维普资讯 http://www.cqvip.com

计算机科学 2000Vol. 27№. 12

嵌入式系统中 Java 的硬件支持策略*)

The Strategy of Java's Hardware Support in Embedded Systems

李宗伯 胡守仁
(国防科技大学计算机学院 长沙 410073)

Abstract This paper first analyzes the inside drive of support in hardware for Java virtual machine in embedded Java systems, and the basic requirement and design principles, then it introduces 5 kinds of designs and their primary technology features, and analyzes their chief motivations, advantages and disadvantages, and the possible fields in which they can be applied. Based on all these, the paper presents our Java chip JC401 in which JIT compiler is used, and examines its technology details. At last, the paper makes a simple analysis and summary on all these designs.

Keywords Java virtual machine, Java chip, Hardware support, Just-In-Time compiler

一、概述

近来,具有平台无关、面向对象等诸多特点的 Java 语言得到了迅速的发展,以即时编译技术(JIT)为代表的软件实现的 Java 虚拟机在桌面系统中日趋成熟,在运行速度、内存需求、实时性能等方面都有较大的提高,以网页中的 Java 小程序为主的 Java 应用大量涌现,具备了一定的应用基础。

另一方面,随着 Internet 技术的迅速普及,已经出现并将出现更多的新型消费类设备,满足人们随时随地访问 Internet 的需求,如便携式的电子邮件接受器、网页浏览器、Web 电话、WebTV,以及集成了 Internet 访问功能的移动电话等,同时还出现了以 Java 为基础的分布计算技术(Jini、Java space),可以动态执行流动 Java 程序的网络设备等,这些新型设备代表了一种强劲的发展趋势,在这些新型设备中,将不可避免地执行数量不断增加的 Java 程序,但是它们却具有许多不同于桌面环境的特点,如资源、速度、功耗受限以及更强的实时性要求等,目前在桌面系统中采用的日益复杂的高性能微处理器芯片和行之有效的 Java 虚拟机的软件实现方法显然不能适应这种环境,为了满足这些新的需要,采用专门针对 Java 设计的新型微处理器是一种新的解决办法,目前就已经出现了多种 Java 芯片。

本文介绍了目前已经出现的几种面向 Java 的处理器芯片,分析了它们的设计策略、主要技术特点、优缺点与适用范围,在此基础上提出了我们自行设计的一种采用 JIT 技术的 Java 芯片—JC401—的设计思想,分析了它的主要技术特点,最后进行了总结与归纳。

二、是否需要硬件支持

在讨论专门的 Java 支持硬件之前,回顾一下历史是有益的,在 Java 出现之前,也不乏许多类似的优秀语言,如 Smalltalk、self 等,曾风行一时,并出现了支持虚拟机的硬件,但它们并没有成功,平台无关方面的尝试也以失败而告终,P-code 虚拟机采用 Pascal 程序编译成的中间代码为虚拟机的指令集,其思想与 Java 虚拟机有惊人的相似,但是这种虚拟机与相应的支持硬件一同消失了。在 PC 时代,由标准控制者制定游戏规则似乎已是一条金科玉律,摧毁了无数优秀的产品与思想,Java 能例外吗?

Java 却是幸运的,因为时代已经变了:Internet 的迅猛发展使之成为一种新的先进的基础媒体,吸引了大量的资金和庞大的用户群,不同知识背景、不同工作环境下的群体都需要以他们自己要求的方式来获取信息,多样化的需求必然要求多样化的产品,它们又要能统一在 Internet 这种多平台的环境中,Java 无疑满足了这种需求,"Write once, run anywhere(一写永逸)"的精髓使它迅速发展壮大,成为 Internet 上一种重要的编程语言与运行环境。坚实的应用基础,再加上多样化的需求,出现支持 Java 的硬件是理所当然的。同时,

*)本项目获国家自然科学基金资助。李宗伯 在职博士生,主要研究方向为 Java 技术、Java 硬件支持、Internet 技术。胡守仁教授,博士生导师,主要研究方向为计算机体系结构等。

