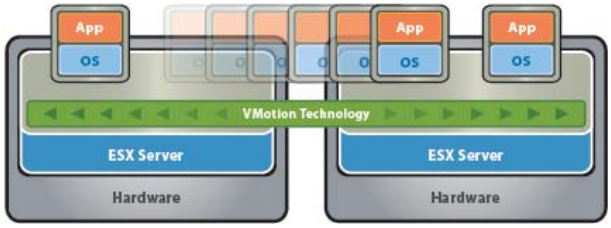


Fig.6 Live migration (Vmotion in VMware[28])

图 6 动态迁移机制(VMware 的 VMotion[28])

(4) 多层备份机制:备份机制是传统的增强系统可信赖性的手段.在虚拟化技术中,由于应用服务和 Guest OS 只不过是一个文件,这使得系统的备份更加容易,且成本极低.通常把应用服务和承载它的 Guest OS 捆绑在一起看成一个虚拟服务设备(virtual service appliance)[31],以虚拟服务设备为单位,使得备份变得十分容易且成本极低.另外,在虚拟机系统中,备份可以分成不同的级别,同一硬件平台上的不同虚拟服务设备之间可以形成备份,备份时可以采用全镜像备份(full-image backup)或者文件级备份(file-level backup)[32];同时,不同虚拟服务设备和硬件平台构成一个整体,可以对某些关键的服务进行备份,从而形成多层的备份机制.

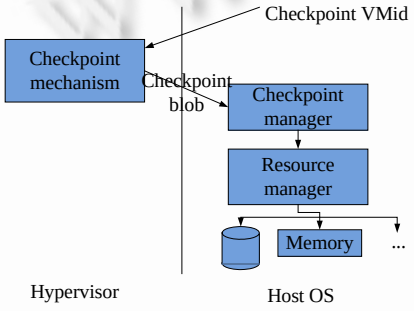


Fig.7 Implementation of checkpoint mechanism[33]

图 7 虚拟机的检查点机制的一种设计实现[33]

(5) 检查点/重启机制:由于运行着服务的虚拟操作系统一般以文件的形式存在于虚拟机当中,因此,虚拟操作系统可以停在运行的任何阶段,并保存当时的运行状态.当系统面临可信赖性威胁时,可以通过虚拟机的这种特性设置一个检查点,当威胁解除后,重启虚拟操作系统,导入运行状态,便可以恢复系统原来的运行.这种检查点/重启机制极大地提高了虚拟机系统的可靠性和容错特性,增强了系统的可信赖性.如何更好地实现和应用虚拟机的检查点/重启机制,也是目前的一个热点[33−35].一个典型的实现如图 7 所示.

以上是我们总结的虚拟化技术增强系统可信赖性的 5 种最基本的机制,至于一些其他技术,如安全日志和重放、入侵检测、操作系统行为约束等[36],归根到底可以看成隔离、动态迁移以及检查点等机制核技术的综合运用.

3 虚拟化技术增强系统可信赖性上的已有研究

3.1 已有研究

通过前面的分析可见,虚拟化技术在改善系统可信赖性等方面有着显著的优势.如何运用现有的虚拟化技术的机制和特性,并且进一步发掘虚拟化技术的潜在能力以进一步提高系统可信赖性,成为近年来研究的热点问题.

冗余备份是构造高容错可信赖性系统的主要技术之一,虚拟化技术的引入使得备份更加简单而且成本大为降低.Bressoud 和 Schneider[37]实现了一个在 VMM 级实现容错机制的主备份复制协议.这个协议解决了非确

定性事件发生时如何维持状态连续的问题.VMware 的 Double-Take[38]利用了基于硬件的实时同步复制应用数据,即将应用层的数据从 VMs 复制到一个单独的物理机上,以便于 VMs 可以及时地恢复为一个备用的状态,通过再次引入数据,可以避免造成的中断等故障.

由于虚拟机实现了上层操作系统和应用服务与底层硬件的解耦,使得上层遭受的攻击不会影响到硬件,因此可以将虚拟机用于检测和重放系统的攻击或者故障以便分析和预防.Joshi 等人将 VM 自省与 VM 重放结合在一起[39],在打补丁之前进行 VM 中的 OS 攻击性与应用层攻击性的活跃性对比.此项分析基于预期 vulnerability-specific,由补丁作者创造.补丁应用之后,同样的预期可以在 VM 正常运行期间应用,用来进行探测和消除攻击.Backtracker[40]可以用来确定在给定 host 基础上确定的 VM 中的应用问题,一个 Backtracker 的扩展[41]用来跟踪来自单个 host 的攻击.

隔离机制是虚拟化技术提高系统可信赖性的主要特性之一,Livewire[42]和 ReVirt[43]利用了虚拟机的隔离机制提高了系统的安全性能.VMM 提供了重构现有软件系统以增强安全的能力,同时也便于使用新的方法去构建安全系统.将有安全漏洞的某些功能移到虚拟机的外面,使它们与客体操作系统一起运行但又与之隔离,提供了相同的功能但具有更强的抗攻击能力.这方面的两个典型研究实例就是 Livewire 和 ReVirt.其中,ReVirt 将日志功能移出虚拟机,以实现安全的日志.将监控保险措施放在虚拟机之外,提供了一个有吸引力的方法去检疫网络,从而限制虚拟机只访问没有恶意和攻击行为的网.而 Livewire 提出的是借助于虚拟化技术解决系统入侵检测中的存在的缺陷:由于基于网络的入侵检测(network intrusion detection system,简称 NIDS)不能获取被监视机器的状态信息,而基于被监视主机的入侵检测(host intrusion detection system,简称 HIDS)不能有效地保护自己不受到攻击.因此,Livewire 将入侵检测移到被监视虚拟机的外面,借助 VMM 的支持,使之能够同时兼有 NIDS 和 HIDS 的优点.Livewire 的基本结构如图 8 所示.

