DOI: 10.3969/j.issn.1671-0673.2017.03.021

一种基于本地验证的可信度量方法

周少皇,郭玉东,王 炜,林 键

(数学工程与先进计算国家重点实验室,河南 郑州 450001)

摘要: 针对 TPM-IMA 远程验证中隐私泄露等安全问题,提出一种基于本地验证的可信度量方法,在计算平台的可信模块内建立验证主体,在模块内部一次性完成度量计算和完整性验证。基于该度量方法,采用某国产安全密码芯片实现完整性度量架构 SCC-IMA 原型系统。实验结果表明,该方法能有效阻止未知程序的运行,避免远程验证的安全问题,对系统启动时间的影响仅为 8.2%,系统性能损失不超过 9%。

关键词: 本地验证; 可信度量; 安全密码芯片; 完整性度量架构

中图分类号: TP309 **文献标识码:** A **文章编号:** 1671-0673(2017)03-0364-06

Trusted Measurement Method Based on Local Verification

ZHOU Shaohuang, GUO Yudong, WANG Wei, LIN Jian

(State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China)

Abstract: In order to solve the security problems in the remote verification of TPM-IMA, a trusted measurement method based on local verification is proposed in this paper, which sets up the verification subject in the trusted module of computing platform, and accomplishes metric calculation and integrity verification inside the module by one time. Based on this measurement method, we implement an integrity measurement architecture SCC-IMA prototype system on a certain domestic security cipher chip. The experimental result shows that the method is able to prevent the unknown program from running effectively with no security problems of remote verification, while its effect on system boot time is just 8.2% with the system performance loss no greater than 9%.

Key words: local verification; trusted measurement; security cipher chip; integrity measurement architecture

可信计算的核心思想是“逐层度量,传递信任”。一个可信计算平台从可信根开始进行可信引导,如 CRTM 度量 BIOS、BIOS 度量引导程序、引导程序度量操作系统内核、再通过完整性度量架构 IMA(integrity measurement architecture)[1] 度量应用层程序,将信任关系从可信根逐步扩大到整个计算平台。完整性度量是可信计算平台确定信任关系的主要度量技术[2],而度量方法则直接影响到可信计算平台确定信任关系的安全和性能,度量方法成为可信计算研究中的关键问题。

国际可信计算组织 TCG 以 TPM(trusted platform module)为可信度量的基础,制定了相应的 TPM 技术规范,而中国可信计算工作组 TCMU 则制定了以 TCM(trusted cryptography module)为核心的技术规范。目前,大多数完整性度量方法采用基于 TPM 的度量机制。以 TPM-IMA 的度量流程为例,其平台间的度量机制[3] 虽然有完整性挑战协议的安全保护,但在远程验证过程中仍容易遭受攻

收稿日期: 2015-11-03; 修回日期: 2015-12-06

作者简介: 周少皇(1991-),男,硕士生,主要研究方向为计算机系统结构、信息安全;
郭玉东(1964-),男,教授,硕士,主要研究方向为操作系统、信息安全。

击,比如系统隐私泄露问题,攻击者可窃取用户的系统配置信息[4]。为了解决系统隐私泄露问题,有些研究者提出平台属性的概念[5],并实现了相应的证明系统[6-7],但平台属性依然是系统隐私信息,仅增加了攻击者分析数据的难度,攻击者仍可以获取到。文献[8]针对验证挑战协议进行改进,在度量的过程中增加随机变量,使得每次生成的度量值都不相同,但也为验证过程增加了难度。以上的解决方法均是对基于 TPM 度量机制的改进,增加了度量机制安全的同时却也增加了实现的复杂度。

