计算机科学 2001Vol. 28№. 7

智能卡的新发展——JavaCard 技术综述

The New Development of Smart Card——An Overview to JavaCard

李增智 李 钢 韩 冬 王志文

(西安交通大学系统结构与网络研究所 西安 710049)

Abstract Smart card is currently a popular embedded device, but it has some drawbacks. On the other hand, Java's reliability, cross platform support, and many other advantages make it fit into embedded systems. The paper discusses every aspect of JavaCard, the combination of smart card and Java technology, form which, we can conclude Java has great applied potential in embedded devices

Keywords Embedded system, Smart card, Java, JavaCard

一、前言

随着 CPU 处理功能的日益强大、存储设备体积越来越小、价格的普遍下降以及安全技术的日趋完善，具有内嵌处理器的设备变得很普遍。在这些设备中，智能卡是最为流行的，它是一种具有内嵌处理器芯片的信用卡，主要应用在诸如电信、传输、银行和医疗保险等领域中。智能卡是通过编程来执行任务和存储信息的，但当前主要是通过 C 或汇编语言来实现的，所以在开发和维护上具有一定的难度，而且还缺乏灵活性，这些都是不利于智能卡得到广泛应用的主要障碍。另一方面，Java 语言轻量级的字节码、在网络上移动对象的能力、高度可移植性以及完善的安全机制都使得它非常适合于作为智能卡的编程语言。JavaCard 就是这二者结合的产物，它是一种典型的智能卡，遵从所有的智能卡标准，但它在智能卡 ROM 中实现了一个 JVM，该 JVM 将执行一个 Java 字节码的子集，提供外部可以访问的功能，负责控制对智能卡资源的访问(如内存和 I/O)。与以前的智能卡技术相比，采用 JavaCard 技术的智能卡可以采用 Java 进行编程，因此具有高度的可移植性和安全性。更为重要的是，在卡被发送到用户以后，应用程序还可以被动安全地下载到卡中，从而动态地扩展了智能卡的功能。下面，我们将分别讨论 JavaCard 的历史、需要解决的关键问题、技术基础、相关规范、体系结构以及安装和运行过程。

二、JavaCard 的历史

JavaCrad 的思想并不是在今天才提出的，早在 1996 年，一个著名的智能卡生产商 Schlumberger 就演示了一个基于 Java 的智能卡。根据 Schlumberger 的实验，Sun 公司于 1996 年 10 月发布了第一个 JavaCard 规范，该规范主要描述了 JavaCard 的总体目标和体系结构，即支持 Java 语言的子集并为智能卡的特定功能(如加密)提供 API。接着，Schlumberger 和 GemPlus 于 1997 年 2 月成立了 JavaCard 论坛，并在 1997 年年底发布了 JavaCard2.0 规范，JavaCard2.0 规范包含了更多的关于 API 规范和 JavaCard 虚拟机(JCVM)的技术细节，其中 JCVM 是一个更为精致和简单的 JVM，它去掉了很多标准 Java 的特点。然而，由于当时 Java 本身并不具备满足嵌入式系统开发要求的基础结构，同时相关的 API 规范描述得比较粗糙，因此不同 JavaCard 上的程序缺乏移植性和互操作性，将一个稍微复杂的 JavaCard 应用程序移植到另外一个 JavaCard 上几乎是不可能的。

三、关键问题

由于智能卡发行数量巨大，成本费用是一个非常关键的因素，这就决定了智能卡中的硬件资源是非常有限的，一般而言，一个最普通的智能卡具有 256 字节的 RAM，16K 字节的 ROM 和 4K～8K 字节的 EEPROM；而最高档次的智能卡当前也只有 4K 字节的 RAM，64K 字节的 ROM 和 64K 字节的 EEPROM；更多智能卡的硬件配置位于二者之间。另一方面，智能卡对安全性、执行效率、事务处理等方面的要求也是很高的。因此，在保证 Java 基本特点不变(如跨平台、安全、面向对象等)的前提下，如何开发出满足智能卡需求的精致、简单、高效的 Java 基础结构是摆在业界面前的一个关键问题。

四、技术基础

经过业界的不懈努力，用于嵌入式系统(包括智能卡领域)的 Java 技术取得了惊人的进步，这表现在以

下几个方面：

4.1 Java 字节码执行技术

JVM 的软件实现需要一些方法，以便将 Java 字节码解释或翻译为本地代码，为做到这点，当前有如下技术可以执行 Java 字节码，包括：Java 解释器、"及时"翻译器以及静态编译器。其中，解释器是直截了当的，它在运行时简单地装载每个 Java 操作码并执行相应的操作，但这种方法的效率是很低的；"及时"翻译器克服了低效问题，它在执行具体操作之前，先将接收到的 Java 字节码一次性地翻译为本地代码，而且它还解决了 JVM 设计中的一个关键问题：动态类装载。当然，使用动态类装载需要耗费大量的处理器时间和内存，在智能卡中这应该是绝对避免的；另一方面，静态编译器可以接收 Java 原代码或中间字节码并产生特定平台的本地代码，这与常规的 C 编译器是类似的。