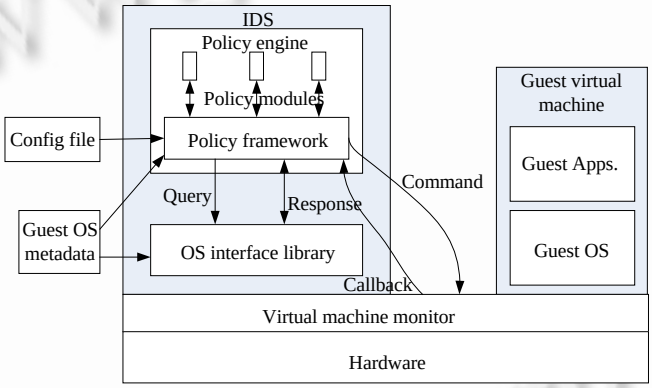


Fig.8 Architecture of Livewire[42]

图 8 Livewire 系统基本结构[42]

其中,入侵检测系统(intrusion detection system,简称 IDS)可以看作 VMM 上的一个虚拟机(VM),但较其他 VMs 有着特殊的优先权.IDS 的主要构件之一是操作系统接口库(OS interface library),它提供了该被监测 VM 的 OS 视图.操作系统接口库的实现是通过由 Guest OS 解释通过 VMM 返回的元数据(metadata)来获取的.另一个 IDS 的主要构件是策略引擎(policy engine),提供了一个实现通用策略(common policies)以及策略模块(policy modules)集合(用以实现入侵监测机制)的框架.VMM 层将 IDS 从被监测的 VM 中隔离出来.Terra[44]系统由斯坦福的 Garfinkel 等人提出,其基本体系结构如图 9 所示.Terra 使用一个可信虚拟机监控器来产生并支持各种单独的虚拟机.基于隔离机制,VMM 支持运行多个不同安全级别的软件栈.例如,用户桌面系统可以同时运行几个虚拟机.相对安全级较低的 Windows 虚拟机用于网络浏览,安全级稍高的、强化的 Linux 虚拟机进行日常工作,由高安全的操作系统和专用邮件客户端构成的虚拟机用于敏感内部邮件.VMM 的这些能力使得它很适合构建可信计算环境,如 Terra 系统.其主要优点在于采用了成熟的虚拟机技术,仿真不同的运行环境,因而能够满足不

同的安全需求.但该系统并不是独立的可信计算平台结构,而是将可信计算环境纳入到虚拟机架构中,实现比较复杂.此外,虚拟机也特别适合作为构件去构建高安全保障的系统.例如,美国国家安全管理局的 Net Top 体系结构使用 VMware 去隔离多个环境,每个环境有权使用不同安全分类的、分离的网络.

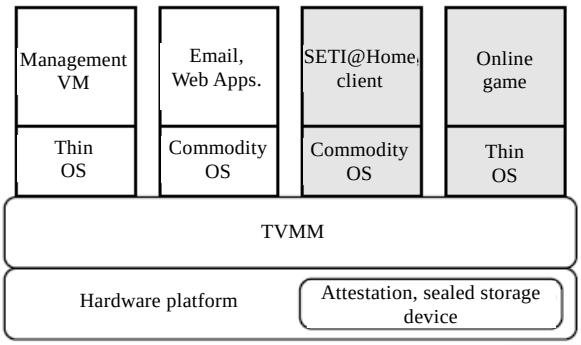


Fig.9 Architecture of Terra[44]
图 9 Terra 系统结构[44]

3.2 已有研究的不足及分析

近些年对虚拟化技术已有的研究,一方面集中在系统实现的简单性、系统的兼容性和性能上,以满足对应用系统进行整合的需求,提高系统的可负担性;另一方面则利用 VMM 提供的隔离、备份和动态迁移等机制进一步提高系统的可靠性、可用性等.但从如何有效地综合运用虚拟化的各种机制和特性、提高系统整体的可信赖性上来讲,已有的研究还存在一些不足,主要有:

(1) 片面性

现有的解决方案往往着眼于系统可信赖性的某一种侧面的属性,比如保险性、可靠性等等,缺乏对系统可信赖性的全面的考虑.往往收之桑榆、失之东隅,牺牲了系统其他方面的特性来片面追求某一方面特性.究其原因是因为没有全面分析一个系统要对外提供可信赖的服务所应具有的所有性质,对系统的可信赖性缺乏全面而清晰的认识.

(2) 缺乏定量的研究

上面综述的借助虚拟化技术来增强系统可信赖性某个特性的研究,大多停留在定性的判断上,缺乏一个定量的度量,难以得到令人信服的结论.其原因主要有两点:一方面是因为对系统的可信赖性和它所包含的各种属性缺乏定量的描述和分析,没有系统地提出一套定量的衡量指标;另一方面,对于虚拟化技术或者虚拟机缺乏数学理论模型来描述,因而也就难以对一些机制和算法进行定量的分析.

(3) 部分解耦

虚拟化技术实现了操作系统和系统硬件资源的解耦,现有的一些研究多是在这种 OS 和硬件资源松散耦合的基础上开展的,在隔离了底层硬件之后,研究如何应用虚拟化中某些特定技术来保证硬件上层 Guest OS 及应用程序的安全.但是,用户应用服务和 OS 软件平台仍然是紧耦合的,这使得用户应用服务过分依赖于 Guest OS 的可靠性.同时,用户应用服务固定在某个虚拟机中,无法实现调度以优化利用软件资源.如何实现用户服务和硬件平台与软件平台的解耦,使得用户应用服务在多个虚拟机之上和多个 Guest OS 之上有效地分配和调度,现有研究没有给出一个解决思路.

(4) 缺乏通用的解决框架,通用性和可扩展性差

现有的研究大多是针对某一个特定系统量身定做,零散地利用虚拟化技术的一些特性和机制来保证系统的某些安全和性能特性,而缺乏一种整体的解决框架.这样带来的结果就是缺乏通用性和可扩展性差.

上述关于现有研究和解决方案的这些缺点和不足,一个很重要的原因是没有适应虚拟化技术而提出一种新的系统架构[24],从而未能更好地发挥虚拟化技术的优势.传统的系统体系结构架构存在着一些自身的限制,难

以迎合和更好地发挥虚拟化技术的一些特性和机制.在改善系统体系架构上,不少研究人员也在不断地进行探索.面向服务的体系架构 SOA,由于其在平台无关、松散耦合和高扩展性等方面的优势,近些年受到了极大的关注.

4 面向服务的体系结构 SOA

4.1 SOA概述

SOA 由 Gartner 于 1996 年提出,其最初的目的是为了改变当时的 C/S 模型,使得系统构架更加容易改变和重新部署.然而,由于当时的技术条件限制等原因,SOA 并没有受到人们的广泛关注.在 2003 年前后,随着 Web Service 的盛行,SOA 重新回到了人们的视野中,并且受到了如 IBM,Intel,Microsoft,HP 和 Sun 等主要 IT 公司的关注,并且应用于一些政府机构部门的大型系统的构建中,如美国国防部[45]的 C2 系统,包括 NCES,JBMC2,FORCEnet,JBI,FCS 和 GIG-ES[46].

SOA 概念的核心是“服务”,包括 GGF(the global grid forum)[5],IBM[47],W3C[48],OASIA[49]等很多企业、组织都给出了其相应的服务定义.参考 W3C 给出的定义[48],结合本文中 SOA 的讨论范围,在这里将服务定义如下:

服务:服务是一个实体,它是可以完成一定任务的抽象资源,并可以由使用者通过交换信息进行调用,满足用户一定的需求.

