文章编号:1671-4598(2018)06-0180-04 DOI:10.16526/j.cnki.11—4762/tp.2018.06.046 中图分类号:TP302 文献标识码:A

基于多级队列缓存淘汰算法的处理器全数字仿真优化

范延芳，韦涌泉，王向晖
(北京空间飞行器总体设计部，北京 100094)

摘要：通过虚拟目标机实现星载软件的测试是节约卫星开发成本，提高卫星软件测试效率的重要手段；作为星载计算机的核心部件，虚拟处理器的指令集执行速度直接决定虚拟目标机的仿真效率；采用多级队列缓存淘汰策略对 QEMU 原有的动态指令翻译实现进行优化，提高仿真处理器的执行效率，因此若采用仿真处理系统加载星载嵌入式软件进行测试，可以根据测试需要，在仿真处理器可实现范围内加速运行被测软件，从而实现缩短软件测试周期的目的；选取某星载中心计算机嵌入式应用软件为测试对象，实验表明，采用优化算法后的仿真处理器的运行速度可以达到平均 260 MIPS，是 QEMU—2.6.1 版未优化前实现的仿真处理器处理速度的 9.3 倍，即，采用仿真处理器能够使被测软件运行在 9 倍于硬件处理器的运行速度下，大大提升了软件测试效率，缩短了测试周期。

关键词：动态指令翻译；多级队列缓存淘汰算法；处理器仿真

Dynamic Instruction Translation Based Processor Full Digital Simulation optimization

Fan Yanfang, Wei Yongquan, Wang Xianghui
(Beijing Institute of Spacecraft System Engineering, Beijing 100094, China)

Abstract: Flight software testing with virtual target machine may decrease Satellite development costs and increase the product's quality. During the development of flight software, the processor emulator is an essential tool for software development, verification and the core component of the satellite simulator, which can be substituted for the real hardware. A BM3803 processor simulation method based on optimiezed dynamic instruction translation is proposed, in which, multi queue algorithm is applied to optimize cache elimination strategy. Experimental results show that, after the opitimiztion, the processing speed of the optimized process emulator may reach 260MIPS on average, whichi is 9. 6 times of the speed of process emulator provided by QEMU 2. 6. 1 version. Comparing with the testing speed given by the hardware system, the testing system developed by with the emulator may increase the testing speed by 9 times.

Keywords: dynamic instruction translation; multi queue cache elimination algorithm; processor emulator

0 引言

从国外卫星系列的成功发射可以看出，卫星从生产、组装建造到发射的时间已从传统模式的数月缩短至以小时为单位，并且随着空间技术的提高和应用需求的扩展，星载嵌入式应用软件的功能越来越多，逻辑也越来越复杂[1]。这些繁多的、复杂的需求和功能都需要通过完备的软件测试得以验证。嵌入式软件测试的一大特点就是绝大部分测试均需要通过外部激励触发执行。对于智能卫星这样的复杂大系统而言，很多外部激励需要具备一定的等待时长才能产生。因此，若采用硬件设备实施嵌入式软件测试，其测试周期必然是各测试用例等待时间的总和。以卫星数管软件的热控逻辑测试为例，遍历一颗卫星的所有热控回路的测试常常需要 24 小时不间断地持续几天甚至 1 周时间，大大影响了卫星的研制周期。

为了解决此问题，提出采用全数字虚拟机替代硬件目标机实现星载软件逻辑测试。此方法一方面能够为星载软件的前期预研提供调试环境，大大节约开发成本，另一方面，如果能够让虚拟目标机实现超速运行，也能提高软件测试的效率，缩短软件研制周期，从而实现提升软件质量的目的[2]。

QEMU（quick emulator）是一款开源仿真平台，支持多种处理器的功能仿真，包括 x86、PowerPC、ARM 和 Sparc 系列处理器[3]，目前被业内公认为应用最广、效率最高的虚拟机实现软件。其采用动态指令翻译技术，在目标码运行时动态实现将目标机代码转为宿主机代码的过程全过程。但经试验测试，采用 QEMU 开源代码搭建的虚拟处理器其指仿真效率仅能达到 20～25 MIPS[4]，远不能达到超速模拟星载软件运行的效果。