本文分析了 TPM-IMA 的度量流程,针对其远程验证中存在的安全问题提出了一种基于本地验证的可信度量方法,该方法不依赖可信第三方,在本地集中完成度量计算和验证,避免了 TPM-IMA 远程验证中的安全问题。基于这种度量方法,以某国产安全密码芯片 SCC(security cipher chip) 为计算平台的可信模块,构建出了 SCC-IMA 的度量流程,实现了在 SCC 内部完成本地验证。此外,本文还依据度量流程设计出了 SCC-IMA 原型系统,基于本地验证对应用层程序进行可信度量,进一步证明了该度量方法的有效性。

1　TPM-IMA 的可信度量流程

如图 1,TPM-IMA 由度量系统、证明系统和验证系统组成,前两个系统部署在可信计算平台端,而验证系统则部署在可信验证端。度量系统负责对计算平台上的度量对象进行度量并保存度量值; 证明系统负责将度量值安全传递给验证方; 验证系统负责对被验证平台的度量值进行验证。TPM-IMA 通过在内核中插入度量点,利用 TPM 的硬件密码计算功能完成对度量对象的散列计算,将度量值以度量列表 ML(measurement list) 的形式存储在内核空间,TPM 在每次度量计算的同时都会扩展其内部 PCR(platform configuration register) 寄存器的值,以此保证度量列表的完整性。远程验证过程中,应用层度量代理首先读取内核空间的 ML,并读取 PCR 扩展值,然后通过完整性挑战机制将 ML 和 PCR 扩展值一并发送给验证方。验证方对 ML 中的度量值按顺序重新扩展计算,并和 PCR 扩展值进行对比校验,以此验证了 ML 的完整性,然后再将 ML 中的度量值和自身存储的基准值进行比对验证,并将最终验证结果返回给可信计算平台。

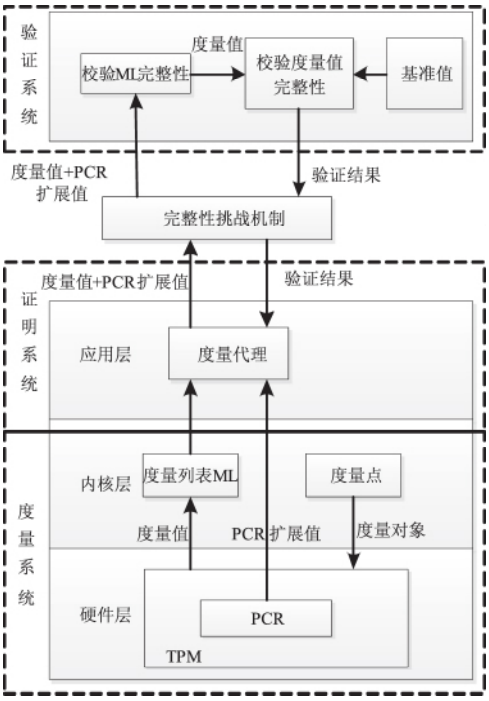

图 1　TPM-IMA 度量流程

由于 TPM 自身不具备完整性验证功能,因此 TPM-IMA 不能使用 TPM 进行完整性验证,需要通过完整性挑战协议依靠第三可信方进行远程验证。远程验证过程中,由于 PCR 寄存器的扩展值保证了被验证平台 ML 的完整性,攻击者无法对度量列表进行篡改,但却容易造成系统隐私泄露,其原理如下: 攻击者虽然不能伪造度量值,但可以在证明系统和验证系统之间截获通信数据,获得 ML。又因为度量值采用的是标准散列值,攻击者可以事先对自身系统的软件进行相应的散列计算,建立基准值库,然后解析截取到的 ML,将其中的度量值和基准值库中的基准值逐个进行匹配,进而获取度量计算平台的系统配置信息。

除了隐私泄露问题,攻击者还可利用 DDoS 等网络攻击阻断度量验证过程的进行。此外,TPM-IMA 的完整性度量均以标准散列值作为度量基准值,虽然散列算法具有单向性,但随着针对于 MD5、SHA-1 等散列算法的碰撞攻击技术[9]的出现,攻击者可以伪造数据并仍能得到相同的散列值,这为恶意程序正常通过完整性验证提供了可能,因此基于散列值的完整性度量已经不可靠,需要更强的安全机制对基准值提供安全保护。

