第33卷 第17期 计 算 机 工 程 2007年9月
Vol.33 No.17 Computer Engineering September 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)17—0273—02 文献标识码：A 中图分类号：TP311.54

# 面向MHP的嵌入式Java虚拟机的移植与优化

石学锋[1]，陈 智[1]，李政道[2]
(1. 中国科学技术大学软件学院，合肥 230052；2. 扬智电子(上海)有限公司系统应用处，上海 200233)

**摘 要：**为了增强MHP机顶盒的网络交互能力，必须构建Java运行环境。该文介绍了Java技术、MHP机顶盒软件结构模型、嵌入式Java虚拟机(JVM)概念以及开源Java虚拟机——Kaffe的软件层次结构，阐述了Kaffe在ALI公司数字电视机顶盒开发平台上移植的实现过程，提出了在嵌入式环境下，Java虚拟机执行引擎的性能优化策略。实际运行结果证明了JVM的移植性和性能优化策略的可行性。
**关键词：**数字电视；机顶盒；Java虚拟机；多媒体家庭平台

# Transplantation and Optimization of Embedded Java Virtual Machine for MHP

SHI Xue-feng[1], CHEN Zhi[1], LI Zheng-dao[2]
(1. School of Software, University of Science and Technology of China, Hefei 230052;
2. Department of System Application, ALi(Shanghai) Ltd., Shanghai 200233)

**【Abstract】**This paper enhances the ability of interaction of MHP set-top-box by constructing Java runtime environment as a necessary technical approach. It introduces Java technology, software structural model of MHP set-top-box, basic concept of embedded Java virtual machine(JVM) and software layered structure of Kaffe(an open-source Java virtual machine). The process of Kaffe's transplantation and implementation on the platform of Ali's set-top-box is explained, the policy of performance optimization for Java virtual machine under embedded environment is presented. The final result proves that the success of transplantation and feasibility of policy about performance optimization for JVM's execution engine.
**【Key words】**digital TV; set-top-box; Java virtual machine; multimedia home platform

随着数字电视市场的进一步发展，交互式电视越来越受到重视。交互式电视为用户提供丰富的互动功能，包括电子节目指南、视频点播、股票信息、互动游戏等。实现交互式数字电视的关键问题是构建交互式数字电视平台。在DVB项目开发中，多媒体家庭平台(multimedia home platform, MHP)是第1个交互式电视的开放式标准[1]。它基于Java环境、定义交互式数码应用软件和这些软件运行的终端机之间的一个一般性界面。MHP应用的中间件技术中包含一个必不可少的关键技术组件——Java运行环境，即Java虚拟机。笔者以数字电视机顶盒技术为参考，基于MHP机顶盒软件结构模型，提出了一种将嵌入式Java虚拟机移植到数字电视机顶盒平台上的方案，阐述了实现过程，在成功移植的基础上，提出了性能优化的策略，运行结果证明了优化策略的可行性。

## 1 MHP机顶盒软件结构模型

MHP机顶盒软件技术结构如图1所示。

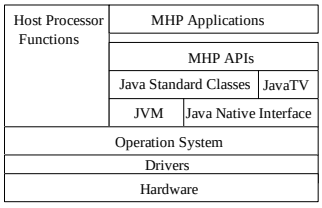


**图1 MHP机顶盒软件结构模型**

MHP应用程序是基于Java技术实现的，Java虚拟机担任着执行MHP应用程序的重要角色。

## 2 Java技术

Java技术包含Java编程语言、Java类文件格式、Java API、Java虚拟机4个既相互区别又相互联系的技术组件。从编写一个Java程序到执行，可以同时体验到这4种技术，其中，Java虚拟机和Java API构成了一个Java平台。

### 2.1 Java简介

通常一个可执行的二进制文件总是与特定的平台相关的，只能运行在一个指定类型的平台之上，而不能同时运行在其他平台上。相反，一个Java源程序被编译成类文件之后，能够运行在任何一个部署有Java虚拟机的平台之上，这也体现了Java平台无关性的优点，这样由Java编制的MHP程序可以在任何一个拥有Java虚拟机的机顶盒平台上正常运行。