本文提出采用多级队列缓存淘汰策略对 QEMU 的动态指令翻译实现进行改进和优化。该策略通过提高被翻译指令块的重用效率提高仿真处理器的整体处理速度。试验证明，采用此仿真处理器实现的虚拟目标机不但能够在正常运行速度下满足测试需要，还能够成倍加速目标二进制代码的执行速度，从而达到缩短软件测试周期的目的。相对直接采用 QEMU 开源代码实现的仿真处理器，该方法实现的仿真处理器能更高效的完成星载软件的仿真测试任务，为提高星载软件的开发效率，缩短了软件开发周期，提升软件产品质量提供有力保障。

收稿日期:2017-12-25； **修回日期**:2018-01-22。

作者简介:范延芳(1977-),女,江苏丹阳人,硕士研究生,主要从事星载嵌入式软件方向的研究。

1　影响指令集仿真速度因素分析

QEMU 的动态指令编译系统由 4 个模块和一块缓存组成，如图 1 所示。

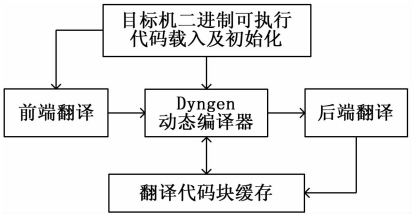

图 1　动态翻译系统模块组成

目标机二进制可执行代码载入和初始化模块是系统启动后首先调用的模块，通过建立客户端栈和客户端数据空间等建立客户端运行环境。前端解码器负责将二进制指令码生成中间代码块；后端编译器负责将中间代码块翻译为客户机可识别的二进制代码。如如建立客户端栈和客户端数据空间等。然后前端翻译模块将目标二进制代码分割为相对短小、简单的指令集，称为“微操作”。每个“微操作”均由一小段 C 函数实现，且每个 C 函数均被 GCC 编译为一个对象文件（基本代码块）。“微操作”由大量实验精选产生，其数量远小于目标 CPU 的指令和操作数的所有组合，且由其组成的集合能够完整表示不同指令集体系下的所有指令。为了增强“微操作”的可读性和紧凑性，从目标机可执行代码到“微操作”的转换完全通过人工编写代码实现。由于此转换过程是在目标机代码执行前完成的，此部分的转换效率并不直接影响仿真处理器的处理速度。

代码运行时，若动态指令翻译模块发现有未被翻译的“基本代码块”，则调用后端翻译模块将其翻译为宿主机能够识别和执行的主机代码，将翻译结果存储在缓存中，并执行，否则直接跳转到翻译代码地址处直接执行。鉴于由于此操作是在目标机代码执行过程中实时进行的，此步骤的执行效率将直接影响和决定仿真处理器的执行速度。

2　QEMU 缓存淘汰策略的问题

如图 1 所示，动态指令编译系统内包含一个地址连续的存储空间。系统用该空间缓存所有被翻译过的指令块 TBs (Translated Blocks)，并采用链表的方式将所有 TB 关联起来，如图 2 所示。链表中每个“TB”都拥两个指针，分别指向跳转到当前“TB”的上一个“TB”的地址和当前“TB”跳转至的下一个“TB”的地址。随着目标机代码的不断执行，链表上各“TB”之间的关联关系不断得到完善，指令集仿真的执行效率也会随之提高。但随着执行时间持续增加，也会引发另一个问题，就是翻译代码缓存空间会被占满。为了实现简便，一旦缓存空间被占满，QEMU 会将该空间全部清空[5]。