2　基于本地验证的可信度量方法

从 TPM-IMA 度量流程中可以看出,虽然证明系统为度量系统和验证系统之间提供了保护,但依

然存在诸如系统隐私泄露等安全问题。若将其中的验证系统置于计算平台本地进行本地验证，则可避免远程验证带来的安全问题。因此，本文在 TPM-IMA 的度量方法上进行改进，提出一种基于本地验证的可信度量方法，并采用 SCC 实现了 SCC-IMA 的本地验证可信度量。

2.1 本地验证度量过程

完整性度量过程主要包括度量计算和完整性验证两个子过程。度量计算由度量主体、度量客体、度量计算实体组成；完整性验证由验证主体、验证客体、验证标准组成。度量主体即度量的发起者，如插入在内核中的各个度量点的度量代码；度量客体即度量的对象，如应用层程序等；度量计算实体即度量计算的实施者，一般位于计算平台的可信模块中，如 TPM；验证主体即完整性验证的实施者，如第三方验证平台；验证客体即完整性验证的对象，如度量值；验证标准即验证的参考系，如验证平台的基准值。根据以上定义，TPM-IMA 的远程验证度量过程如图 2 所示。

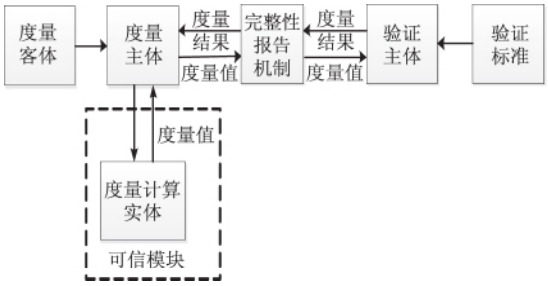

图 2 TPM-IMA 远程验证度量过程

在 TPM-IMA 度量过程的基础上，将验证主体内置于可信模块中。当度量主体度量度量客体的时候，度量主体将度量客体和验证标准一并发送给可信模块。在可信模块中，度量计算实体首先对度量客体进行度量计算，然后将计算出来的度量值直接发送给验证主体，验证主体根据验证标准对度量值进行校验，最终将验证结果返回给度量主体。本地验证度量过程如图 3 所示。

根据可信传递的思想，只有度量过程中所涉及到的所有实体是可信的，其度量出来的结果才能是可信的。在本地验证度量过程中，只有确保度量主体、度量计算实体、验证主体、验证标准的可信，才能保证最终验证结果的可信。在 IMA 中，度量主体位于操作系统内核层，计算平台的可信引导已经确保了内核是可信的。度量计算实体和验证主体在可信模块中，其可信性毋庸置疑。验证标准若由可信方提供，能防止被篡改或者被伪造，则亦是可信的。

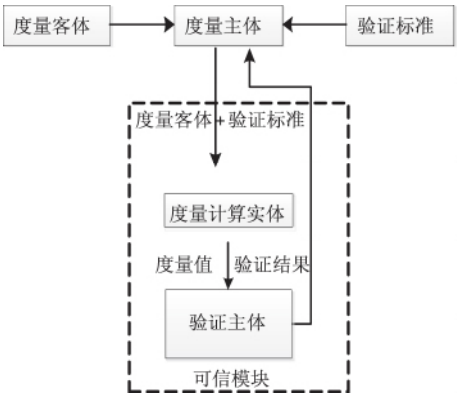

图 3 本地验证度量过程

2.2 SCC-IMA 的度量流程

SCC 是我国自主研发的一种密码设备，具有加解密、签名验签、可信度量及安全存储等功能。SCC 作为外部设备通过 PCI-E 总线连接在计算平台上，并在可信引导过程中作为可信根，因此本文所研究的 SCC-IMA 是在计算平台以 SCC 为可信根进行可信引导的前提下进行的。