在服务的基础上可以定义 SOA,OASIS 在 2005 年 5 月给出了一个 SOA 参考模型[49],用来“维持一个通用的对于 SOA 是什么的理解层面”.在这里,我们引用 GGF[5]关于 SOA 的定义:

SOA:用于搭建可靠的分布式系统的体系结构风格,它用服务的形式实现各种功能,并且强调松散的服务耦合.

时至今日,SOA 这种体系结构已经不再仅限于软件应用了,基于服务的硬件和网络项目都已经有实际应用,比如 Intel 的 SOI(service oriented networks)架构和 Cisco 的面向应用的网络(application oriented networks,简称 AON[50,51]).

4.2 SOA的层次模型

在文献[52]中提出了一个 SOA 的 9 层模型,各层分别为 Operational systems,Service component,Services,Business process,Consumer,Integration,Quality of service,Information architecture 和 Governance an policies(如图 10 所示).每层具体作用如下:

Operational systems:包括运行在 IT 的操作环境中的已存在的应用程序,如操作系统、数据库管理系统、CRM、ERP、商业智能(BI)等,使用已有的应用可以降低总系统的成本.

Service component:每个服务组件都是一个服务的实现,或者是一个服务的部分操作.这些组件保证服务的执行与它们的描述相一致,反映了服务的功能和 QoS,并且对用户屏蔽了繁琐的细节操作.

Service:这一层是整个 SOA 体系结构的核心层,包括由提供商提供给客户的各种服务.服务被看作是与业务相结合的 IT 功能或操作的抽象描述.服务是平台无关的,并且它们之间可以引用、编排形成组合服务.

Business process:服务在这一层被组合成实用的业务流程.SOA 强调服务在不同业务流程间的重用,一个业务流程可以由一些服务组合而成,并且这些服务可以用相同功能的其他服务所替代.按照业务流程的规则、策略和业务需求,可以用串行或并行的若干服务来表达业务逻辑.从上层往下层看,业务流程是由用户需求决定的,为了优化各种业务流程的实施,它们被分解成为可以重用的各种服务.从下层往上层看,当各种服务被实现之后,企业将它们整合成为有意义的业务上下文以满足用户的需求.

Consumer:处理与用户或其他程序的交互层.通过这一层,企业可以交付已有的 IT 功能给用户或其他程序.这样就可以快速地创建业务流程的前端或组合已有的应用.这一层多会采用成熟的前端访问机制(如 portal)以及标准的协议(如 WSRP),这样可以降低开发和部署的周期和简化对于原有访问的重用.

Integration:主要用于整合 Service components,Service 和 Business process 层,它负责各种服务请求正确到达

服务提供者,并返回响应信息.通常,SOA 系统的这一层会提供相邻层次之间的交互、引用和 QoS 的施行.

Quality of service:SOA 系统需要加强已有的计算机体系结构的 QoS,包括虚拟化技术、松散耦合、广泛使用 XML、联合服务的组合、异构基础结构、非中心化的服务级别协议等.这一层负责捕获、监视、记录并示意与服务质量需求不相符合的事件,监视其余层的运作,当侦测到与需求不相符合的事件时发送信号.同时,这一层确保 SOA 系统满足相应的可靠性、可用性、可管理性、可扩展性以及安全性的需求.

Information architecture:这一层确保企业包含影响关键因素的数据和信息结构,可以通过数据集市(data mart)和数据仓库(data warehouse)创建商务智能(business intelligence,简称 BI).

Governance and policies:本层覆盖管理业务操作的生命周期的所有方面,包括从手动管理到 WS-Policy 的所有策略;提供对服务基本协议的指导和策略,包括容量、性能、安全和监视.因此,这一层应用到所有其他层次上,从 QoS 和性能方面讲,它与 QoS 层的结合非常紧密.这一层的管理框架包括基于 QoS 和关键过程记录的服务级别的协议,为设计和调整系统的容量计划和性能管理策略.

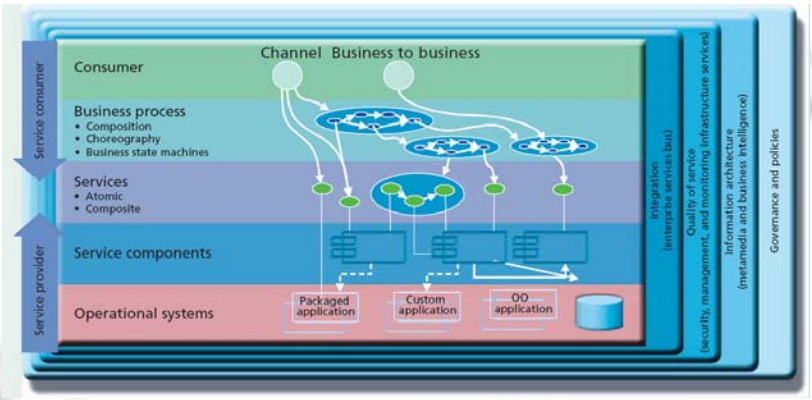


Fig.10 Logical layers in SOA[52]

图 10 SOA 中的逻辑层[52]

4.3 SOA的关键特征

作为一种新近兴起的体系结构,SOA 还在不断地发展和完善.关于 SOA 的特性,现在还不能给出一个完整的描述.然而在 SOA 的一些关键特征上,目前还存在一定的共识[53−56].目前公认的 SOA 与传统体系结构相比,所具有的关键特征主要有:

(1) SOA 提高了服务质量:SOA 继承了现有分布式系统的安全策略,并在此基础上增强了服务的安全行为,并且可以保护消息内容和单个服务.其松散耦合的性质也使得提高服务质量更容易实现.

(2) SOA 从根本上讲是自治的:在满足其基础的逻辑情况下,每个服务都是尽量独立且自包含的,服务内部独立维护和运行.各服务之间的消息也是自治的,并且提供足够的信息给接受这个消息的服务.SOA 服务内部封装完好,其内部细节和数据的复杂性都对外界隐藏.消息是服务调用的唯一途径,且不依赖环境上下文.

(3) SOA 的消息格式是开放的标准格式:消息通常遵循 SOAP,WSDL,XML 等协议.应用标准的格式,可以保证消息的自治性.服务只需要其他服务的描述,即可正确执行所需操作.增强了服务的逻辑和松耦合的特点.

(4) SOA 的平台无关性:由于消息的格式是标准的,所以各服务之间的交流与其所应用的平台无关,包括嵌入式系统在内的各种平台都可以作为服务组件的基础.

(5) SOA 的松耦合性:由于 SOA 的服务是自治的,其服务间传递的消息也是自治的,服务不必与特定系统和网络连接,服务之间的关系是松耦合性质.这一点保障了在一个服务不可用或失效的情况下,基本不影响其余服务的完成.

(6) SOA 的复用性:同一服务可以被多个服务、用户所调用,依据不同的服务策略,可以提高系统的利用率、服务质量并降低成本.服务的可重用性简化了服务的整合,并为以此产生新服务节省了大量的开发、部署时间.