### 2.2 Java API

Java API是一套能够提供Java应用程访问主机系统资源的标准接口的运行库，程序员编写Java程序时总是假定Java API的类文件在任何一个虚拟机上是可用的，而这些API的类文件本身总是与特定的主机平台相关的，因为API里所实现的功能总是依赖主机系统的本地方法来实现，所以Sun公司针对不同的行业定义子集，如Java TV™ API。

### 2.3 Java虚拟机

Java已被推广到各种平台，针对不同应用平台SUN公

**作者简介：**石学锋(1978－)，男，硕士研究生，主研方向：嵌入式系统；陈 智，硕士研究生，李政道，硕士、工程师
**收稿日期：**2006-11-02 **E-mail：**shixf@mail.ustc.edu.cn

司开发了多个版本的 Java 运行环境，如 PDA、STB 这样的硬件运算能力不高，且存储有限的各式消费性电子产品，并提出了 J2ME(Java 2 Micro Edition)版本。

Java 虚拟机的主要任务是：装载 Java class 文件并且执行其中的字节码，字节码的具体执行是由虚拟机的执行引擎来实现的，不同的 Java 虚拟机，其执行引擎的实现也存在很大的差异，在由软件实现的虚拟机中，最简单的执行引擎就是一次性解释字节码。这种类型的执行引擎也就是第一代 Java 虚拟机，执行效率不高。第 2 种执行引擎速度更快，但是也更消耗内存，称为“即时编译器”(just-in-time compiler)。其主要的原理是，第一次执行的字节码会被编译成本地机器代码，编译出的本地机器代码被缓存起来，当该方法再被调用时可以重用，这可以提高虚拟机的工作效率。第 3 种 Java 虚拟机执行引擎是自适应优化器。在这种引擎里，虚拟机一开始就解释字节码，监视运行中程序的活动，并且记录下使用最频繁的代码段。程序运行时，虚拟机只把那些活动频繁的代码编译成本地代码，其他的代码由于使用并不频繁，继续保留为字节码，由虚拟机继续解释它们。最后一种虚拟机是由硬件芯片构成，它们用本地方法执行 Java 字节码，这种执行引擎实际上是内嵌在芯片里的。

本方案选用 Ali 公司 S3601 数字电视机顶盒作为目标平台，该平台的 CPU 为 RISC 架构 CPU，没有协处理器，频率为 180MHz，内存为 32MB。选择开源的 Java 虚拟机软件——Kaffe(版本号：1.1.6)作为目标移植软件。

## 3 Kaffe 的软件体系结构

在移植前，首先分析一下 Kaffe 的软件体系结构，掌握其对本地系统依赖的接口情况，以便正确地实施移植工作，Kaffe 从层次结构上可以分为 3 个不同的层次：Java API 层，虚拟机核心层和本地方法层。Kaffe 的软件层次结构见图 2。

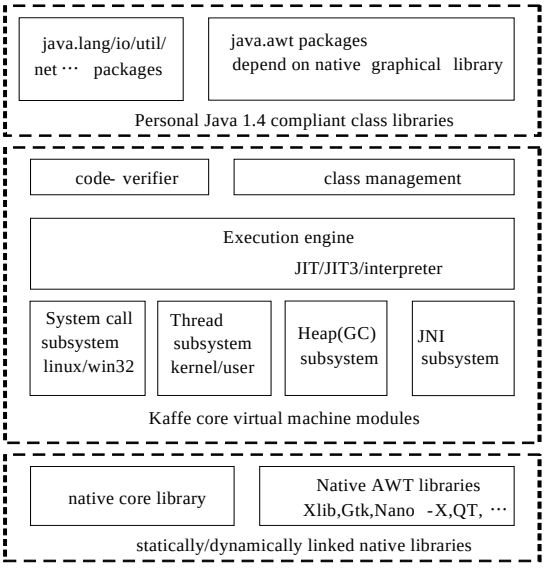


**图 2 Kaffe 软件层次结构**

## 4 移植与实现过程

### 4.1 实现方案

笔者采取了“自底向上”的移植策略，即在目标平台的基础上，根据 Kaffe 对本地接口的依赖情况，逐步为 Kaffe 建立正常运行所要求的本地方法调用服务，因为 Java 应用程序的执行最终是由本地方法予以实现的，所以本地方法的正确实现是 Kaffe 能否正常运行的关键所在。