SCC-IMA 将 SCC 作为计算平台的可信模块，采用本地验证可信度量方法，在 SCC 内部完成度量计算和完整性验证。同时，验证标准即完整性验证所使用的基准值在原来散列值的基础上进行数字签名，进一步确保基准值只能由特定可信方提供，其他人没有生成基准值的私钥，故其无法伪造合法的基准值。SCC-IMA 度量流程主要由基准值模块、完整性度量模块和 SCC 3 部分进行。基准值模块为完整性度量模块提供可信的基准值；完整性度量模块是 SCC-IMA 中的度量主体，位于可信内核层，其从基准值文件中提取基准值，并将度量对象和基准值传递给 SCC 密码服务接口，调用 SCC 驱动对度量对象进行度量。SCC 完成实际的度量验证操作，并返回度量验证结果。度量流程图如图 4 所示。

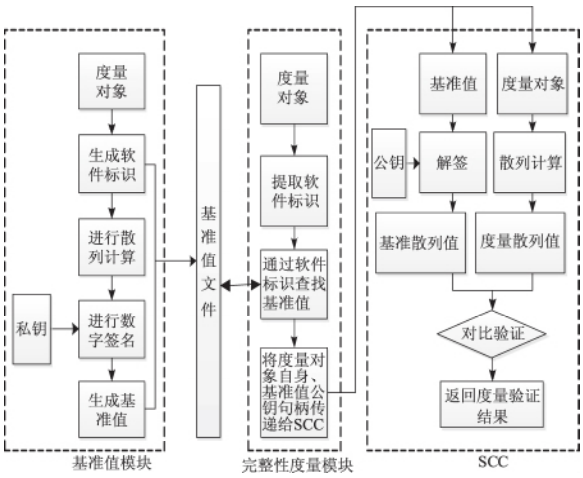

图 4 SCC-IMA 度量流程图

3　SCC-IMA 原型系统的设计与实现

本文根据 SCC-IMA 度量流程,设计和实现了如图 5 所示的原型系统。该原型系统的主要任务是从内核初始化运行第 1 个用户层程序 init 开始,对所有用户层进程的程序文件进行加载时度量,只允许度量通过的程序开启进程。在本系统中,基准值保存在基准值文件中,在度量的过程中映射到内核空间,以供完整性度量模块提取。度量主体位于内核层,由度量点、完整性度量模块及 SCC 度量设备驱动组成,通过在度量对象的加载过程中插入度量点,在度量点调用完整性度量模块的度量接口,进而由 SCC 设备驱动和 SCC 交互完成度量任务。在系统的实现过程中,根据主要功能可将 SCC-IMA 原型系统划分为基准值功能模块和度量功能模块,以下分别对两个模块进行实现。

