一种基于多核的完整性度量实施方法*

石文昌**,宋　元,周春喜
(中国人民大学信息学院,北京　100872)

摘要:为了解决在使用内核完整性度量机制对操作系统进行实时度量时,由于操作系统内核在设计之初没有考虑内核完整性度量机制的出现,导致不能对系统进行及时度量的问题,本研究提出一种基于多核的完整性度量实施方法,该方法通过在操作系统内核相关部分加入对内核完整性度量机制的支持来解决这一问题。本文首先介绍该方法的设计和实现,然后通过实验对该方法的有效性及性能开销进行分析。该方法可以为内核完整性度量机制分配核运行,实现对操作系统内核的及时度量,且性能开销极小。

关键词:操作系统　内核　完整性度量　度量实施方法　多核处理器　调度

中图分类号:TP309　　**文献标识码**:A　　**文章编号**:1002-7378(2020)03-0317-06
DOI:10.13657/j.cnki.gxkxyxb.20201027.008

0　引言

随着处理器制造工艺的不断进步以及多核架构的成功,多核处理器成为现代处理器的主流选择,并且处理器中核的数目越来越多,如Intel推出了80核的处理器原型Teraflop[1],Tilera公司推出全球首款100核处理器TILE-Gx100[2]。

一直以来,计算机软件的发展远远落后于硬件的发展。如何尽可能发挥硬件的性能,一直是研究人员必须面对的问题。多核处理器的出现,对如何发挥硬件的性能提出新的挑战,但同时也为优化软件性能带来新的机遇。国际上有不少工作致力于新型多核操作系统的研究,如Corey[3]、Multikernel[4]、fos[5]、ROS[6]、OpenPiton[7]、Arrakis[8]、LegoOS[9]等。此外,Boydwickizer等[10]、Siddha等[11]以及许多工作团队[12-14]围绕多核处理器对现代操作系统造成的影响展开讨论。但多核的潜力并没有得到充分挖掘,许多问题还有待研究。

如何应对多核处理器带来的挑战,国际上有很多研究。Corey[3]、Multikernel[4]、fos[5]、ROS[6]等都是不同于传统通用操作系统设计理念的新型操作系统的设计成果。Boydwickizer等[10]认为Linux内核可适用于多核处理器,Siddha等[11]分析了多核调度面临的挑战。

完整性度量方法的研究一直是国际上的热点,研究人员开展了大量的研究工作,我们在这方面也已开展不少工作[15-17]。Upen AEGIS系统[18]等为建立可信的系统运行环境,结合硬件在系统引导的过程中进

*国家自然科学基金项目(61472429,61070192和61170240),北京市自然科学基金项目(4122041)和国家社科重大项目(20ZDA062)资助。
【作者简介】
石文昌(1964—),男,教授,博士生导师,主要从事网络空间系统安全研究,E-mail:wenchang@ruc.edu.cn。
【**通信作者】
【引用本文】
石文昌,宋元,周春喜.一种基于多核的完整性度量实施方法[J].广西科学院学报,2020,36(3):317-322.
SHI W C,SONG Y,ZHOU C X. A Multi-Cores-Based Approach to Integrity Measurement Enforcement [J]. Journal of Guangxi Academy of Sciences,2020,36(3):317-322.

行系统完整性度量。IMA 系统[19]等在可执行内容装载时,对其进行完整性检查。Copilot 系统[20]通过增加一个独立于主机的硬件协处理器,实现对运行中的系统的内存映象进行完整性度量。NFORCE 系统[21]借助 Intel 处理器的 SMM 特性,对 Linux 操作系统的内核映象和执行中的进程进行度量。LKIM 系统[22]借助内核的关键数据结构,基于内核的执行流度量 Linux 内核的完整性。SBCFI[23]等通过控制流完整性的角度来动态监控操作系统内核的完整性。OSck[24]通过扫描内核堆来对内核数据的完整性进行度量。但是现有的这些工作都没有涉及如何利用多核资源来进行完整性度量的实施,而这正是本研究所关注的内容。