可见，当控制流从一个基本代码块跳转到另一个基本代码块时，若可以根据 TB 指针直接跳转执行，则可以节省很多上下文切换的开销，保持仿真系统的高效执行。但如果因为缓存空间不够导致存储在其中的 TBs 被全部清空或某个 TB 的被新的 TB 所替换，则直接跳转的链路将被打断。此时对于一个需要执行的基本块，仿真系统或者按照全系统查找的方法在缓存

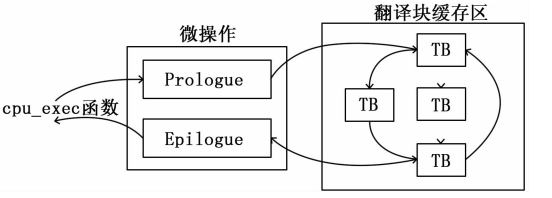

图 2　指令集转换原理

区中找到其对应的 TB 执行，或者在根本找不到对应的 TB 时，重新将该基本快翻译为 TB，存入缓存，并利用链表重新建立各 TB 间的关联关系，然后才能执行该基本块的代码。显然，此时仿真速度将大大降低。

为了改善上述问题，提出采用多级队列的缓存淘汰策略替代原有的缓存清空策略，实现提高指令集仿真速度的目的。

3　多级队列缓存淘汰算法实现

3.1　多级队列缓存淘汰算法分析

多级队列缓存淘汰算法的原理是根据访问频率将“TB”块划分至多个队列，不同的队列具有不同的访问优先级，优先缓存在目标代码执行过程中调用次数多的 TB。如图 3 所示，该算法的具体实现方法如下：

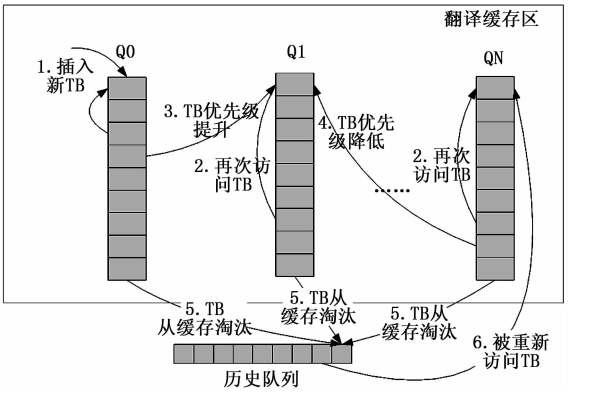

图 3　多级缓存淘汰算法示意

1）仿真系统调用某个“TB”后，将其插入队列 Q0。

2）针对翻译缓存中的每一个队列，根据其中挂接的“TB”块的历史访问记录实施淘汰策略，其原则是：如果某个“TB”块刚被调用过，则其被再次访问的概率也最高。实现方法如下：

（1）最新被调用的“TB”块插入链表头部；

（2）当队列中的某个“TB”块被再次调用后，该“TB”块被移至链表头部；

（3）当链表存满时，丢弃链表尾部的“TB”块。

3）当“TB”块被调用的次数到达升级门限时，将该“TB”块从当前队列删除，加入到高一级队列的头部；

4）为了防止高优先级“TB”块永远霸占最高优先级队列空间，当“TB”块在阈值门限时间段内没有被再次调用时，对其做降级处理，将其从当前队列删除，加入到低一级的队列头部；

5）当翻译缓存存满需要淘汰“TB”时，从最低一级队列 QN 开始按照步骤 2）中给出的策略淘汰“TB”；每个队列淘

汰“TB”时，将其从队列中删除，并将其索引加入历史队列头部；

6）如果某个“TB”在历史队列中被重新调用，就可以重新计算该“TB”块的优先级，将其移到优先级最低的队列头部；

7）历史队列按照同样按照步骤 2）中的策略淘汰“TB”块的索引。

多级队列缓存淘汰算法解决了翻译指令缓存占满后清空造成的指令仿真速度降低的问题，同时能够避免采用单队列的 LRU 算法造成的“缓存污染”问题，数据重用命中率大大高于 LRU 算法。但是鉴于多级队列缓存缓存淘汰算法需要维护多个队列，且需要维护每个数据访问时间，复杂度稍高，因此在具体工程实践中需要根据业务需要和对数据的访问情况适当选择队列数量。