4.2 picoJava 技术

picoJava 是由 Sun 近来推出的最令人激动的技术之一，使用该技术可以在硬件上直接执行 Java 字节码[1]。picoJava 的核心是使用 Verilog 注册传输语言(RTL)开发的，这是一种用于数字逻辑设计的工业标准语言。设计 picoJava 的目的主要用于提高性能，它支持硬件浮点运算、一个六级指令流水线、指令/数据缓存和一个堆栈管理单元。picoJava 的出现对传统的用软件实现的 JVM 而言是一巨大挑战，实验表明，在给定的时钟频率下，picoJava 在性能上已经超越了传统的软件 JVM。在实际的智能卡设计中，picoJava 往往是和软件配合使用的，这是因为有些操作(如实例方法调用、异常处理、垃圾回收等)纯粹用硬件实现代价太高，除了标准的 Java 字节码之外，picoJava 还可以执行所有的快速字节码和扩展字节码。

4.3 并行垃圾回收器

传统的 JVM 都使用一个标记/清除垃圾回收器，这种回收器在垃圾回收阶段将悬挂起用户进程，因此可能造成无法忍受的延迟。先进的并行垃圾回收器将持续操作，并只在很短的时间内挂起用户进程，因此是非常适合嵌入式实时系统的。

4.4 硬件设备的访问

嵌入式系统中的很多硬件部件(如串口、通用接口等)都在处理器的地址空间中有一个相应的内存地址，对这些部件的访问需要访问这些特定的内存位置，但 Java 规范并没有定义可以直接访问内存的跨平台的 API。为此，JavaSoft 在 Java 操作系统中，定义了一个包含本地方法的类，这些本地方法可以高效地进行内存访问和数据处理。

4.5 精简的 Java 类库

JDK1.1 和 JDK1.2 都需要占用几兆以上的空间，这对桌面系统算不了什么，但对智能卡而言却是一个巨大的资源，因此是不能接受的。唯一的解决办法就是精减传统的类库，使之适合智能卡有限的硬件资源，当前，Sun 和 JavaCard 论坛已经对标准 Java 作出了适用于 JavaCard 的修改，如不支持几种数据类型(如 Float、Double 和 Long)，不支持线程，而且，JavaCard 中的字节码也与传统的字节码有所不同。

五、相关规范

目前 JavaCard 的设计遵循两个重要规范：OpenPlatform 和 JavaCard2.1。

5.1 OpenPlatform[2]

OpenPlatform 是由 Visa 在 1998 年 4 月开发的一个规范[3]，规定了在智能卡上安全下载和安装应用程序的标准。在该标准中，OpenPlatform 为智能卡上不同的软件提供者指定了如下角色：智能卡操作系统提供者、卡的生产者、卡的发布者和卡上应用程序提供者。不同的角色具有不同的权限。对于卡的发布者而言，一旦拥有了智能卡，他就拥有了一个被称为"卡域"的钥匙，用此钥匙，他可以定义卡的功能和特性，如可以决定是否下载一个新的应用程序，是否启动某个应用程序等。为了允许动态下载应用程序到卡上，OpenPlatform 给每个卡上应用程序提供者分配了一个被称为"安全域"的钥匙，"安全域"钥匙的安装由卡的发布者完成，但他并不知道具体应用程序的内容(由于不拥有相应的"安全域"钥匙)。通过这些手段，保证了当卡被发布到用户手中后，只有卡发布者可以改变卡的功能。目前，OpenPlatform 已经成为管理多应用程序智能卡的事实上的标准。

5.2 JavaCard2.1[4]

JavaCard2.1 规范是在 1999 年 3 月发布的，它提供了 JavaCard 字节码验证方案，这点正好弥补了 OpenPlatform 的不足。JavaCard2.1 支持一个被称为"代码签名"的安全模型，这意味着在使用由 OpenPlatform 规定的安全下载应用过程下载应用程序之前，必须先将程序代码经过安全的转换、评估和签名。JavaCard2.1 引入了一个新概念：软件防火墙，此时，对每一个对象访问，都要检验相应的访问权限。另外，JavaCard2.1 规范删除了文件系统 API，并将整个加密 API 扩展为大量的类。

六、JavaCard 体系结构

符合 JavaCard2.1 的 JavaCard 体系结构如图 1 所示。其中，JavaCard ROM 中的最底层代码是访问存储器(包括 RAM、ROM 和 EEPROM)和 I/O 的设备驱动程序，根据需要，也可能包括访问加密处理器的驱动