提高信息系统的安全可信程度是业界长期追求的目标[25],我国网络安全法明文规定要推广应用安全可信的网络安全产品[26]。系统完整性度量是建立系统可信性的关键技术,国际上完整性度量的研究和系统开发都很活跃,并从不同方面对操作系统的完整性进行度量,如 Upen AEGIS 系统[18]、IMA 系统[19]、Copilot 系统[20]、NFORCE 系统[21]、LKIM 系统[22]、SBCFI[23]、OSck[23]、LBM[27]、BehaviorKI[28]、DRIVE[29]、Container-IMA[30]、OB-IMA[31]、Cloud-Monatt[32]等。但是在这个领域中,针对内核完整性度量机制如何充分利用多核处理器潜能的研究还很少。现有的操作系统内核,如 Linux 内核,在设计时没有将内核完整性度量机制考虑在内,不能很好地对内核完整性度量机制提供及时度量的支持;现有的研究成果鲜有涉及如何利用多核来实施内核完整性度量机制;一些简单的方法可以实现及时度量的目的,但却会造成较大的性能开销。针对这些问题,本研究提出一种基于多核的完整性度量实施方法,该方法通过在操作系统内核中加入对内核完整度量机制进行及时度量的支持机制,来实现及时度量并尽量减小性能开销的目标。

1 问题的提出

为了能够真实地反映操作系统的完整性状态,内核完整性度量机制需要对正在运行中的操作系统进行完整性度量,即对操作系统进行实时度量。

多核处理器出现之前,在单核系统上进行实时度量时,由于内核完整性度量机制的引入,加重了原有系统的负载,并且内核完整性度量机制和系统中其他任务竞争核资源运行,经常不能及时运行。多核处理器出现之后,虽然有多个核可以同时用于运行任务,但是在对操作系统内核进行实时度量时,这种情况依然存在。归根结底是由于操作系统内核在进行设计时没有考虑内核完整性度量机制,更没有使其实现及时度量的机制存在。即使内核完整性度量机制能够准确地度量并反映内核的完整性状态,但是由于内核完整性度量机制不能及时运行,度量结果也将不能及时反映操作系统内核的真实状态。

针对现有操作系统中,完整性度量机制实施时存在的这些问题,亟须一种适合多核计算机系统的完整性度量实施方法。这种方法应该可以让内核完整性度量机制对操作系统内核进行及时度量,且其度量结果可以及时反映内核状态。及时度量是指完整性度量机制可以在需要运行时,优先获取核资源并得到及时运行,对系统进行实时度量。

2 完整性度量实施方法的分析与设计

内核完整性度量机制是引入到操作系统内核中的一种新机制。它作为操作系统内核的一部分,对操作系统内核其他部分的完整性进行度量。操作系统内核的很多工作都以内核线程的方式进行处理,这里研究的内核完整性度量机制也将以内核线程的形式出现,我们称其为内核完整性度量任务。

在多核计算机系统上,如何使得内核完整性度量机制对操作系统内核进行及时度量,从本质上来说是为内核完整性度量任务分配核(CPU)资源,并优先调度其多核调度的问题。

2.1 现有调度算法及核资源的分配

操作系统通过调度算法这一内核的关键部分控制 CPU 资源在任务之间的分配。一个好的调度算法是增加系统吞吐量、缩短系统响应时间、维护系统各个任务间平等关系的关键。

现代操作系统内核根据系统中任务的不同特性进行分类,并对不同类型的任务执行不同的调度策略。本文针对国内外应用广泛的 Linux 操作系统进行研究。Linux 内核将任务分为 4 类,并为实时类的任务提供 First in First Out(FIFO)调度策略和 Round Robin (RR)调度策略(图 1)。前者实现了简单的、先入先出的调度算法;后者与前者大致相同,只是任务在耗尽事先分配给它的时间片后才进行调度。此外,Linux 内核还引入了调度类(Scheduling Class)[33]——一种结构化可扩展的调度算法模块。调度类封装了调度策略的细节,使得调度算法的核心

