

一种异构多核系统的编译方法及实现

刘丹丹¹, 杨灿美¹, 倪素萍², 杜学亮²

(1 中国科学技术大学 电子科学与技术系 微纳电子系统集成研究中心, 安徽 合肥 230027;

2 中国科学院 自动化研究所 国家专用集成电路设计工程技术研究中心, 北京 100190)

摘 要: 面向专用领域计算加速的异构多核处理器近年来得到长足发展, 异构多核处理器中集成了多个不同架构的处理器核。由于该类处理器的异构性, 其编程方法较传统的同构多核处理器有很大不同, 编程者需要就不同架构的处理器核分别编写程序代码并分别编译, 增加了软件开发难度。在分析异构多核处理器体系结构、程序执行模型的基础上, 提出了一种异构多核系统的编译方法, 并给出系统实现, 解决了分别编写程序代码和编译的困难, 支持异构多核代码的统一编程, 屏蔽底层硬件的异构性, 为上层用户开发提供方便。

关键词: 异构多核; 异构编程; 编译方法

中图分类号: TP302.1

文献标识码: A

文章编号: 1000-7180(2015)11-0001-05

A Compilation Method and Realization for Heterogeneous Multi-core Systems

LIU Dan-dan¹, YANG Can-mei¹, NI Su-ping², DU Xue-liang²

(1 Department of Electronic Science and Technology, Micro-/Nano-Electronic System
Integration Center, University of Science and Technology of China, Hefei 230000, China;

2 The National Engineering & Technology Research Center for
ASIC Design, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Dedicated areas for heterogeneous multi-core processors to accelerate computing has made rapid progress in recent years. Heterogeneous multi-core processor integrates a number of different processor cores. Due to the heterogeneity of the class processor, its programming are very different compared to the traditional homogeneous multicore processors. The programmer needs to write program code separately and compiled respectively for different architecture processor core, which increasing the difficulty of software development. On the basis of on the analysis of the heterogeneous multi-core processor architecture, program execution model, we propose a compilation method for compiling heterogeneous multicore systems, solving the difficulties of programming code and compiled separately, and supports heterogeneous multicore unified programming code. It shields the underlying hardware heterogeneity, providing convenience for the development of upper user.

Key words: heterogeneous multi-core; heterogeneous programming; compilation method

1 引言

随着半导体工艺的进步以及处理器体系的发展, 在单芯片上集成多个处理器核是当今片上系统设计的发展趋势^[1]。多核处理器分为同构多核处理

器和异构多核处理器。同构多核处理器中集成了多个架构相同的处理器核, 每个处理器核处理工作负载中相同类型的操作; 异构多核处理器中集成了多个架构不同的处理器核, 不同架构的处理器核处理工作负载不同类型的操作。目前最常见的异构多核

收稿日期: 2015-01-18; 修回日期: 2015-03-09

基金项目: 中国科学院战略性先导科技专项(XDA06010402)

处理器由通用处理器核和专用处理器核组成,通用处理器核采用成熟的标量处理器,提供通用的计算能力和控制操作,适应多方面的应用;专用处理器核则面向加速特定类型的工作负载进行定制设计,提供强大的计算性能,这样在特定应用领域,异构多核处理器比同构多核处理器能发挥出更高的性能和效能^[2].

通用处理器核和专用处理器核各自有不同的指令集^[3],所以运行在异构处理器核上的程序一般需要分开设计和编译.目前支持异构平台上编程的成熟架构有 NVIDIA 公司的 CUDA^[4-5]和 Khronos 组织的 OpenCL^[6-7],它们都需要对原始的代码进行分析,分离出异构处理器的相应程序后进行编译,得到最终的可执行文件. CUDA 架构只支持 CPU 和 NVIDIA 公司 GPU 的异构,其体系结构不同,有很多繁杂的 GPU 的控制代码. OpenCL 为异构系统提供了一种通用目的并行编程的统一编程环境,支持 CPU、GPU、DSP 等其他并行处理器,但因其要支持多种不同编译类型的异构系统,其体系复杂,因此编程困难.本文提出一种异构多核系统的编译方法和系统实现,包括系统设计原理、系统实现和异构多核可执行程序的构建,最后结合应用实例说明系统的使用.