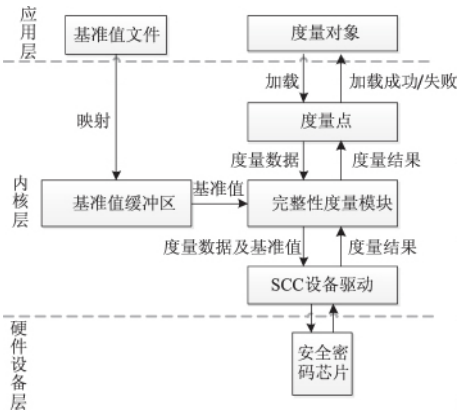

图 5　SCC-IMA 原型系统结构图

3.1　基准值功能模块

基准值的格式、提取是基准值功能模块需要解决的主要问题。本系统使用软件 ID 索引的方式将基准值和 ID 相关联,存储到基准值文件中。由于 Linux 系统上执行的二进制程序主要为可执行程序、动态加载库和内核模块,三者均为 ELF 格式文件,而脚本文件以及一些关键的系统配置文件为非 ELF 格式文件。因此,SCC-IMA 的度量对象可分为 ELF 格式文件和非 ELF 格式文件。ELF 格式文件头部有特定的格式,分析其头部结构体 ELF32_Ehdr,其中,e_ident 字段中的 8 位 ~ 16 位作为保留位没有定义,可在其 10 位 ~ 13 位填充随机数,将软件 ID 和度量对象绑定,在提取基准值时可根据文件头部的 ID 索引查找其对应的基准值。而非 ELF 格式文件,比如脚本文件,没有特定格式,但同一路径下其文件名是唯一的,因此,可使用经典的字符串哈希算法 BKDRHash 对其绝对路径进行哈希,将哈希值作为非 ELF 格式文件的软件 ID,在提取基准值时将绝对路径文件名字符串重新哈希得到软件 ID,然后索引查找其对应的基准值。

为了提高度量时基准值的提取速度,本原型系统在内核空间建立哈希表作为缓存区,其结点数据结构如图 6 所示。结点以软件 ID 为键值,由于 ELF 格式 ID 所生成随机数和 BKDHash 生成的哈希值均为 4 字节,因此,结点中 id 字段可同时兼容 ELF 格式和非 ELF 格式文件的 ID 格式,sign_value 字段存储基准值,其它字段可存储度量次数和度量标志位等度量属性。在内核初始化时建立该哈希表,之后每次度量成功都将所对应的基准值保存为结点加入到哈希表中,当再次提取基准值时便可从哈希表中直接快速查找,提取出对应的基准值,若未查找到则从基准值文件中查找。

```
Struct hash_node
{
        char id[4]
        char sign_value[64];
        int count;
        struct hlist_node list;
        bool flag;
};
```

图 6　基准值哈希表结点数据结构

3.2　度量功能模块

度量功能模块是 SCC-IMA 原型系统中实现度量功能的主要部分,其逻辑结构如图 7 所示,上一层通过调用下一层的接口实现所需功能。

图 7　度量功能模块逻辑结构图

SCC-IMA 采用拦截系统调用的方式完成加载时度量,即在程序文件加载之后、运行之前进行度量,因此,需分析度量对象的加载过程并插入度量点。在度量点调用功能逻辑层的接口函数,功能逻辑层包含与度量相关的各种逻辑功能的接口函数,如 check_file_integrity( ),measure_file( ),get_standardvalue( ) 等; 密码服务层为 SCC 的密码服务组件接口,如 generate_random( ),hash_file( ),import_key( ),get_key( ),trusted_measure( ) 等,为逻辑功能层提供密码服务功能; 设备驱动层为 SCC 的驱动程序,通过 PCI-E 总线负责和 SCC 交互。

ELF 格式的程序文件主要为可执行程序、动态链接库及内核可加载模块，通过分析三者在内核层的加载路径，分别在 execve( )、mmap_pgoff( ) 和 init_module( ) 等 3 个系统调用适当地方插入度量点。度量点的选取直接关系到度量的效率，若插入过早，则容易造成 TOCTOU[10] 攻击，若插入太晚，则容易造成大量的冗余度量。以 ELF 格式可执行程序的加载流程为例，进入 execve( ) 系统调用后，内核首先根据文件名读取文件头部的 128 字节写入 linux_binprm 结构体中，然后调用 search_binary_handler( ) 确定当前可执行程序对应的处理程序，接着处理程序 load_elf_binary( )，根据 ELF 文件类型进一步加载可执行程序文件[11]。因此，可利用内核自身对 ELF 文件类别的识别功能，将度量点设置在 load_elf_binary( ) 中进行，并且还要确保度量点必须在先后调用的 deny_write_access( ) 和 allow_write_access( ) 两个函数之间，保证文件在度量期间不会被修改。

对于非 ELF 格式的文件，比如一些重要的脚本文件和配置文件则需要在系统启动脚本 /etc/rc.d/rc.sysinit 中添加度量点，在系统启动初期尽可能早地调用度量程序对重要的配置文件进行度量。