程序，这些驱动程序都是用 C 或汇编语言实现的，从而大大提高了 JavaCard 执行效率。在这之上，就是 JavaCard 虚拟机(JCVM)，它是传统的 JVM 的简化版本，将负责控制上层应用程序对 JavaCard 硬件驱动程序的访问。JCVM 之上就是实现了 JavaCard2.1 和 OpenPlatform 规范中所定义的各种 API 的 Java 中间字节码。最后，各种实现了 JavaCard 专用功能的应用程序位于最上层。需要注意的是，在 JavaCard 中，各种上层应用程序都被称为 applet。

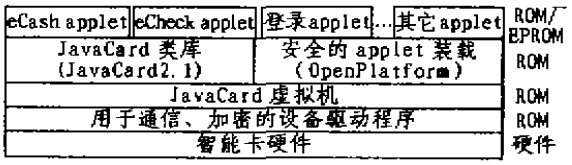

图 1　JavaCard 体系结构

七、JavaCard 中 applet 的安装和运行过程

7.1　安装过程

JavaCard 中 applet 的安装过程分为如下几个步骤(见图 2)：

1)实现 JavaCard applet：可以使用通常的 Java 集成开发环境，如 VisualCafe 等，这样的开发环境都通常带有一个 JavaCard 模拟器，从而使得开发者可以将精力集中在功能实现和正确 API 的使用上。

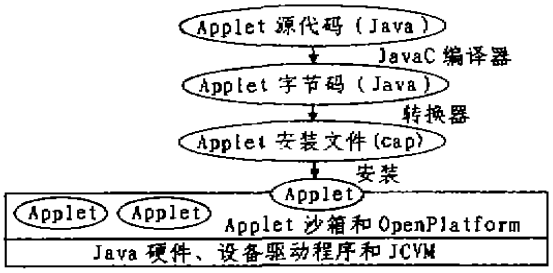

图 2　appelt 安装到 JavaCard 的过程

2)将编译好的 JavaCard applet 字节码转换为某种特殊格式：为了适应 JavaCard 有限的硬件资源，在正确编译 JavaCard applet 源代码后，必须运行一个转换器以便将生成的字节码转换为 JavaCard 的 Cardlet-Package 文件，即 CAP 格式。在转换过程中，用于在运行时进行动态类装载的大量代码数据将被删除，因为 JavaCard 平台不需要这种特性。

3)一次性装载和链接所有的类：由于禁止动态类装载，JavaCard 引入了一个被称为"闭合的包"的新概念，此时一个包或程序中的所有类将被同时下载到卡中，并进行适当的链接，显然，这是一个简化的装载、捆绑和初始化模型。

7.2　运行过程

JavaCard 的操作方式象一个典型的智能卡：当智能卡终端发送一个命令后，JavaCard 将处理该命令并返回相应的结果，为了维护和传统智能卡上应用程序的兼容性，一个 JavaCard 一次只能处理一个命令。具体来说，当 JavaCard 插入某个智能卡终端后，该终端将启动 JavaCard 硬件，从而使 JCVM 进入无限循环等待状态，当某个命令到达后，JCVM 对该命令解码并传送到相应的 applet，这是通过调用该 applet 的 process 方法完成的，以后的操作就取决于该 applet 所要完成的功能了。一旦该方法执行完毕，JavaCard 运行环境将返回一个状态字，并在后面附上该 applet 发送到 I/O 接口的数据。在整个 applet 执行过程中，标准 Java 的许多关键概念，如类继承、异常处理、类型安全的对象分配等都被遵循，但需要注意的是，为了保证事务处理的安全性，JavaCard applet 中内部数据结构的处理是一种原子方式，此外，对于 RAM 和 EEPROM 的访问，JavaCard 使用了不同的 API。

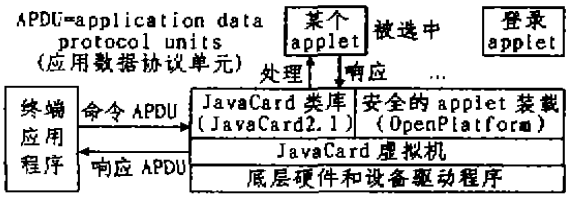

图 3　JavaCard 中 applet 的运行过程

结束语　在本文，我们详细探讨了有关 JavaCard 的各方面问题，相信随着 Java 技术的不断成熟和进步，它必将在嵌入式系统中发挥越来越大的作用。

参 考 文 献

1　Available at：http.//www.javaworld.com/javaworld/jw-05-1999/f-jw-05-sigs-embedded.html

2　Available at：http.//www.visa.com/nt/suppliers/open/main.html

3　Oestreicher M、Ksheeradbhi K. Object Lifetimes in JavaCard Proc Usenix Workshop Smart Card Technology，Usenix Assoc，Berkeley，Calif.，1999.129～137

4　Available at：http://www.javacardforum.org