维普资讯 http://www.cqvip.com

由于微电子技术的不断发展，计算能力不断增强，价格不断下降，使得嵌入式产品开始有能力来考虑平台无关所带来的巨大好处，用较少的硬件代价、合理的资源、较低的功耗和有限的计算能力来换取对 Java 的支持是非常有价值的。事实上，支持 Java 的硬件已经出现，因此，现在的问题不再是是否需要支持，而是如何更好地支持。

三、已有的硬件支持方案

从体系结构设计的观点出发，对 Java 虚拟机的硬件支持需要考虑的就是如何以较少的硬件代价和功耗来较好地满足目标产品的需求，达到较高的性能价格比，其中有两个重要的考虑因素：需求与性能价格比。不同的需求将会有不同的设计考虑，而性能价格比则是在满足需求的基础上评估设计质量的重要标准。

对于 Java 支持硬件，其主要的需求是什么呢？由于桌面系统已经基本上被 Wintel 标准所垄断，而且其 Java 性能已能满足需要，基本上不存在设计支持硬件的市场动力；对于层次更高的工作站、服务器乃至大、巨型机而言，它们目前虽然也使用 Java，但是还不是其关键性的任务，对 Java 性能的要求并不迫切，因此在 Java 取得更大的发展前，这些领域不太可能存在对专门硬件支持的需求；由于 Internet 连接的嵌入式设备较多地使用 Java，其中主要是网页浏览中的 Java 小程序，随着 Java 的进一步发展，各种 Java 应用程序将会大量使用，而在面向个人的便携式 Internet 访问设备、其他消费类电子产品以及瘦型客户机等系统中，由于市场价格、功耗、尺寸等方面的限制而不能提供很高的速度和足够的资源，套用桌面系统中使用的软件解释或即时编译的方式不再适用，因此存在设计 Java 支持硬件的需求，使之能以合理的价格、功耗、尺寸获得较好的 Java 性能。

在这些产品中，根据应用领域的不同，对 Java 性能的要求情况也不相同，除了不断增长的 Java 性能需求外，还需要适当考虑性能较为关键的本地程序，其中包括高效的系统程序、媒体处理能力等，如一个具有 Internet 访问能力的移动电话，Java 在浏览网页、运行小型应用程序时会被使用，而通讯信号、协议以及话音的处理也是很关键的任务，因此，除了上述这些考虑因素外，在设计 Java 支持硬件时还必须权衡 Java 性能与本地任务性能方面的支持程度，在下面介绍的几种 Java 硬件支持方案中，可以清楚地看出它们在这些方面所作出的选择。

1. picoJava

picoJava 是 SUN 公司设计的一种可以直接执行 Java 字节码程序的处理器内核，它有 I、II 两种设计[2]，这里主要是指 picoJava-II 内核，基于该内核已经生产出了包括 MicroJava701(SUN)、MJ501(LG)等处理器芯片，它的主要技术特点有：

(1)直接支持 Java 虚拟机的绝大部分指令，较为复杂的指令通过微码来实现，很复杂的指令则通过软陷入来处理，使 picoJava 可以直接运行字节码程序。

(2)为了在运行系统程序及一般的应用程序时具有较好的性能，picoJava 提供了多达 115 条的扩展指令，从而使得它的非 Java 程序能达到与传统 C++程序接近的性能，并能满足对硬件的直接访问。

(3)使用了 64 个 32 位的环形寄存器作为栈 cache，并有专门的硬件来管理这个环形栈，直接支持 JVM 的堆栈结构，并能随机访问。

(4)在采用环形寄存器的基础上，在解码段引进了指令折叠逻辑，最多可以把四条相邻指令折叠成一条指令，可减少大约 28%的指令数目，改善堆栈指令的性能。

(5)对运行时系统的支持：为支持分代式的垃圾回收，提供了 write-barrier 硬件支持及与之配套的灵活的分段能力来定义大量的内存段的边界，同时还对线程管理、对象处理提供了一定的硬件支持。