3.2 算法实现

本文根据被仿真处理器的应用场景，即其应用在航天领域，其上运行的代码为星载应用软件的实际情况，选择采用二级队列实现多级队列淘汰缓存算法。

首先将整个翻译指令块缓存空间分割为两个队列，分割比例为 2：1。第一个队列长度占总缓存空间的三分之二，用于记录翻译指令块的访问历史。具体而言，当某个翻译指令块第一次被调用后，将其加入此列表。当此列表空间被占满而需要通过淘汰已有翻译指令块腾出队列空间时，则按照先进先出的原则丢弃此列表中被调用次数小于 35 的翻译指令块。反之，若此队列中某个翻译指令块的访问次数超过 35 次，则将该翻译指令块索引从当前队列删除，同时将其插入第二个缓存队列头部。

第二个缓存队列占总翻译指令块缓存空间的三分之一，用于根据历史访问记录存储翻译指令块。其排序原则是，总是保持最新被调用的翻译指令块排在最靠近链表头部的为止。如此，当此队列排满必须淘汰某个翻译指令块时，只需要淘汰排在队列最末尾的翻译指令块即可。

3.3 加速仿真实现

采用上节中介绍的多级队列缓存淘汰算法，在原有 QEMU 处理器仿真模型的框架下实现仿真处理器。为了实现加速仿真并准确计算仿真处理器与目标处理器之间的频率比例，对 QEMU 系统时钟进行分析。

QEMU 系统中有 3 个时钟，分别为 REALTIME、VIRTUA 时钟和 HOST 时钟。REALTIME 时钟是通过直接获取宿主机 CPU 时钟信息来计时；VIRTUAL 时钟通过仿真处理器执行的指令数来计时；HOST 时钟则是直接从宿主操作系统中获取时间信息。

通过实际测量的方法估计仿真处理器与目标处理器之间的运行速度比。采用 VIRTUAL 时钟记时测算仿真处理器的运行速率可以屏蔽宿主机本身其性能对测算结果的影响，因此采用该时钟进行运行速度比估算。设计一段由 CPU 计算密集型指令组成的测试代码（不考虑 I/O 指令对运行速度的影响）。在仿真处理器上运行该测试代码，同时在目标处理器上运行同一段测试代码，记录两次运行时间并计算二者之间的比值，将该值称为“最大加速率”。当用户在采用仿真处理器运行目标代码时，可以根据测试项目需要，在 1 到“最大加速率”之间任选一值。当选 1 时，表示仿真处理器的运行速度与目标处理器一致；当选“最大加速率”时，表示仿真处理器的运行速度为目标处理器的“最大加速率”倍；当选二者之间的任意其他值时，仿真处理器该值大小等比例放缓仿真处理器执行速度，实现按需模拟的目的。

4 实验结果

4.1 实验环境

仿真处理器宿主机 CPU 为 2.2 G 的 Intel（R）Core（TM）i7 处理器，目标系统为 Linux－13.04 版。仿真处理器基于 QEMU－ 2.6.1 版本，并在该版本基础上，采用多级队里缓存淘汰算法对 TB 缓存淘汰策略进行优化实现。

被仿真处理器为我国自主研制 BM3803 处理器，该处理器设计面向空间应用，是我目前国星载计算机的常用芯片，其主频可达 20 MHz，具有 32 位精简指令系统（RISC）[6]。芯片内部包含整数处理单元（IU）、浮点数处理单元（FPU）、独立的指令和数据缓存（Cache）、硬件乘法器、除法器等，指令分为加载/存储，算术/逻辑/移位，控制转移，读/写控制寄存器，浮点操作等指令。