4 实验与结果

4.1 实验说明

本实验在一台连接有 SCC 设备的可信计算平台上进行，采用中标麒麟国产操作系统，内核版本号 3.7.1。首先在内核中添加 SCC 设备驱动及 IMA 度量文件并成功编译出具备度量功能的内核文件 vmlinuz.ima，然后使用 SCC 生成一对非对称密钥 Prv 和 Pub，并运行基准值生成工具 sv 遍历系统上所有 ELF 文件，在文件头部写入软件标识并使用 Prv 生成基准值，最终在 /boot 目录下生成基准值文件 standardfile.elf，其存储内容如图 8 所示。遍历系统中的脚本文件和配置文件，采集基准值生成基准值文件 standardfle.noelf。在加载启动 vmlinuz.ima 内核之前将 Pub 注入到 SCC 中作为可信度量密钥，在度量过程中所使用的散列算法和签名算法分别为 SM3 和 SM2 国产密码算法。

4.2 实验结果

实验的目的主要是测试 SCC-IMA 原型系统在安全和性能两个方面对计算系统的影响。在可信计算平台，启动内核文件 vmlinuz.ima，通过运行未知应用程序 test 和加载未知可加载内核模块 testmodule.ko，界面显示分别如图 9 和图 10 所示。

图 8 基准值文件存储内容

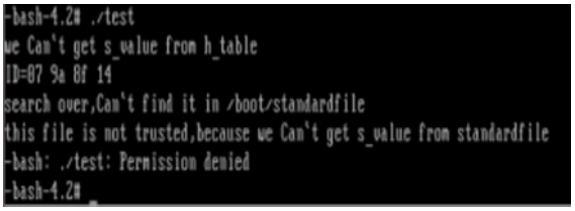

图 9 运行未知可执行程序 test

图 10 加载未知可加载内核模块 testmodule.ko

结果表明，SCC-IMA 既能有效阻止未知可执行程序的运行，也能阻止未知可加载内核模块的加载，在安全上达到预期效果。

为了测试 SCC-IMA 对系统启动时间和系统整体性能的影响，系统使用 lilo 引导程序分别启动加载 vmlinuz.ima 和不带度量功能的内核文件 vmlinuz，以此分别在开启和关闭 SCC-IMA 度量功能的情况下测试系统启动时间，实验结果如表 1 所示。

表 1 系统启动时间对比

启动条件	系统启动时间
启动 SCC-IMA	30.1s
不启动 SCC-IMA	27.8s

通过计算分析，SCC-IMA 关于系统启动时间的延迟影响为 8.2% 左右，通过运行 unixbench-5.1.2测试对系统整体性能的影响，两次得分分别为 1595.9 和 1741.8，整体性能损耗为 9%。可以看出，SCC-IMA 对于系统性能的影响不大，在用户的可接受范围之内。

5 结束语

本文提出了一种基于本地验证的可信度量方法，并采用该度量方法实现了 SCC-IMA 原型系统。基于本地验证的度量方法是对基于 TPM-IMA 度量

方法的一种改进,目的在于解决远程验证过程中的安全问题。该度量方法最大的特点是将完整性验证过程内置于可信模块中进行,利用硬件保护了本地验证的安全可信,此外,使用公钥密码体制进一步增强了基准值的安全可信。实验结果充分证明了本地验证度量方法的有效性。但在原型系统中,基准值采集对象的范围仅是当前系统中的文件,还需要解决基准值随软件变化而进行相应的增删和更新等问题,因此,下一步需建立一个基准值管理系统,进一步完善此原型系统。

参考文献:

[1] 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: 223-238.

[2] 冯登国,秦宇,汪丹,等. 可信计算技术研究 [J]. 计算机研究与发展, 2011, 48(8): 1332-1349.

[3] 张立强,张焕国,张帆. 可信计算中的可信度量机制 [J]. 北京工业大学学报, 2010,36(5): 586-591.