从这些技术特点可以看出，picoJava 的设计目标偏重于最优的 Java 性能，同时还兼顾到一定的本地程序的性能，picoJava 从指令集、寄存器组织、解码能力等方面来提高程序本身的性能，同时还对 Java 虚拟机特有的一些性能关键的部分进行了一定的硬件支持，从而达到较好的 Java 性能。这种取舍策略与 picoJava 的设计目标有关，根据 Sun 的资料，picoJava 面向那些需要经常运行大量 Java 程序，需要较好 Java 性能的产品。但是，由于对 Java 的全面支持及对本地程序的考虑，使得 picoJava 较为复杂，具有较高的价格与功耗，限制了它在嵌入式系统中的应用。由于业界对其产品反应冷淡，目前 Sun 公司已宣布自己不生产基于该内核的芯片，并将其设计原码免费公开。

2. TinyJ

TinyJ 是 Advancel logic 公司设计的一种嵌入式处理器内核[3]，专门为 Java 进行了优化，目前设计了 Tiny²J、TinyJ、TinyJ^{DSP}系列内核，都是在 TinyJ 的基础上根据本地程序性能的需要加以适当修改而成，它的主要技术特点是：

(1)双语言支持：TinyJ 以具有较强性能和较好代码密度的 RISC 内核为基础，通过增加一个流水线段来完成 Java 字节码的解码，从而支持字节码的直接执行；

(2)堆栈与寄存器共存：为了保证 RISC 内核的高性能，TinyJ 内核保持了寄存器的组织结构，同时，为

维普资讯 http://www.cqvip.com

了降低字节码的解码复杂度，加入了简单的堆栈支持。

TinyJ 的设计思想比较注重对本地程序的支持，保持了传统 RISC 处理器的优点及其成熟的技术与应用基础，在此基础上以简单的支持获得一定的 Java 处理能力，虽然其价格、功耗等都能符合嵌入式系统的要求，但是它的 Java 性能比较低，仅能满足一些较少使用 Java 应用程序或对 Java 性能要求不高的产品的需要，目前主要用于 Java Card 市场。

3. GP1000

GP1000 是瑞典的一家名为 Imsys AB 的公司设计的一种具有可改写微码的微处理器芯片[4]，它可以在基本硬件功能的基础上，通过对微码的编程创建出满足实际需要的指令集，它的主要技术特点是：

(1)可改写微码：通过微码编程来实现所有 Java 虚拟机指令，同时还通过微码编程来实现一些虚拟机的运行时功能，如垃圾回收、线程调度等。

(2)多寄存器组，支持快速现场切换：有 8 组 8 个 8 位的寄存器组。

GP1000 的设计目标更注重灵活性：通过对特定工作负载的分析统计，设立专门的指令来支持使用频繁、性能关键的操作，从而获得较好的性能。对 Java 而言，只是简单地实现指令集及部分运行时功能，这有点象实现了一个软件解释器，解释程序放于微码 ROM 或 RAM 中，只是用微码级编程代替了机器语言级的编程。因此，Java 性能必然不高，同时也没有消除解释程序本身的内存开销(用片上微码 ROM 和 RAM 来存放)，36K 字节的片上微码 ROM 及 18K 字节的片上微码 RAM，无疑降低了性能价格比、增加了系统功耗。因此，这种芯片作为 Java 支持芯片而言并不具备太大的优势。

4. PSC1000

PSC1000 是 Patriot Scientific 公司在现有芯片的基础上，通过软件在代码验证时把 Java 字节码翻译成本地代码来支持 Java 字节码程序的运行[5]，精确地说，它是一种具有有限硬件支持的即时编译器执行模式，通过对 Java 虚拟机的一定的支持来简化 JIT 的复杂度，并在一定程度上消除一般 JIT 较高的内存开销。主要技术特点如下：

(1)双栈结构的堆栈机：操作数栈用来存放一般运算的操作数和运算结果，局部寄存器栈被用来存放方法调用时的返回地址及局部寄存器数据，较好地支持了 Java 虚拟机的堆栈结构，因此可以较容易地把字节码翻译成本机代码。