(7) SOA 的可扩展性:针对各个服务的替换、升级、修复以及新服务的加入,都是非常简单的事情,只要服务有其相应的描述,并支持统一标准的消息传递即可.另外,SOA 这种体系结构可以基于现有的系统进行扩展,而不需要完全重新设计系统.这一特点也使得系统的设计和部署更加便捷、灵活.

(8) SOA 服务的动态发现和整合:在可扩展性中已经说明,只要新服务有相应的描述,并支持统一的消息格式,这个服务就可以加入到系统中,并被其余服务或用户所使用.保证了服务可以被动态地整合生成新的服务.

5 SOA 增强系统可信赖性

5.1 SOA的可信赖性机制

作为一种分布式系统,传统的增强可信赖性的办法如防火墙、入侵检测和安全监视等方法,仍然会应用于这个系统,但是在用户认证、授权以及数据加密等方面需要进行相应的修改,以适应 SOA 这种体系结构.其中典型的问题就是业务流程对不同企业或组织所提供的服务的访问,以及服务之间组合形成新服务后所带来的隐患.因为通常情况下,数据和操作不仅仅局限于同一企业之内,对于跨企业的服务使用权限以及用户信息的访问等都可能引发实际中的问题.具体可参考文献[57].增强 SOA 的可信赖机制有(但不局限于)以下几种:

创建副本机制(replication):这种机制用于保证冗余信息(包括软件和硬件组件)之间的一致性,有利于增强分布式系统的可靠性、容错性和可访问性.如果同样的数据存储在多个不同的设备上,则称为数据副本,这使得对不同副本间的访问就像对同一内容的访问,并且使得数据满足 ACID(atomicity,consistency,isolation,urability)的特性;如果相同的计算任务被执行多次,则称为计算副本.例如在电话交换系统中,当主交换机出现问题时,备用交换机可以在不影响用户使用的情况下接替其工作.创建副本机制与备份机制(backup)不同,备份保存一份相对较长时间的数据备份,而创建副本机制则频繁地使各设备上的信息同步并丢弃历史状态.在 SOA 系统中,数据副本可以保证不同用户访问同一个服务,或不同服务访问同一个服务组件时数据的 ACID 特性;而计算副本则可以保证一个服务在主服务器失效的情况下,副本服务器可以继续提供服务.

软件更新机制(software rejuvenation):软件更新机制是一种主动的错误防御机制,旨在清除系统中的一些中间状态,以避免将来可能发生的严重的系统崩溃事件.其操作是用合适的方法终止当前的应用程序,并且立刻以一个“干净”的中间状态将其重启.在 SOA 系统中,一个服务可以选择在空闲的时候进行更新操作,这样对服务的可用性影响很小,而对其可靠性、保险性和完整性上有一定的促进.

基于组件的重构机制(component-based reconfiguration):对于有缺陷的功能部件进行隔离,并替换为相同功能的部件来完成原部件的工作.对于一个 SOA 系统来说,可以根据服务或服务组件的状态(如繁忙程度或健康状况)进行业务流程或服务的重新配置.这个机制使得系统的可扩展性更强,这一机制增强了系统的可靠性、可用性以及完整性.

5.2 与传统架构相比,SOA在可信赖性上的优势

(1) 与以应用程序为中心的架构比较

传统的构建一个系统的方法往往是以应用程序为中心,总是先构建一个个的应用,每个应用可能有自己的功能层次、数据架构、安全架构等,然后,随着系统的发展需要在应用间进行整合,包括表示层整合、数据整合和流程整合等.传统的以应用程序为中心的体系结构,通常容易形成孤立的“数据化岛”和“自动机岛”[58],每个数据岛分别具有自己的对象含义或定义,每个自动机岛集中于有限的一组活动.这样就导致各应用程序之间的交互、合作十分困难.为了实现交互而造成它们之间的耦合十分紧密,重复使用性和可扩展性不强,这就使得在传统的以应用程序为中心的系统中,QoS 要求和可信赖性等得不到保障.SOA 和以应用程序为中心的架构不同,在 SOA 中将功能方面涉及的对象、数据、组件、业务流程、界面等从服务提供者和服务消费者角度进行层次化.与此同时,将安全架构、数据架构、集成架构、服务质量管理等应用共用的设施提取出来,形成不同的层次,为所有的服务所共有.SOA 与传统的以应用程序为中心的体系结构主要不同点列于表 1.

Table 1 Comparison between SOA and the application-central architecture[59]

表 1 以应用程序为中心的方法与 SOA 实现的比较[59]

Characteristic	Application-Centric architecture	SOA
Design and implementation	(1) Function oriented (2) Build to last (3) Long development cycles	(1) Coordination oriented (2) Build to change (3) Build and deployed incrementally
Resulting system	(1) Application silos (2) Tightly coupled (3) Object-Oriented interactions	(1) Enterprise solutions (2) Loosely coupled (3) Semantic message-oriented interactions

(2) SOA 与传统的 Client/Server 架构比较

分别从几个方面比较 SOA 和 C/S 架构:

① 安全性:在 Client/Server 结构中,服务器是资源、用户管理及数据运算的中心,只要对服务器的服务策略进行管理,即可保证整个系统的安全性.在 SOA 中,继承了 C/S 的管理、认证、访问机制,并在此基础上进一步加强对于消息内容的安全保障以及服务访问的管理,其安全性不逊于 C/S 结构;

② 可靠性与可用性:在 Client/Server 结构中,系统过于依赖服务器,一旦服务器失效,所有的服务都会失效,任何用户请求都无法得到满足,整个系统瘫痪.而在 SOA 中,由于 SOA 的松散耦合特点,个别服务的失效不影响其余服务的使用,增加了系统的可靠性和可用性;

③ 性能保证:在 Client/Server 结构中,当用户数量增加时,每个用户得到的系统资源减少,整个系统性能下降.而在 SOA 中,当针对某一服务的用户数量增加时,系统可以根据用户的需求,通过任务调度和减少其他服务的资源分配,尽量保证用户需求得到满足;

④ 可扩展性:在 Client/Server 结构中,服务器提供的服务与平台结合紧密,并且服务的增加、替换、维护、升级等一系列行为都十分繁琐,有时甚至要重新部署系统.而在 SOA 中,服务的增加、替换、维护、升级等行为十分简便,只需要支持统一的服务描述和消息接口即可.同时,服务是平台无关的,更大大提升了系统的可扩展性.

(3) SOA 与分布式系统

从前面的 SOA 的定义我们可以看出,SOA 从本质上来说是一种分布式系统.但 SOA 与传统分布式系统有所区别[60],主要表现在:SOA 是面向消息的,服务是形式化定义并且对内部进行抽象,服务接口独立于服务的实现;服务的描述基于计算机可处理的元数据/服务定义;服务是粗粒度的;服务实现和消息传递是平台无关的.