选取采用国产 BM3803 处理器的某卫星中心计算机为实验对象，针对加载在其内部的嵌入式应用软件进行测试。测试环境构成如图 4 所示，由外围辅助测试系统和被测系统组成。外围辅助测试系统由测试激励模块和遥测处理模块组成；被测系统由仿真处理器及其外围总线、FPGA 模块等组成。测试时，采用测试脚本通过测试激励模块向被测系统注入指令或仿真数据，引导被测软件执行各项任务。

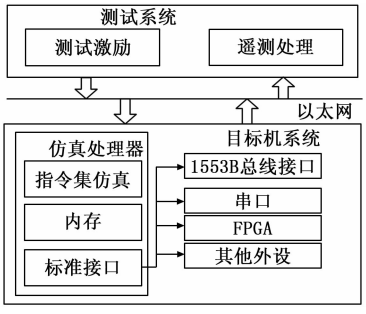

图 4 测试系统结构示意图

4.2 目标应用程序

选用某型号星载中心计算机应用程序作为目标应用程序。首先在硬件目标机上采用硬件处理器对目标代码的遥测、遥控、总线、内务、热控、能源等功能进行测试，然后在仿真处理器上加载相同目标代码，测试相同的功能模块。二次测试的测试激励完全一致，通过比较两次测试得到的测试结果能够证明仿真处理器实现的正确性和仿真加速效果的有效性[7]。

4.3 仿真结果

仿真系统初始化完成后，将某星载计算机嵌入式应用软件以二进制码流的方式读取至仿真系统，保存该文件的代码段、数据段和堆栈段地址等基本信息，并根据上述信息设置仿真处理器段寄存器，将堆栈指针移至虚拟 RAM 的起始位置，设置各虚拟寄存器工作模式，模拟真实硬件处理器的启动流程，引

导虚拟处理启动并开始运行[8]。

通过动态指令翻译模块接口统计了各目标功能模块的动态指令数和仿真时间，得到了采用多级队列缓存淘汰算法优化前后的仿真速度比较结果，如图 5 所示。从图中可以看出，采用优化算法后的仿真速度比优化前平均提高了 9.3 倍。

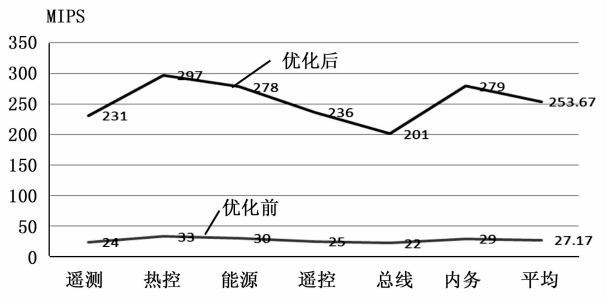

图 5　优化算法前后仿真速比对比

5　结束语

本文在开源软件 QEMU 的动态指令翻译基础上，提出了基于多级队列缓存淘汰算法的处理器全数字仿真优化实现方法，实现了国产 BM3803 处理器的全数字系统仿真。实验证明，仿真处理器加载星载嵌入式软件测试能够正确实现软件功能、性能需求，且优化后的指令集仿真算法比原方法的平均处理速度提高了 9 倍，为软件测试提供加速运行途径。以热控测试为例，采用本文方法实现的仿真处理器能够将热控测试周期从 7 天缩短为 1.5 天。大大提高了星载嵌入式软件确认测试的效率，也为缩短软件研制周期，提升软件研制质量奠定了基础。

参考文献：

[1] Shoemaker J，Wright M. 0rbital express space operations architecture program E [A]. International Society for 0ptical Engineering. 0rlando：The International Commission for 0Dtics [C]. 2003，5088：1－9.

[2] Austin T，Larson E，Enrst D. SimpleSealar：An infrastructure for computer system modeling. IEEE Computer，2002，35 (2)：59－67.

[3] Qeemu B F. A fast and portable dynamic translator [A]. Proc. USENIX Annual Tech. Conf. [C]. FREENIX Track，2005：41－46.