(2)简单的设计：没有采用流水技术，设计简单，从而使价格及功耗都比较低。

PSC1000 因为它本身采用堆栈结构及相当简单的设计思想，因此它的价格很低，但是 Java 和本地性能都比较低，适用于一些需要较低计算能力的应用场合。

5. JEM1

JEM1 是 Rockwell Collins 公司在其现有芯片结构 AAMP(Advanced Architecture Microprocessor)的基础上改造而成，由于 AAMP 的堆栈体系结构与 Java 虚拟机的基本一样，因此通过适当的改造，并编写一些相关的微码程序，最终使它可以直接执行 Java 字节码程序，同时为便于访问和管理系统资源，还增加了一定的扩展字节码。该芯片并未商业化，许多细节并不清楚，但是总的来说，它与 PSC1000 的设计思路有相同之处，都是基于一种现成的堆栈式嵌入芯片，对设计进行适当修改或通过软件辅助而达到执行字节码的目的，具有较低的价格和功耗，但本地和 Java 性能较低。

6. 讨论

根据 Java 语言及虚拟机的特点来考虑硬件支持方案是很重要的，首先，Java 虚拟机是基于堆栈的，由于存在栈顶瓶颈，会造成大量的数据相关及程序的低效，据统计，采用堆栈会造成 30%左右的指令增加。在实现时为了提高效率，必须有效去除这些增加的指令，在所有 5 种设计中，仅有 picoJava 考虑了这一点，但是，它采用的指令折叠技术增加了设计的复杂度。另一方面，当采用流水线结构时，需要减少数据相关，才能提高流水效率，在所有采用流水技术的设计中，由于它们直接执行 Java 字节码程序，不可能修改指令序列来消除数据相关，从而造成流水效率低下。

其次，Java 虚拟机指令集具有比例不大但性能关键的复杂指令，本来，在字节码中引入面向对象等复杂指令的出发点是考虑解释执行的高效性及对 Java 语言的直接支持，但是因为它们过于复杂，不可能用硬件直接实现，一般的办法是采用软陷入或微码实现来解决。但是，软陷入模式会较大地降低系统性能。相对软陷入模式而言，采用微码虽然能在一定程度上提高关键指令的性能，但是需付出价格、功耗方面的代价，使系统更加复杂。在上述 5 种设计中，均在不同程度上采用这两种实现方法。

最后，由于 Java 虚拟机具有面向堆栈、面向对象等特点，造成它的性能不如传统的 C/C++程序，更比不上传统的汇编语言程序，而在嵌入式系统中，一些关键的系统程序和实时性要求较高的功能是无法用 Java 来替代的，因此，在 Java 芯片中还必须考虑到一定的本地性能的支持。但是，如果既考虑对 Java 进行较好的支持，又考虑较好的本地性能的话，会使整个系统复杂化，如 picoJava 为了支持本地程序，增加了 115 条指令，使得总的指令条数达到 341 条，造成系统过于复杂。

维普资讯 http://www.cqvip.com

从上面的分析来看，似乎存在这样一种矛盾：过分强调 Java 支持，则会造成芯片复杂化、本地性能降低等问题，还可能由于指令相关、采用微码或软陷入支持性能关键的复杂指令而达不到较好的整体效果；如果强调本地性能，忽视 Java 支持，则会因为堆栈的低效性和性能关键指令的低效性而使 Java 性能大大下降，达不到应用的需求；若两方面都进行较好的支持则会造成设计过分复杂，不适合嵌入式应用的基本要求。

那么是否有一种较好的方案来解决所有这些问题呢？对桌面系统中的 JIT 技术进行适当精简，并提供一定的硬件来支持 JIT 及 Java 性能关键指令是一种较好的答案。JIT 在运行时把需要使用的方法翻译成本地机器代码，由于它是一种软件，因此可以灵活地处理各种情况：对于堆栈造成的指令增加，在非堆栈机器上可以被 JIT 很容易地去掉；对于堆栈造成的大量数据相关，可以通过简单的相关性分析及指令调度来有效减少数据相关；针对 Java 字节码中的复杂指令，JIT 通过指令分解，并配合一定的运行时信息，可以用其他指令有效地实现这些复杂指令，如 lookup 指令可以生成简单的条件转移指令，对象访问指令可以用 Load/Store 指令高效实现，方法调用指令可以用几条指令来完成，如果还能对这些指令予以适当支持，就可以更加有效地支持这些复杂指令；最后，在 JIT 中，本地性能的提高也意味着 Java 性能的提高，合理的 Java 支持既能大大提高 Java 性能，也能加强本地程序性能，还可以有效控制系统复杂性。但是，在考虑 JIT 的优点时，不能忽视它带来的新问题，JIT 本身的运行时间会占用总体运行时间，同时还可能造成内存的巨大开销，这是 JIT 在嵌入式系统中应用的两大障碍。