可以不必关心各调度策略的具体实现。图 1 对 Linux 内核(2.6.39)调度类及其实现的调度策略进行了说明。

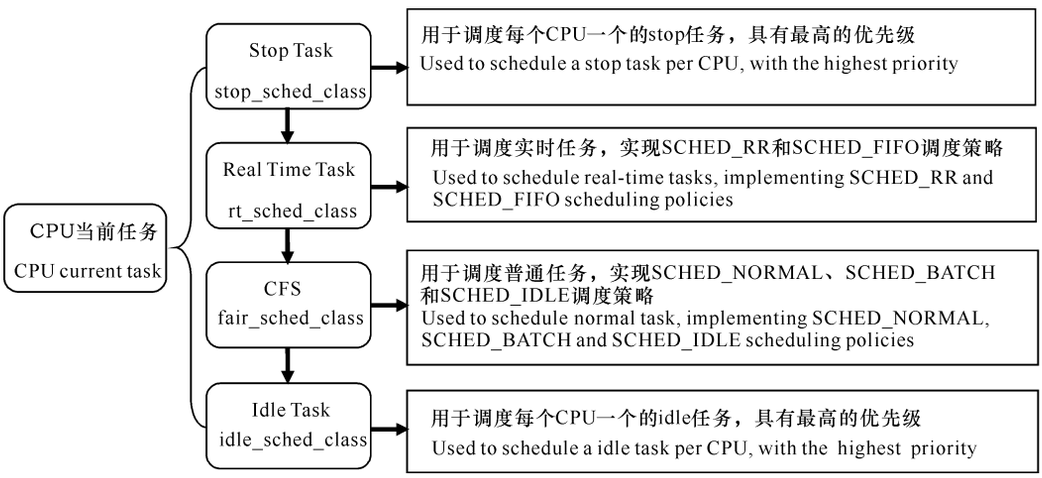

图 1 Linux 中的调度类及其调度策略

Fig. 1 Scheduling classes and scheduling policies in Linux

核完整性度量机制对操作系统内核进行及时度量,才能及时反映系统的完整性状态,因此内核完整性度量任务的特点有两个:(1)运行时,需要及时得到核资源调度运行;(2)运行时,可以一直运行到度量工作完成而尽量不被打断。它的这些特点与操作系统中采用 FIFO 调度策略的实时任务特点最相似,一旦开始就会运行到工作结束,但是这些任务需要等到比它先开始运行的同类任务运行结束后才可以运行,并且有可能被高优先级的任务抢占。

现有的调度算法,如 Linux 内核的调度算法无法为完整性度量机制提供充分的支持,这是因为他们作为通用操作系统调度算法,在设计之初就没有考虑到内核完整性度量实施任务的存在。所以,完整性度量实施方法中须加入对内核完整性度量任务进行调度的机制,以达到及时度量的目的。

2.2 度量实施方案

本方案在现有的调度算法中加入对内核完整性度量机制的支持,使得内核完整性度量任务可以得到核资源并调度。修改后的调度算法对内核完整性度量任务和系统中的其他任务执行不同的调度过程。方案如图 2 所示,在调度时机到来时,调度程序检查当前是否有内核完整性度量任务需要运行,如果是,将当前运行调度程序的核分配给内完整性度量任务,并调度它运行;如果否,按原调度算法调度系统中的其他任务运行。

总体来说,本方案的关键是对完整性度量任务和系统中的其他任务分别进行管理,执行不同的调度过程。目标是及时为内核完整性度量任务分配核资源,进行及时调度,进而达到及时度量的目的。