SOA 提供了一种与平台无关的、松散耦合、可扩展、可重用的体系架构组织方式,为前面提到的目前虚拟化技术构建可信赖性系统方面存在的一些不足提供了一条解决途径.当然,要更好地实现 SOA 架构的上述关键特征和优势,本身也需要底层基础设施(称为面向服务器的基础设施 SOI[61])的支持和配合,而虚拟化技术可以应用在 SOI 的构建当中以发挥虚拟化技术在保证系统可信赖性上面的优势.因此,在 SOA 架构之下融入虚拟化技术,为提高系统可信赖性带来了新的思路.基于这种思想,我们提出了一种构建高可信赖性系统的框架,称为面向服务的虚拟化 SOV(这里需要说明的是,SOV 这个术语,从事 SOA 产品测试的 iTKO 公司于 2007 年 12 月提出过,他们是作为 SOA 相关产品评估和测试的手段,含义与本文中有所不同).

6 增强可信赖性的系统框架——SOV

6.1 SOV系统框架

为了构建具有高可信赖性的系统,我们提出一种在 SOA 架构中融合虚拟化技术的方案 SOV.它结合了 SOA 和虚拟化技术在增强系统可信赖性上的优势和特点,系统的总体框架如图 11 所示.主要分成 3 个平面:业务处理平面、管理和监控平面、可信赖性控制平面.业务处理平面主要负责从用户应用需求交付到系统完成服务的整个业务流程;管理和监控平面提供了一个管理配置系统和监控系统运行状况接口;可信赖性控制平面实时监测和计算系统可信赖性的各项指标,对各种系统威胁启动相应的应急响应机制,保证系统的可信赖性.

图 11 中的业务处理平面可进一步展开成如图 12 所示.整个业务处理平面整体采用 SOA 的组织架构.底层对 SOA 进行支持的基础设施(称为面向服务的基础设施 SOI[61]),利用虚拟化技术构建,以更好地保证系统可信赖性.不同的硬件实体(如服务器等)通过高速介质相连接,每个硬件实体内部都可能包含多种硬件资源,共同构成硬件基础设施层(hardware infrastructure),硬件实体之间可以形成底层的备份.虚拟监控器 VMM 运行在各硬件实体之上,它们之间相互交互和配合形成虚拟基础设施层(virtual infrastructure).它一方面实现了上层操作系统(VOS 特指运行在虚拟机上的 OS)与底层硬件之间的解耦,同时实现了各个 VOS 之间的隔离.另外,VMM 之间的相互通信,实现了动态迁移和 VOS 之间的备份.VOS 中运行专门的应用程序承载上层交付的服务,为了减轻负载,提高效率,还可以对每一种服务定制专门的 VOS,解决通用 OS 功能浪费、效率不高的问题.关于 SOV 框架的详细说明见表 2.

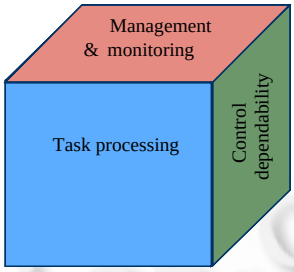


Fig.11 Total framework of SOV

图 11 SOV 总体框架

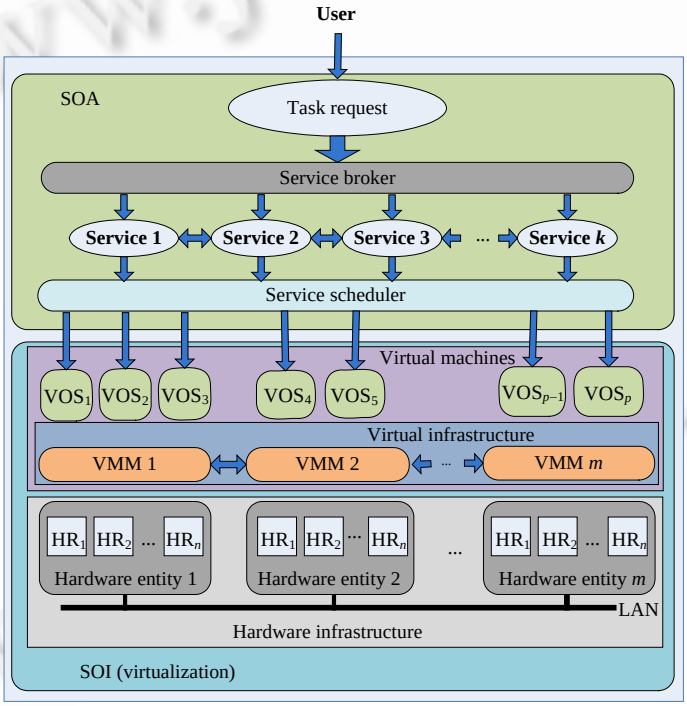


Fig.12 Detail description of SOV task process plane

图 12 SOV 业务处理平面的进一步展开描述

Table 2 Detail description of SOV framework

表 2 面向服务虚拟化框架详细描述

Notation	Definition	Notation	Definition
HR	Hardware resource: the special hardware which implements some special function, such as monitor. Generally, HR cannot provide service independently.	SOI (virtualization)	Service-Oriented Infrastructure represents the infrastructure that supports the whole system of SOA. SOI is composed with hardware infrastructure layer and the virtual machines layer.
Hard entity	Hardware entity is composed by a group of hardware resource, which can provide service independently, such as a hardware server.	Task request	The task request delivered by user, which has to satisfy some descriptive specification.
LAN	The high-speed local area network which is used to connect multi servers.	VMM	Virtual machine monitor
Hardware infrastructure	Hardware infrastructure is the aggregation of all the hardware entities that support the upper software application.	Service scheduler	Service scheduling module takes the responsibility of allocating the tasks into virtual servers, according to the status of each server such as load status and dependability status.
VOS	Virtual operating system: It is referred particularly to the operating system running in the virtual machine. VOS may be either the general operating system or the customized light-weight operating system specified to some application service.	Service broker	Service broker stores the description and interface definition of each service, and provides these information for user requests. When a request arrives, service broker judges whether the QoS and dependability request can be completed, and forward the user request to service.
Virtual infrastructure	Virtual infrastructure is composed by many VMMs which cooperate with each other. It isolates the underlying hardware and is in favor of live migration. It makes the replace and change of underlying hardware not affect the upper application software.	Service	Implementing a certain function by applying some corresponding resources. The services can cooperate with each other, and can composite to form new service. There is loosely coupled relationship between different services, and the service interfaces satisfy normal specification..
Virtual machines	A virtual machine includes the VMM and the upper VOSs here.	User	User of the system services generates and sends service requests according to the service description and interfaces information acquired from broker.