2 异构多核体系结构

2.1 体系结构的抽象模型

如图 1 所示的异构多核处理器抽象模型,采用“通用处理器核+专用处理器核”的设计.通常设计在通用处理器核上运行操作系统,通用处理器核负责系统的资源管理、任务分配和调度;专用处理器核抽象为通用处理器核的一个子设备,负责特定应用领域的计算加速工作^[8].通用处理器核在整个结构占主要地位,可以访问全部内存空间,专用处理器核只能访问有限的内存空间,通用处理器核通过内部逻辑控制模块来设置专用处理器核的访问权限,并且控制专用处理器核的运行,通用处理器核与专用处理器核之间的交互是单边的,是一种主从关系^[9],通用处理器核为主核,专用处理器核为从核.

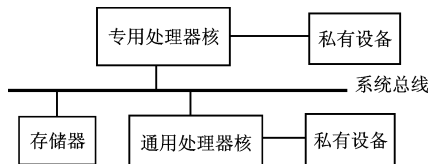


图 1 异构多核处理器抽象模型

2.2 程序执行模型

异构多核处理器的程序执行模型如图 2 所示,其中,主核程序段指在通用处理器核上运行的程序段,从核程序段指在专用处理器核运行的程序.

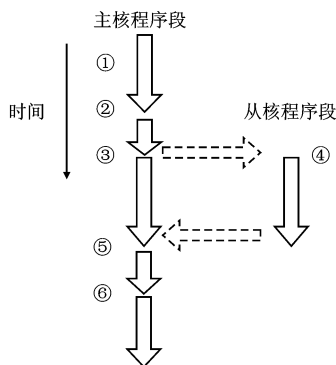


图 2 异构多核处理器的程序执行模型^[9]

程序执行体首先在通用处理器上加加载并执行,见阶段①;在该程序串行执行过程中,将程序中适用于专用处理器完成的计算功能卸载到专用处理器上完成,见阶段②;然后执行阶段③和④,主、从核并行执行;在专用处理器完成执行任务后,通用处理器取回计算结果(阶段⑤),继续执行(阶段⑥).

根据上述异构多核处理器的程序执行模型,将最初在通用处理器核上执行程序的一部分卸载到专用处理器核上执行,专用处理器核用作加速单元对程序中部分工作进行加速处理,从而提高整个程序的性能.卸载的程序功能将以函数的形式来描述,这种编程模型也称功能卸载编程模型^[9].

3 新的编译方法

在功能卸载编程模型中,编程者需面向通用处理器和专用处理器分别设计程序源代码模块,并调用不同的编译器分别进行编译,增加了软件开发难度.为解决面向通用处理器和专用处理器分别编写程序代码和编译的困难,本文提出了一种异构多核系统的编译方法,支持异构多核代码的统一编程.

通用处理器可以沿用标准的串行程序设计语言(如 C、C++ 等);根据功能卸载编程模型特点,对专用处理器的实现方式是在标准串行程序设计语言的基础上进行扩展,本文设计在专用处理器核运行的函数加前缀标记.这样,专用处理器上的程序模块在通用处理器看来就像一个特定实现的函数.具体的编译方法如下:(1)根据函数前缀标记,识别源程序中的主核程序与从核程序,分离主核程序与从核程序;(2)将标记函数的调用处归并为主核程序段,并修改标记函

数的调用为运行库(运行库负责主核与从核之间的交互,包括设备间数据通信、同步操作等)加载从核可执行实体,建立主核程序工程,调用主核编译器生成主核目标文件;(3)将标记函数的定义实现处归并为从核程序段,分析并提取标记函数中嵌套调用的子函数相关文件,建立从核程序工程,调用从核编译器和链接器生成从核可执行实体;(4)创建外壳主核目标文件,嵌入从核可执行实体到外壳主核目标文件中;(5)调用主核链接器链接主核目标文件和外壳主核目标文件,得到最终的可执行实体。