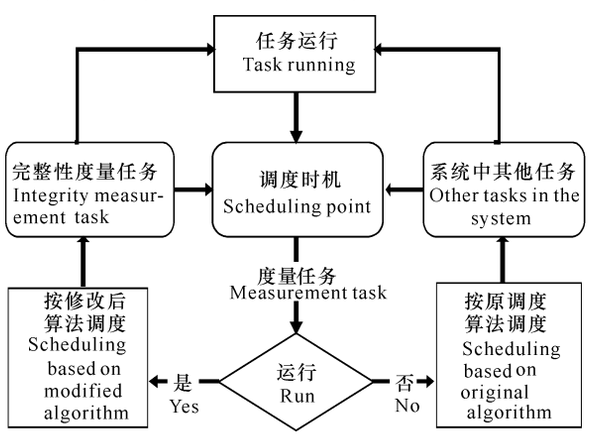

图 2 方法整体流程

Fig. 2 Flow chart of our scheme

3 完整性度量实施方法的实现

3.1 调度算法的选择

Linux 实现了多种调度策略,而这些调度策略的具体实现都封装在调度类中,本研究需要选择最适合的调度算法(调度类),在其中加入对内核完整性度量任务的支持。

如图 1 所示,Linux 内核实现了 4 个调度类。内核完整性度量任务的特点与实时任务的特征最相似,且 Linux 的 4 个调度类中 fair_sched_class、rt_sched_class 可用于对内核完整性度量任务进行调度,其他两个调度类只适用于特定类型任务的调度。两个可取的调度类中,前者的优先级低于后者,且对前者的修改会影响系统中大多数任务的运行调度,因

此本研究选择在实时调度类中加入对内核完整性度量任务的支持。

3.2 调度支持的实现

在实时调度类 rt_sched_class 中加入对内核完整性度量任务的支持,使得内核完整性度量任务可以优先于系统中其他任务得到核资源并调度运行。

如表 1 所示,调度类提供了封装后调度算法的接口,供主调度函数 schedule()调用。在每次调度时机到来时,schedule()会调用调度类的 pick_next_task 函数选取下一个运行的任务,pick_next_task_rt 是函数 schedule()在实时调度类中的实现。本研究在函数 schedule()中加入内核完整性度量任务是否需要运行的判断条件,优先为其分配核资源调度运行。

表 1 调度类结构的主要入口函数

Table 1 Main entry functions of scheduling class structure

sched_class	rt_sched_class	备注 Notes
enqueue_task	enqueue_task_rt	入队列 Enqueue
dequeue_task	dequeue_task_rt	出队列 Dequeue
check_preempt_curr	check_preempt_curr_rt	是否抢占 Preempt check
pick_next_task	pick_next_task_rt	选取新任务 Pick next task
set_curr_task	set_curr_task_rt	改任务策略 Change task policy

当调度时机没有到来,内核完整性度量任务便被唤醒时,我们希望它可以抢占当前正在运行的任务。通过 check_preempt_curr_rt 触发一次新的调度时机可以实现这一目标。需要注意的是,内核完整性度量任务不能抢占如负载均衡内核线程 migration 等任务。

3.3 内核完整性度量任务的管理

通过对内核完整性度量任务进行设置,使其可以被调度算法中的内核完整性度量支持机制辨识,得到优先调度。

将内核完整性度量任务设置为实时任务,使其按照修改后的实时调度算法调度运行;将其调度策略设置为 FIFO,并设置较高的优先级,使其在处于运行状态时,不会被其他实时任务抢占。另外,需要初始化内核完整性度量任务的标识变量,作为调度算法辨识内核完整性度量任务的标识。

3.4 实现效果测试

以 Linux 为基础,直接修改 2.6.39 版本内核上实现了该方法的原型系统。整个实验是在 Dell Power Edge R510 服务器上进行的,处理器为 Intel Xeon E5620 CPU,操作系统是 Ubuntu 11.04 Desktop。以一个周期性运行的内核完整性度量方法对该度量实施方法的有效性进行验证。

实验结果表明,在未采用本完整性度量实施方法的原始系统中,内核完整性度量任务会因为系统中的其他任务而延迟执行,尽管当时系统的负载不重;在采用本完整性度量实施方法的系统中,即使在有大量普通任务和实时任务运行的情况下,当完整性度量任务需要运行时,完整性度量实施方法仍能为其分配核资源,使其及时得到调度运行。此外,根据 unixbench 性能测试结果(表 2),采用本方法的新内核运行内核完整性度量机制,相较于原始系统运行内核完整性度量机制,性能消耗仅为 0.32%。