6.2 SOV系统的业务流程和可信赖性保证

在数据业务层中,系统首先依据下层的面向服务的基础设施 SOI 可以提供的功能生成可调用的服务,这些服务有各自的描述和统一的消息协议,并向服务代理(service broker)注册,服务代理储存各服务的描述和接口定义.

管理和监视平面负责实时监控系统的运行情况,并记录系统运行日志,这种监控贯穿整个 SOV 系统,同时将这些数据提交给可信赖性控制平面.可信赖性控制层依据一定的算法,将预先设定的信息和实时采集的运行情况数据加以运算,得到系统各个部分的性能评估.并且对不可以接受的可信赖性降级启动相应的应急响应措施.用户向系统提交业务请求和相应的服务质量 QoS 要求,服务代理根据用户的业务请求和服务接口定义按功能划分成各种具体的服务调用.服务代理根据存储的服务描述和可信赖性控制平面返回的系统可信赖性状况和服务评估情况,判断是否能够满足 QoS 要求,若能,则将请求转发给相应服务,否则,进一步与用户协商.

运行着各种应用程序的 VOS 是服务的最终执行实体.管理和监控平面实时监控各 VOS 的健康状况和功能参数,并汇报给可信赖性控制平面计算可信赖性水平.服务调度器(service scheduler)获取各 VOS 的可信赖性水平和功能参数,并且根据各服务的 QoS 要求将服务分配给各执行实体 VOS,并且管理服务在各 VOS 之间动态的调度和迁移.执行实体 VOS 运行在底层虚拟基础设施之上.虚拟基础设施由运行在各硬件实体之上的 VMM 构成,每个 VMM 之上承载着多个 VOS,各 VOS 之间相互隔离同时相互备份.VMM 之间可以通过动态迁移等技术保证上层执行实体正常工作.虚拟基础设施层实现执行实体和底层硬件基础设施的解耦,保证底层硬件实体的故障、维修等,以尽量少地对上层服务造成影响,从而保证高可用性和可维护性.

下面我们结合在第 1.2 节中提到的系统可信赖性的威胁,考察 SOV 系统如何对几种典型的可信赖性威胁采

取的相应应对措施,来进一步说明 SOV 系统是如何保证系统可信赖性的(如图 13 所示).

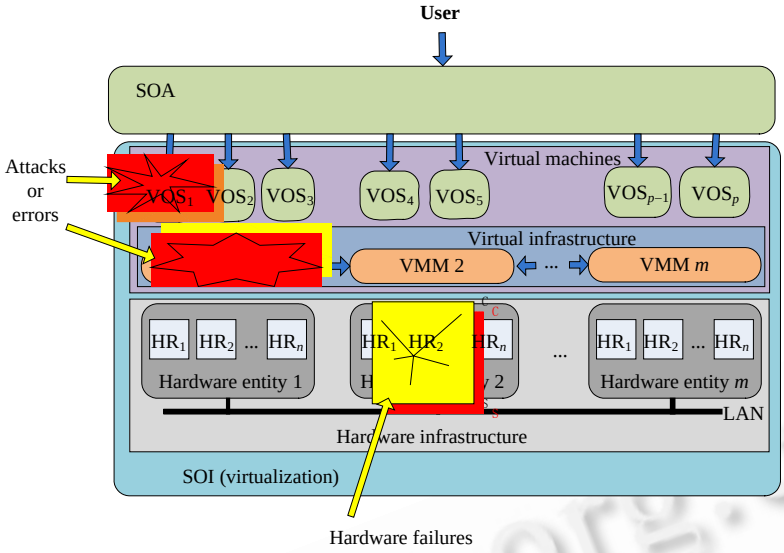


Fig.13 Potential threats to SOV system

图 13 SOV 系统可能面临的主要威胁

由于系统的威胁,如故障、攻击等等,一般都是针对底层基础设施 SOI 进行的,所以下面来分析针对 SOI 的威胁的 3 种典型情况:

(1) 底层硬件故障

由 VMM 构成的虚拟基础设施层实现了上层服务执行实体和底层硬件的解耦.如果一台硬件实体因为故障而宕机,其上承载的应用服务可以和执行实体 VOS 一起,通过动态迁移技术(live migration)迁移到其他硬件实体上运行.动态迁移过程在用户看来几乎感觉不到,从而保证了高可用性.硬件故障修复之后,可以重新接管原来的服务执行实体.整个过程需要各 VMM 在控制与监控平面的管理和控制下进行.随着迁移的进行,上层 VOS 的性能参数也会发生改变,从而触发可信赖性控制平面重新计算.服务调度器根据重新计算的各 VOS 的可信赖性和功能参数以及各服务的 QoS 要求,决策是否重新进行服务的分配和调度.

(2) 虚拟基础设施层运行错误或遭受攻击引起故障

随着虚拟机的日益流行,目前也已经出现了针对虚拟机 VMM 的一些软件缺陷进行的外部攻击.一般的 VMM 中的运行错误可以借助其自身的自检和容错机制来解决,但是,如果严重错误或者遭受攻击而导致 VMM 发生故障时,需要借助于其他机制.在 SOV 系统中,对于 VMM 这样的关键设施需要进行备份.在主 VMM 工作时,其工作状态被实时地记录,一旦 VMM 由于遭受攻击而引起故障时,可以启动它的备份副本,同时将记录的主 VMM 的工作状态导入到备份 VMM 中.另外,当然也可以借助动态迁移技术将发生故障的 VMM 之上承载的 VOS 全部迁移到另外一台负载较轻的 VMM 之上.

(3) 操作系统和应用服务程序运行错误或遭受攻击引起故障

由于软件规模的日益庞大,操作系统和应用服务程序不可避免地会有漏洞或 bug 存在.这些漏洞或 bug 有些会引起系统的运行错误,还有一些被恶意的黑客利用对系统进行攻击.在 SOV 系统中,由于虚拟化技术的隔离机制,使得一个对 VOS 的攻击或者运行错误限制在 VOS 内部,不会对其他 VOS 造成影响,同时也不会影响到底层硬件.对于关键服务,还可以通过在同一 VMM 之上运行一个备份的 VOS,或者不同 VMM 之上的 VOS 通过动态迁移技术保证关键服务的完成,以提高系统的可用性和可靠性.