通过上述编译方法,自动分析、分离出主核程序和从核程序,然后调用相应的编译工具链编译链接,得到最后的可执行实体。该方法屏蔽异构系统中的底层硬件异构性,允许用户采用同构视角的编程方法,有效提升了异构系统的可用性,降低了异构系统的编程难度。下面给出此编译方法的系统实现,称作自动编译系统。

4 自动编译系统设计与实现

4.1 自动编译系统设计原理

图3是自动编译系统设计原理示意图。异构多核程序中包含了通用处理器(主核)上运行的程序和专用处理器(从核)上运行的程序,其中带有标记的程序为从核程序。自动编译系统据此分离出主核程序和从核程序,并分别编译,如图3所示的“分离和编译”。在执行时,通过运行库加载从核程序段并启动运行,如图3所示的“运行库”。自动编译系统屏蔽了处理器异构带来的差异性,编程者在面向通用处理器和专用处理器编程时,无需分别编写程序源代码并调用相应的编译器分别进行编译,仅需在专用处理器上运行的函数/程序加入简单的前缀标记,自动编译系统可以自动分离出主核运行程序和从核运行程序,并调用相应的编译器分别进行编译。

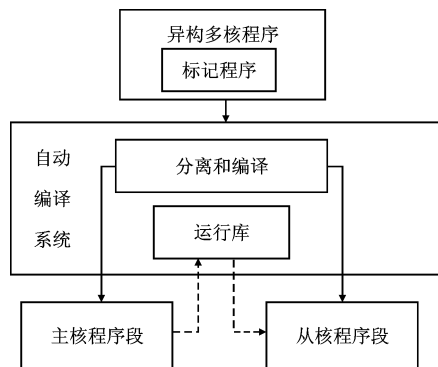


图3 自动编译系统的原理示意图

在无自动编译系统的情况下,程序员需要就主核程序和从核程序分开编辑源代码,然后手动调用相应的编译器编译,同时在设计主核程序时还需加入对从核程序调用的运行库操作,步骤繁杂,编程效率低下;自动编译系统解决了异构多核编程下分开编辑源代码的不便,可以在一个工程或文件中统一编辑主、从核程序源代码,系统根据前缀标记分离主、从核程序并分别编译,同时会自动加入运行库的调用。使用自动编译系统后的工作流程如同同构处理器的编程,为程序员提供了极大方便。图4为异构多核编程下使用自动编译系统前后的工作流程对比。