表 2 性能测试结果

Table 2 Results of performance test

测试项目 Test items	性能评分 Performance score
原始系统 Original system	3 793.56
原型系统 Prototype system	3 781.567
性能开销 Performance overhead	0.32%

4 讨论

本研究设计并实现一种基于多核处理器的完整性度量实施方法,提供对内核完整性度量的实施支持,使其可以对内核进行及时度量。新加入操作系统内核的内核完整性度量机制作为操作系统内核的一部分,无法保证自身的完整性,它的完整性以及度量实施方法的完整性要采用其他方法来保证,这部分工作有待进一步研究。

本方法对以一个内核线程存在的内核完整性度量机制的实施进行探索,当内核完整性度量任务较多时,动态地分配多个核进行度量是下一步的工作方向。

参考文献

[1] VANGAL S R, HOWARD J, RUHL G, et al. An 80-tile sub-100-w teraFLOPS processor in 65-nm CMOS [J]. IEEE Journal of Solid-state Circuits, 2008, 43(1): 29-41.

[2] BORGHINO D. Tilera unveils TILE GX100, the 100-

core general purpose processor [EB/OL]. [2020-06-05]. https://newatlas.com/tilera-tile-gx100-100-core-general-purpose-processor/13236/. Nov 01,2009.

[3] BOYDWICKIZER S,CHEN H,CHEN R,et al. Corey: An operating system for many cores [C]. 8th USENIX Symposium on Operating Systems Design and Implementation,2008:43-57.

[4] BAUMANN A,BARHAM P,DAGAND P,et al. The multikernel:A new OS architecture for scalable multicore systems [C]. Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles,2009:29-44.

[5] WENTZLAFF D,GRUENWALD C,BECKMANN N,et al. An operating system for multicore and clouds: Mechanisms and implementation [C]. Proceedings of the 1st ACM Symposium on Cloud Computing,2010:3-14.

[6] KLUES K,RHODEN B,ZHU Y,et al. Processes and resource management in a scalable many-core OS [C]. 2nd USENIX Workshop on Hot Topics in Parallelism (HotPar'10),2010.

[7] BALKIND J,MCKEOWN M,FU Y S,et al. OpenPiton: an open source manycore research framework [C]. Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2016),2016:217-232. DOI:10.1145/2872362.2872414.

[8] PETER S,LI J L,ZHANG I,et al. Arrakis:The operating system is the control plane [J]. ACM Transactions on Computer Systems,2016,33(4):11:1-11:30.

[9] SHAN Y Z,HUANG Y T,CHEN Y L,et al. LegoOS: A disseminated,distributed OS for hardware resource disaggregation [C]. Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2018),2018:69-87.

[10] BOYDWICKIZER S,CLEMENTS A T,MAO Y,et al. An analysis of Linux scalability to many cores [C]. Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation,2010:1-16.

[11] SIDDHA S,PALLIPADI V,MALLICK A. Process scheduling challenges in the era of multi-core processors [J]. Intel Technology Journal,2007,11(4):361-369.

[12] DEVADAS V,CURTIS-MAURY M. Scalable coordination of hierarchical parallelism [C]. Proceedings of the 49th International Conference on Parallel Processing,2020,77:1-11.

[13] CURTIS-MAURY M,DEVADAS V,FANG V,et al. To Waffinity and beyond: A scalable architecture for incremental parallelization of file system code [C]. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016),2016:419-434.

[14] MIN C,KASHYAP S,MAASS S,et al. Understanding manycore scalability of file systems [C]. Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference,2016:71-85.

[15] LI X,SHI W C,LIANG Z H,et al. Operating system mechanisms for TPM-based lifetime measurement of process integrity [C]. IEEE 6th International Conference Mobile Adhoc and Sensor Systems,2009:783-789.

[16] SHI W C,ZHOU H W,YUAN J H,et al. Detecting compromised kernel hooks with support of hardware debugging features [J]. China Communications,2012,9(10):78-90.