### 4.2 本地调用方法的移植与实现

在移植的具体过程中，根据图 2 所示，将本地方法库的移植分为 2 个部分：

(1)本地图形库的移植，由于目标平台没有现成的图形库可供调用，考虑到机顶盒的资源利用情况，因此选择 Nano-X 作为 Kaffe 的本地图形库。

(2)除图形库外的本地核心库的移植与实现，例如 Kaffe 所要求诸如线程处理、信号处理、内存管理等相关的本地系统调用。

本地方法的移植实现过程如下：

(1)本地图形库 Nano-X 的移植。Nano-X 是上层 Java AWT 库的本地实现。Java 的图形绘制最终是由 Nano-X 来完成的，因此，它是 AWT 库能否正常运行的关键。在移植图形库的过程中，主要完成 3 个方面的工作：

1)图形输入设备的加载，例如，这里主要完成的是将通用的键盘输入接口改写成机顶盒的遥控器的输入接口。由于目标平台没有鼠标这种类型的输入设备，因此需对与鼠标相关的接口进行裁减。

2)图形输出接口的修改。

3)分析 Kaffe AWT 库的 JNI 功能调用情况，对 Nano-X 的功能进行裁减，保留 Kaffe 所需的功能。另外完成图形库对 OS API 调用接口的修改和完善工作，通过相关调试与测试，最终实现 Nano-X 在目标平台上正常运行。

(2)本地核心库的移植。根据目标平台的 API 库的实际情况，解决 Kaffe 对目标操作系统的依赖，对本地不满足的系统调用重新实现与完善。由于目标平台所运行的 OS 仅仅是一个简单的任务调度系统，不提供真正意义上的多线程、信号等功能的支持，API 的支持有限，因此选择 Kaffe 的中的 jthread 线程模型，即用户态线程模型，作为 Kaffe 的线程模型，其对 OS 的依赖较另外一种 linux 环境下的 pthread 线程模型更少，利于快速移植。Kaffe 所要求的信号功能的支持主要是服务其自身的线程调度与同步。为了解决信号问题，在移植过程中，采用在目标平台的操作系统上模拟各种信号的方式，间接地实现信号功能的支持。

### 4.3 平台差异的修改实现

为了提高 Java 虚拟机的执行性能，选择 JIT 类型的执行引擎，这种类型的重要特点就是在执行过程中，将 Java 方法的字节码翻译成本地指令，目标平台没有协处理器，不提供浮点相关的操作与运算，而 Kaffe 在翻译过程中会产生浮点相关的指令，自然翻译后的部分指令无法被目标平台的 CPU 执行。笔者通过修改 JIT 指令翻译处的部分源程序，采用软浮点方式来解决此类问题。

### 4.4 Java API 的移植与实现

目标平台上的 Java API 是为 MHP 应用程序和 MHP APIs 服务的。在移植的过程中，主要基于 MHP 的应用需求进行裁减，同时根据系统本地图形库的类型，选择相应的 Java AWT 库，来实现 Java API 库的移植工作。实现过程可以概括为 2 个方面：

(1)基于 MHP 的应用需求，对 Kaffe 的 Java API 进行裁减，减小类库所占的存储空间。

(2)Java API 本地方法的的修改与完善，即 Java 技术规范中的 JNI(Java native interface)方法，这些本地方法与目标平台息息相关，移植过程就是要正确修改这些本地方法的接口依赖，让 kaffe 的 JNI 方法能够正常运行，从而保证 Java API 能够正常工作。

（下转封三）