[4] Choi J W，Nam B G. Development of high performance space processor emulator based on QEMU－open source dynamic Translator [A]. Conference on Control，Automation and Systems，2012 12th International Conference [C]. 2012.

[5] Cmelik B，Keppel D. Shade：A fast instruction—set simulator for execution profiling [A]. Proc of SIGMETRICS Conf on Measurement and Modeling of Computer Systems [C]. New York：ACM，1994：128－137.

[6] Witchel E，Rosenblum M. Embra：Fast and flexible machine simulation [A]. Proc of the ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems [C]. New York：ACM，1996，24 (1)：68－79.

[7] 北京微电子技术研究所. BM3803MG 32 微空间处理器用户手册 [Z]. 北京：北京微电子技术研究所，2010.

[8] Marques P，Feiteirinha J，Pureza L，et al. cLeonVM：using dynamic translation for developing high－speed space processor emulators [A]. Proc. of the 9th International Workshop on Simulation for European Space Programmes (SESP'2006) [C]. Noordwijk，The Netherlands，2006.

[9] 张震波，杨鹤标，马振华. 基于 LRU 算法的 Web 系统缓存机制 [J]. 计算机工程，2006，32 (19)：68－70.

[10] 黄　晨，汪文明，张义超，等. 基于国产 CPU 的并行冗余计算机系统研究 [J]. 计算机测量与控制，2017，25 (7)：257－259.

(上接第 133 页)

软件测评中心 C 应用其全数字仿真方式，模拟被测系统的硬件环境，使软件测试工作尽早介入。在 S 阶段，通过全数字仿真与快速原型系统 (RCP) 相结合，并在此基础上加入多学科仿真，完全满足了软件验证的需求，并缓解了测试环境资源冲突的问题[8]。在 D 及 P 阶段，前期采用全数字仿真环境，将大部分软件功能验证工作在虚拟仿真条件完成，后期的软件确认测试、软件定型则采用半实物仿真方式或者在真实机载环境中开展。经验证，此验证平台的搭建与之前软件设计、开发与验证脱离的模式相比，极大地提高了软件验证效率以及软件质量安全性[9]。

7　结论

用全数字仿真技术及数据分发服务，建设全数字航空机载软件验证平台，可以大大缓解企业的综合试验台的压力，使得企业内测试资源得到最大程度的合理利用；可以收集型号软件测试过程数据，建立型号软件验证知识经验库；对成品单位及软件测评单位，可开展多地远程协同软件验证。这对飞机设计所提升软件管理水平、研制能力，优化测试资源配置有着深刻的意义。

参考文献：

[1] 蒲小勃. 现代航空电子系统与综合 [M]. 北京：航空工业出版社，2013.

[2] 陈真勇，唐　龙，唐泽圣，等. 以鲁棒性为目标的数字多水印研究 [M]. 北京：北京化学出版社，2002.

[3] 卞华星，陈　丹，庄　毅. DDS 在飞机协同设计系统中的应用 [J]. 计算机与现代化，2015 (5)：35－39.

[4] 任昊利，等. 数据分发服务——以数据为中心的发布/订阅式通信 [M]. 北京：清华大学出版社，2014.

[5] 蔡建平. 嵌入式软件测试实用技术 [M]. 北京：清华大学出版社，2010.

[6] 郑　凯. 飞机综合航电系统集成验证平台互联技术研究 [J]. 航天科学技术，2015 (36)：43－44.

[7] 韩德强，冯云贺，王宗侠，高雪园. 基于 Simics 的嵌入式系统的研究与开发 [J]. 航天科学技术，2015 (2)：32－34.

[8] 杨广华，齐　璇，施寅生. 基于场景模式的嵌入式软件测试用例设计 [J]. 计算机工程，2010 (15)：89－91.

[9] 熊志刚，李　晶，苏振扬，等. 一种 DDS 与 ESB 通信转换的适配器模型 [J]. 计算机工程，2015 (2).