[17] WEI C L,SHI W C,QIN B,et al. Expanding an operating system's working space with a new mode to support trust measurement [C]. International Conference on Information Security Practice and Experience,2015:18-32.

[18] ARBAUGH W A,FARBER D J,SMITH J M,et al. A secure and reliable bootstrap architecture [C]. IEEE Symposium on Security and Privacy,1997:65-71.

[19] SAILER R,ZHANG X,JAEGER T,et al. Design and implementation of a TCG-based integrity measurement architecture [C]. Proceedings of the 13th Conference on USENIX Security Symposium,2004,13:16.

[20] PETRONI N L,FRASER T,MOLINA J,et al. Copilot-a coprocessor-based kernel runtime integrity monitor [C]. Proceedings of the 13th Conference on USENIX Security Symposium,2004,13:13.

[21] HEINE D,KOUSKOULAS Y. N-force daemon prototype technical description [R]. Technical Report VS-03-021,The Johns Hopkins University Applied Physics Laboratory,2003.

[22] LOSCOCCO P,WILSON P W,PENDERGRASS J A,et al. Linux kernel integrity measurement using contextual inspection [C]. Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing,2007:21-29.

[23] PETRONI N L,HICKS M. Automated detection of persistent kernel control-flow attacks [C]. Proceedings of the 14th ACM Conference on Computer and Com-

munications Security,2007:103-115.
[24] HOFMANN O S,DUNN A M,KIM S,et al. Ensuring operating system kernel integrity with Osck [C]. Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems,2011:279-290.
[25] 石文昌.网络空间系统安全概论[M].3版.北京:电子工业出版社,2020.
[26] 中华人民共和国网络安全法(含草案说明)[M].北京:中国法制出版社,2016.
[27] TIAN D J,HERNANDEZ G,CHOI J I,et al. LBM:A security framework for peripherals within the Linux kernel [C]. 2019 IEEE Symposium on Security and Privacy,2019:967-984.
[28] FENG X Y,YANG Q S,SHI L,et al. BehaviorKI:Behavior pattern based runtime integrity checking for operating system kernel [C]. 2018 IEEE International Conference on Software Quality,Reliability and Security,2018:13-24.
[29] REIN A. DRIVE:Dynamic runtime integrity verification and evaluation [C]. Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security,2017:728-742.
[30] LUO W,SHEN Q N,XIA Y T,et al. Container-IMA:A privacy-preserving integrity measurement architecture for containers [C]. 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019),2019:487-500.
[31] XING B,HAN Z,CHANG X L,et al. OB-IMA:Out-of-the-box integrity measurement approach for guest virtual machines [J]. Concurrency and Computation:Practice and Experience,2015,27(5):1092-1109.
[32] ZHANG T W,LEE R B. CloudMonatt:An architecture for security health monitoring and attestation of virtual machines in cloud computing [C]. 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA),2015:362-374.
[33] Linux Kernel Organization. CFS scheduler [EB/OL]. [2020-06-02]. https://www.kernel.org/doc/html/latest/scheduler/sched-design-CFS.html.

A Multi-Cores-Based Approach to Integrity Measurement Enforcement

SHI Wenchang,SONG Yuan,ZHOU Chunxi
(School of Information,Renmin University of China,Beijing,100872,China)

Abstract:In order to solve the problem that when using the kernel integrity measurement mechanisms to measure the integrity of the operating system in real-time,because the kernel integrity measurement mechanisms had not been taken into consideration when designing the kernels,it usually causes the issue that integrity measurement mechanisms are not able to run on time. This paper proposes a multi-cores-based integrity measurement enforcement to solve this problem by adding mechanism supporting the measurement mechanisms to the kernel. This paper explains the design and implements of this method,and then analysis the effectiveness and performance of this method. By locating cores for the measurement mechanisms to run,the enforcement can let measurement mechanisms measure the kernel on time with little performance cost.
Key words:operating system,kernel,integrity measurement,measurement enforcement method,multi-cores processor,scheduling

责任编辑:陆　雁