四、JC401

基于上面的分析，为了得到较好的 Java 性能，同时又能满足嵌入式系统对价格、功耗及本地性能关键程序的要求，我们设计了一种 Java 芯片内核 JC401，目前，该内核的 VHDL 模型已经完成，并通过了正确性测试及单项性能测试，效果较好。它的主要设计思想是：在具有较好性能的 RISC 结构的基础上，对 Java 虚拟机进行简单有效的支持，通过这些支持提高 Java 复杂指令的执行速度，有效简化 JIT 的复杂度，大大降低内存开销，从而排除了 JIT 在嵌入式系统中应用的两大障碍，有效解决了在已有设计方案中存在的各种问题。它的主要技术特点是：

(1)典型的 RISC 结构　JC401 采用了典型的 32 位 RISC 指令集作为基础，它有 70 条 32 位 RISC 指令，使用取指、解码、执行、访存、写回五段标量流水，全部采用硬连线逻辑，绝大部分指令可在一拍内执行完毕，由于 RISC 结构具有简单，高效等特点，JC401 以它作为基础，则能降低系统复杂度，降低 CPI，从而获得较好的性能价格比及较低的功耗，使之符合嵌入式系统的基本要求，然后在尽量保持这些优点的前提下，进行适当的 Java 支持。

(2)支持堆栈结构　典型的 RISC 结构一般采用通用寄存器堆的形式来支持对常用变量的高效访问，也有一些 RISC 芯片采用寄存器窗口的组织形式来加速方法调用与返回的速度，减少中断响应延迟。研究表明，通过适当的编译优化技术，可以达到通用寄存器组织方式的性能。在 JC401 中，为了支持 Java 虚拟机面向堆栈的结构，充分利用寄存器资源，限制寄存器地址长度，采用了可变长寄存器窗口的组织形式，并且把每个寄存器窗口的寄存器数目限制在 32 个以内，寄存器窗口的起始位置即为 Java 虚拟机中的局部变量起始地址，也是方法框架的起始地址，窗口上限指针相当于栈顶指针，可以动态浮动。这种组织方式不但提高了寄存器利用率，同时还有效地支持了 Java 虚拟机中的堆栈访问；寄存器的数目可以根据需要设计为 32、64 或更多。对于每个寄存器窗口中的寄存器访问，由指令指定相对于起始寄存器的相对偏移地址，再由相关硬件根据起始寄存器的值计算出实际寄存器地址。为了解决寄存器上下溢出的问题，采用硬件根据设计条件来自动处理。

(3)支持性能关键指令　由于在 JC401 中采用了 JIT 编译器，绝大部分框架内的移动指令可以取消，所有计算、分枝、内存访问指令一般都可以用一条指令来实现，大大提高这些比例较大的指令的性能，对于其他性能较关键的复杂指令也进行了有效的支持。

对于方法调用与返回指令，JC401 增加了 32 个 32 位寄存器组成的系统栈，可以在方法调用时把 3 个 32 位的系统环境变量在一拍内保存，并在返回时在一拍内恢复保存的环境变量，该系统栈还可以通过 Push/Pop 指令来使用栈顶的 32 位寄存器。同时还把具有同步特性的方法分离成监视器进入和方法调用两条指令，降低硬件支持调用指令的复杂度。最终使得 JC401 的方法调用指令可以在一拍内完成，Java 的非同步方法调用在解析后，包括目标地址的生成指令，一般可以在 4 拍内完成(picoJava 的 invoke_static_quick 需要 11 拍、invoke_super_quick 需要 21 拍，invokevirtual 需要更多的拍数)，大大提高了这类指令的性能，同时受益的还有中断处理及软陷入指令。