7 总结和展望

计算系统的硬件种类越来越多,软件规模也越来越庞大,日益增长的复杂性不可避免地带来了各种可信赖性问题.但是,如何定量地、全面地评价一个系统的可信赖性,如何增强系统可信赖性,至今仍是难题.本文首先给出了计算系统的可信赖性的定义,并系统地定义了一整套量化评价指标.同时,对计算系统面临的各种可信赖性威胁进行了详细的归类分析.由于传统方法所存在的不足,人们在解决计算系统可信赖性的研究过程中,不断地去寻求新的技术.虚拟化技术由于其特有的一些特点和机制,受到研究人员的关注.在增强系统可信赖性的应用背景下,虚拟化技术走出了十几年的沉寂而重新成为一大研究热点.本文介绍了已有的虚拟化技术与系统可信赖性相关的研究成果,并且总结了虚拟化技术在增强系统可信赖性方面的各种特性和机制.然而,由于现有的计算系统体系结构的限制,难以将虚拟化技术在增强系统可信赖性方面的优势充分地发挥出来.面向服务的体系结构(SOA)以其松散耦合、平台无关性等特点很好地适应了虚拟化技术的需求.本文对 SOA 的特点和与传统体系结构相比具有的优势作了简要介绍.最后,本文将 SOA 和虚拟化技术相结合,提出了一种增强计算系统可信赖性的系统架构,即面向服务的虚拟化 SOV,并且分析了 SOV 系统如何在遭受各种可信赖性威胁时,运用体系结构优势和虚拟化技术的各种机制保证系统可信赖性.

目前,在虚拟化技术和 SOA 保证系统可信赖性方面的研究尚缺乏数学的理论基础,准确地评价它们在增强计算系统可信赖性的作用需要建立数学模型,这是我们下一步工作的方向.另外,我们要做的工作还有要搭建一个原型系统,对我们提出的这种解决框架应用到实际系统中的一些技术细节进行更深入的研究.

References:

[1] Goldberg RP. Survey of virtual machine research. IEEE Computer Magazine, 1974,7(6):34−45.

[2] Creasy RJ. The origin of the VM/370 time-sharing system. IBM Journal of Research and Development, 1981,25(5):483−490.

[3] Figueiredo R, Dinda PA, Fortes J. Resource virtualization renaissance. IEEE Computer Society, 2005,38(5):28−31.

[4] Gartner Identifies the Top 10 Strategic Technologies for 2008. 2007. http://www.gartner.com/it/page.jsp?id=530109

[5] OGSA glossary of terms. 2005. http://www.ggf.org/documents/GWD-I-E/GFD-I.044.pdf

[6] Algirdas A, Jean-Claude L, Brian R, Carl L. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Dependable and Secure Computing, 2004,1(1):11−33.

[7] Nicol DM, Sanders WH, Trivedi KS. Model-Based evaluation: From dependability to security. IEEE Trans. on Dependable and Secure Computing, 2004,1(1):48−65.

[8] Nahman JM. Dependability of Engineering Systems, Modeling and Evaluation. Berlin, Heidelberg: Springer-Verlag, 2002. 63−74.

[9] Birolini A. Reliability Engineering, Theory and Practice. 5th ed., Berlin, Heidelberg: Springer-Verlag, 2007. 2−24.

[10] Lin C, Wang YZ, Yang Y, Qu Y. Research on network dependability analysis methods based on stochastic Petri net. Chinese Journal of Electronics, 2006,34(2):322−332 (in Chinese with English abstract).

[11] Lin C, Wang Y, Li QL. Stochastic modeling and evaluation for network security. Chinese Journal of Computers, 2005,28(12): 1943−1956 (in Chinese with English abstract).

[12] Cao JH, Cheng K. Introduction to the Reliability of Mathematics. Revised ed., Beijing: Higher Education Press, 2006 (in Chinese).

[13] Madan B, Goševa-Popstojanova K, Vaidyanathan K, Trivedi KS. A method for modeling and quantifying the security attributes of intrusion tolerant systems. Performance Evaluation, 2004,56(1-4):167−186.

[14] Functional safety of electrical/electronic/programmable electronic safety-related systems. IEC Standard 61508, 2000. http://www.iec.ch/zone/fsafety/

[15] Functional safety and IEC 61508. IEC/TR 61508-0, 2005. http://www.iec.ch/zone/fsafety/

[16] Laprie JC. Dependability: Basic concepts and terminology, in English, French, German, Italian and Japanese. In: Proc. of the IFIP WG 10.4 Dependable Computing and Fault Tolerance. Berlin, Heidelberg: Springer-Verlag, 1992. 34−36.

[17] Rose R. Survey of system virtualization techniques. In: Proc. of the 3rd Int'l Conf. on Parallel Processing and Applied Mathematics. 2004. http://www.robertwrose.com/vita/rose-virtualization.pdf

[18] Jones MT. Virtual Linux: An overview of virtualization methods, architectures, and implementations. 2006. http://www-128.ibm.com/developerworks/library/l-linuxvirt/index.html

[19] LeVasseur J, Uhlig V, Chapman M, Chubb P, Leslie B, Heiser G. Pre-Virtualization: Slashing the cost of virtualization. Technical Report, 2005-30, 2005. http://l4ka.org/publications/2005/previrtualization-techreport.pdf

[20] Lawton KP. Bochs: A portable pc emulator for unix/x. Linux Journal, 1996,29(7). http://www.linuxjournal.com/article/1310

[21] Bellard F. QEMU, a fast and portable dynamic translator. In: Proc. of the USENIX Annual Technical Conf. 2000. http://www.usenix.org/event/usenix05/tech/freenix/full_papers/bellard/bellard_html/

[22] Venkitachalam G, Lim BH. Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor. In: Proc. of the General Track: 2002 USENIX Annual Technical Conf. Berkeley: USENIX Association, 2001. 1−14.

[23] IBM System z: z/VM. http://www-900.ibm.com/cn/products/servers/zseries/virtualization/zvm.shtml

[24] Whitaker A, Shaw M, Gribble SD. Denali: Lightweight virtual machines for distributed and networked applications. In: Proc. of the Operating Systems Design and Implementation, 2002. http://www.cs.ucla.edu/~miodrag/cs259-security/whitaker02denali.pdf

[25] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Warfield A. Xen and the art of virtualization. In: Proc. of the ACM Symp. on Operating Systems. Bolton Landing, 2003. 164−177. http://www.cse.unsw.edu.au/~cs9242/current/papers/Barham_DFHHHNPW_03.pdf

[26] Ligneris BD. Virtualization of Linux-based computers: The Linux-vserver project. In: Proc. of the 19th Int'l Symp. on High Performance Computing Systems and Applications (HPCS 2005). 2005. 340−346. http://downloads.revolutionlinux.com/Articles/2005-vserver-theory.pdf

[27] Solaris Internals: Zones. 2007. http://www.solarisinternals.com/wiki/index.php/Zones

[28] Uhlig R, Weiger G, Rodgers D, Santoni AL, Martins FCM, Anderson AV, Bennett SM, Kägi A, Leung FH, Smitb L. Intel virtualization technology. IEEE Internet Computing, 2005,38(5):48−56.

[29] VMware VMotion. Live migration of virtual machines without service interruption. VMware Inc. White Paper. http://www.vmware.com/products/vi/vc/vmotion.html

[30] Clark C, Fraser K, Hand S, Hansen JG, Jul E, Limpach C, Pratt I, Warfield A. Live migration of virtual machines. In: Proc. of the 2nd ACM/USENIX Symp. on Networked Systems Design and Implementation (NSDI). Boston, 2005. 273−286. https://www.usenix.org/publications/library/proceedings/nsdi05/tech/full_papers/clark/clark.pdf

[31] Garfinkel T, Warfield A. What virtualization can do for security. The USENIX Magazine, 2007,32(6):28−34.