[4] 孙丽娜. 可信计算中的远程证明技术研究 [D]. 沈阳: 东北大学, 2012.

[5] Sadeghi A R, Stüble C. Property-based attestation for computing platforms: caring about properties, not mechanisms [C] //Nspw Proceedings of the Workshop on. 2004: 104-109.

[6] Chen L, Landfermann R, Hr H, et al. A protocol for property-based attestation [C] //Proc. of the First ACM Workshop on Scalable Trusted Computing (STC). 2006: 7-16.

[7] Feng D G, Yu Q. A property-based attestation protocol for TCM [J]. Science China (Information Sciences), 2010, 3(3): 454-464.

[8] Jie L S, Ping H Y. A Privacy-Preserving Integrity Measurement Architecture [C] //Electronic Commerce and Security (ISECS). 2010: 242-246.

[9] 沈昌祥,张焕国,冯登国,等. 信息安全综述 [J]. 中国科学,2007,37(2): 129-150.

[10] Chang X, Xing B, Liu J, et al. LWRM: A lightweight response mechanism for TCG TOCTOU attack [C] //Performance Computing and Communications Conference (IPCCC). 2009: 200-207.

[11] 郭玉东. Linux 操作系统结构分析 [M]. 西安: 西安电子科技大学出版社,2002.

(上接第 363 页)

[2] Fauconnier D, De Langhe C, Dick E. A dynamically optimized finite difference scheme for Large-Eddy Simulation [J]. Journal of Computational and Applied Mathematics, 2010, 234(7): 2080-2088.

[3] Feng Wan, Fernando Porte-Agell, Rob Stoll. Evaluation of dynamic subgrid-scale models in large-eddy simulations of neutral turbulent flow over a two-dimensional sinusoidal hill [J]. Atmospheric Environment, 2007,41(31): 2719-2728.

[4] Bakker A, Oshinowo L M. Modeling of turbulence in stirred vessels using large eddy simulation [J]. Chemical Engineering Research and Design, 2004, 82(9): 1169-1178.

[5] Viré A, Knaepen B. On discretization errors and subgrid scale model implementations in large eddy simulation [J]. Journal of Computational Physics, 2009, 228(22): 8203-8213.

[6] Yang C Q, Qiang W, Cheng C, et al. Accelerating Pqmrcgstab Algorithm on Xeon Phi [C] //Advanced Materials Research 709. 2013: 555-562.

[7] Pennycook S J, Hughes C J. Exploring SIMD for Molecular Dynamics, Using Intel Xeon processors and Intel Xeon Phi Coprocessors [C] //IEEE International Symposium on Parallel & Distributed Processing. 2013: 1085-1097.

[8] Andrey Valdimirov and Cliff Addision. Cluster-level Tuning of a Shallow Water Equation Solver on the INTEL XEON PHI Architecture [J]. Colfax International, 2014, 12(5): 145-150.

[9] Intel Manycore Platform Software Stack. [EB/OL]. [2015-11-15]. http://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss.

[10] Reinders J. An Overview of Programming for Intel ® Xeon ® processors and Intel Xeon Phi coprocessors [J]. Advances in Meteorology, 2012, 35(10): 261-269.

[11] Zhen X, Kuan L, CanQun Y, et al. Performance evaluation of communication and programs in MIC-CPU symmetric mode [J]. High Performance Computing for Weather Water & Chinate, 2013, 29(19): 354-359.

[12] Brandfass B, Alrutz T, Gerhold T. Rank reordering for MPI communication optinization [J]. Computers & Fluids, 2013, 80(7): 372-380.

[13] Andrey Vladimirow and Vadim Karpusenko. Heterogeneous Clustering with Homogeneous Code: Accelerate MPI Applications without Code Surgery Using INTEL XEON PHI Coprocessors [J]. Colfax International, 2013, 69(12): 172-180.