此外，还在解码段对引用非空检查进行了支持；设置了基于索引的访存指令支持数组元素访问；对性能较关键的线程同步操作设置了监视器指令及监视器 cache 寄存器，有效提高了监视器操作的速度。所有这

维普资讯 http://www.cqvip.com

些，可以用较少的代价来有效提高性能关键部分的性能。

(4)采用指令压缩技术　除了70条32位指令，经过对典型Java程序的统计，它们使用的指令中绝大部分涉及到的变量相对框架起始地址偏移量较小，涉及到的常数也比较小，因此，我们设计了16条16位指令，为使用频率最高的32位指令设置了16位指令，通过指令的最高位来与32位指令区分，并在取指段加入适当支持来处理这种变长指令，由于区分容易，实现代价不高，却能有效缩短JIT编译后的目标代码长度，减少内存开销。通过对一些典型程序的统计，翻译后指令长度与字节码长度之比为0.8左右，而在一般RISC结构中，这个比例最高可达3.0，体现了该技术的优越性。

(5)支持指令修改　由于Java的多态性，一些指令的运行需要到实际执行时才可以知道细节，为提高翻译后代码重用性能，通过支持指令修改技术，达到优于Sun公司的quick指令的效果。

(6)支持即时编译器　由于JC401具有与Java虚拟机一致的堆栈结构，所以在进行代码验证时，利用验证生成的中间信息就可以很简单地完成代码翻译工作，从而可以去掉复杂的算法，如寄存器分配算法、方法内联优化等，它们需要耗费较多的运行时间，还需要占用大量的内存空间，大大减少了JIT的运行时间，节约了大量运行这些算法造成的内存开销；对于桌面高级JIT中使用的一些垃圾回收支持、方法调用次数减少及其它复杂的优化手段，则通过一定的硬件支持，选用适合嵌入式系统的垃圾回收算法等手段予以去除或简化。

同时，由于JC401采用了针对Java优化的指令压缩技术，使得生成后的代码长度甚至小于编译前代码，可以节约空间，提高编译后代码执行效率，同时，这也使得类库静态与编译成为可能，大大减少了需要即时编译器编译的方法数，也提高了类库方法编译后代码的性能。

为了进一步减少引入Java造成的内存开销和性能损失，我们还参考了Sun在Java 2 ME平台中使用的技术，一方面根据应用的范畴划分出不同的配置，在这些配置中使用最需要的基本类库、底层API及语言功能，同时还减少基本类库及底层API的层次，采用较多的本地程序取代Java方法，从而提高它们的性能、减少内存开销。

所有这些，既能克服一般JIT算法复杂、内存开销大的缺点，又能发挥JIT编译代码高效、灵活的优点，使JC401具有较好的Java性能和本地处理能力。

结论　综上所述，我们认为，TinyJ、PSC1000、GP1000、JEM1价格与功耗较低，但Java性能较低，仅适合要求较低的嵌入式应用；picoJava内核具有较好的Java性能及一定的本地处理能力，但是价格与功耗都比较高，不太适合于一般的嵌入式应用；我们设计的JC401由于成功地引入了即时编译技术，具有简单的结构，较低的价格和功耗，较好的Java和本地处理能力，适合于中、高端嵌入式市场。为了满足嵌入式领域不断增长的对数字信号处理能力的需求，在将来的设计中还可以较容易地融合DSP处理能力，使之更适应嵌入式市场的需要，具有较好的应用前景。

参考文献

1 Michael J, et al PicoJava-I: The Java Virtual Machine in Hardware. IEEE Micro, 1997, 17(2): 45～53

2 McGhan H, et al. PicoJava: A Direct Execution Engine For Java Bytecode. IEEE Computer, 1998(Oct.): 22～30

3 Java-enabled Embedded Applications--The Wave of the future is here, White Paper of Advancel Logic Corporation, 1999

4 Halfhill T R. GP1000 Has Rewritable Microcode: [Microprocessor Report]. 1998, 12(17): 1～4

5 PSC1000 Microprocessor's Data Book. Patriot Scientific Corporation