[32] VMware Consolidated Backup. Best practices and deployment considerations. VMware Inc. White Paper. http://www.vmware.com/pdf/vi3_consolidated_backup.pdf

[33] Vallée G, Naughton T, Ong H, Scott SL. Checkpoint/Restart of virtual machines based on Xen. In: Proc. of the High Availability and Performance Computing Workshop (HAPCW 2006). 2006. 30. http://xcr.cenit.latech.edu/hapcw2006/program/papers/cr-xen-hapcw06-final.pdf

[34] Badrinath R, Krishnakumar R, *et al*. Virtualization aware job schedulers for checkpoint-restart. In: Proc. of the 13th Int'l Conf. on Parallel and Distributed Systems (ICPADS 2007). 2007. 1−7. http://portal.acm.org/citation.cfm?id=1396951

[35] Ta-Shma P, Laden G, Ben-Yehuda M, Factor M. Virtual machine time travel using continuous data protection and checkpointing. ACM SIGOPS Operating Systems Review, 2008,42(1):127−134.

[36] Liu PC, Chen HB, Zang BY. Enhance the reliability of computer systems in virtual environment. Communications of CCF, 2008, 4(4):24−32 (in Chinese with English abstract).

[37] Bressoud TC, Schneider FB. Hypervisor-Based fault tolerance. ACM Trans. on Computer Systems (TOCS), 1996,14(1):80−107.

[38] VMware double-take. VMware Inc. and Double-Take Software, Inc., 2006. http://www.vmware.com/pdf/vmware_doubletake.pdf

[39] Joshi A, King ST, Dunlap GW, Chen PM. Detecting past and present intrusions through vulnerability-specific predicates. In: Proc. of the 20th ACM Symp. on Operating Systems Principles (SOSP 2005). 2005. 91−104. http://comguywoo.googlepages.com/p91-joshi.pdf

[40] King ST, Chen PM. Backtracking intrusions. In: Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP 2003). 2003. 223−236. http://reference.kfupm.edu.sa/content/b/a/backtracking_intrusions__63470.pdf

[41] King ST, Mao ZM, Lucchetti DG, Chen PM. Enriching intrusion alerts through multi-host causality. In: Proc. of the Network and Distributed System Security Symp. (NDSS 2005). 2005. http://www.cs.umich.edu/~zmao/Papers/ndss05.pdf

[42] Garfinkel T, Rosenblum M. A virtual machine introspection-based architecture for intrusion detection. In: Proc. of the Network and Distributed Systems Security Symp. The Internet Society, 2003. 191−206. https://eprints.kfupm.edu.sa/22103/1/22103.pdf

[43] Dunlap GW, King ST, Cinar S, Basrai MA, Chen DM. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In: Proc. of the 5th Symp. Operating Systems Design and Implementation. Usnix, 2002. 211−224. http://www.cs.uiuc.edu/homes/kingst/Research_files/dunlap02.pdf

[44] Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D. Terra: A virtual machine-based platform for trusted computing. In: Proc. of the 19th ACM Symp. Operating Systems Principles. ACM Press, 2003. 193−206. http://www.cs.columbia.edu/~nieh/teaching/e6118_s04/papers/2_17_garfinkel_terra.pdf

[45] Paul R. DoD towards software services. In: Proc. of the 10th IEEE Int'l Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2005). 2005. 3−6. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1544771

[46] Paul R, Tsai WT, Bayne J. The impact of SOA policy based computing on C2 interoperation and computing. In: Proc. of the 10th Int'l Command and Control Research and Technology Symp. (ICCRTS). 2005. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA464259

[47] IBM alpha works: SOA and web services: New to SOA and Web services. http://www-128.ibm.com/developerworks/soa

[48] Web Services Glossary. W3C Working Group Note. 2004. http://www.w3.org/TR/ws-gloss/

[49] OASIS Reference model for service oriented architecture. Committee Draft 1.0, 2006. http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf

[50] Cisco Systems, Inc. Cisco application oriented networking. http://www.cisco.com/en/US/products/ps6692/Products_Sub_Category_Home.html

[51] Waheed A, Ding JJ. Benchmarking XML based application oriented network infrastructure and services. In: Proc. of the 2007 Int'l Symp. on Applications and the Internet (SAINT 2007). 2007. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4090051

[52] Arsanjani A, Zhang LJ, Ellis M, Allam A, Channabasavaiah K. S3: A service-oriented reference architecture. IT Professional, 2007, 9(3):10−17.

[53] Revisiting the definitive SOA definition. 2005. http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1044083,00.htm

[54] Hao H. What is service-oriented architecture? 2003. http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html

[55] Erl T. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR Pub., 2005.

[56] Mao XS, *et al*. SOA Principles·Methods·Practics. Beijing: Publish House of Electronics Industry, 2007 (in Chinese).

[57] Kanneganti R, Chodavarapu P. SOA Security. Manning Publications Co., 2008.

[58] Lublinsky B. Achieving the ultimate EAI implementation: An overview of EAI implementation approaches with a focus on a process-driven EAI. 2001.

[59] Defining SOA as an architectural style. 2007. http://www.ibm.com/developerworks/architecture/library/ar-soastyle/

[60] Web services architecture. W3C Working Group Note. 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[61] Using infrastructure service orchestration to enable a service-oriented architecture. Cisco Inc. White Paper. http://www.cisco.com/en/US/prod/collateral/netmgtsw/ps6505/ps8463/prod_white_paper0900aecd8068ee04.html

附中文参考文献:

[10] 林闯,王元卓,杨扬,曲扬.基于随机 Petri 网的网络可信赖性分析方法研究.电子学报,2006,34(2):322−332.

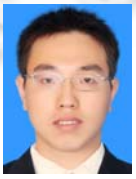
[11] 林闯,汪洋,李泉林.网络安全的随机模型方法与评价技术.计算机学报,2005,28(12):1943−1956.

[12] 曹晋华,程侃.可靠性数学引论.修订版.北京:高等教育出版社,2006.

[36] 刘鹏程,陈海波,臧斌宇.虚拟环境下计算机系统的可靠性增强技术.中国计算机学会通讯,2008,4(4):24−32.

[56] 毛新生,等,编.SOA 原理·方法·实践.北京:电子工业出版社,2007.

林闯(1948－),男,辽宁沈阳人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机网络,系统性能模型及评价.



周寰(1984－),男,博士生,主要研究领域为性能评价.

孔祥震(1985－),男,博士生,主要研究领域为虚拟化技术,系统可信赖性和安全.