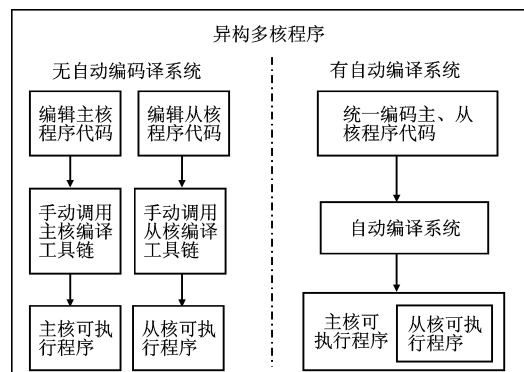


图4 使用自动编译系统前后的工作流程对比

4.2 自动编译系统实现

自动编译系统的实现包含三部分:分离处理模块、主核编译处理模块和从核编译处理模块,其流程图如图5所示。

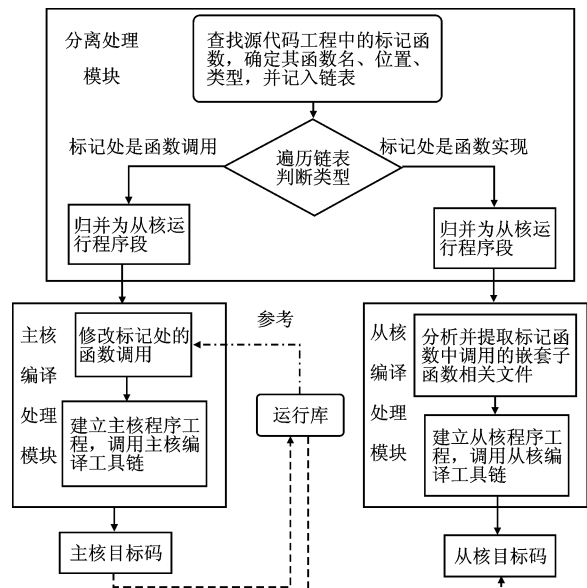


图5 自动编译系统实现的流程图

自动编译系统中利用函数前缀标记来区分主核程序与从核程序,分离处理器模块依此来分离主核程序和从核程序.建立一个全局链表用于记录标记函数的函数名、位置、函数类型等信息.标记函数的类型分为函数实现与函数调用,如果是函数调用,则将源程序中标记函数所在代码归并为主核上执行的主核程序;如果是函数实现,则源程序中标记函数所在代码归并为从核上执行的从核程序.在主核编译处理器模块中,将源程序中该标记函数的调用修改为运行库加载从核目标码,然后建立主核程序工程,调用的主核工具链进行编译,生成主核目标码;在从核编译处理器模块中,将源程序中该标记函数嵌套调用的子函数文件以及相应头文件提取出来,构成从核编译工程,调用异构从核的工具链进行编译,生成从核目标码.

应用程序运行时执行流程如图 5 中的虚线部分所示,主核目标码首先在主核上加载并顺序执行,运行到标记函数时,系统已将其替换成运行库中的函数调用,完成从核目标码在从核上的加载,并启动执行,得到调用结果.

4.3 可执行程序的构建

异构系统中将在不同处理器核上执行的目标代码组织成一个系统环境下的执行实体(可执行程序),图 6 展示了异构多核程序工程中的可执行程序的构建过程,其中的异构多核程序工程中包含了通用处理器核(主核)上运行的程序和专用处理器核(从核)上运行的程序,假设工程有 N 个带前缀标记的程序/函数,每一个标记的程序/函数将在从核上执行,以实现加速的功能.经自动编译系统中的分离模块处理后,得到一个主核程序工程和对应的 N 个从核程序工程(与 N 个带前缀标记的程序/函数相对应);主核编译处理阶段,替换 N 个标记函数的调用处为运行库的函数调用;从核编译处理阶段,提取标记函数中嵌套调用的子函数所在的文件以及所需的头文件和从核加速库文件,通过从核工具链中的编译器和连接器生成从核可执行程序.嵌入器将从核可执行程序作为主核程序的数据段部分嵌入到主核的可执行文件中,在执行时通过运行库实现从核可执行程序的加载和启动.

5 异构并行编程接口

异构并行编程接口需为程序员提供了合理的异构架构抽象,使程序员对异构计算资源加以合理利用,同时又需保证接口的易用性,避免程序员陷入复

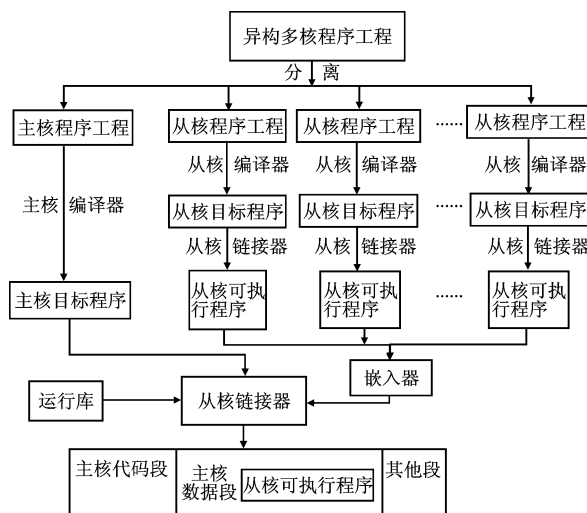


图 6 异构多核程序工程下的可执行程序的构建

杂的硬件细节中,其主要体现在异构任务划分、数据分布、通信、同步这个 4 个因素中^[10].自动编译系统中允许程序员通过函数前缀标记来划分主核上执行的任务与从核上执行的任务,不需要程序员指定异构设备间的数据分布,数据通信与设备间的同步由系统中运行库自动完成.这样的异构并行编程接口无疑减轻了程序员的编程负担,提高编程效率,表 1 所示为自动编译系统与 CUDA、OpenCL 的编程接口的比较.

表 1 CUDA、OpenCL、自动编译系统的编程接口

名称	任务划分	数据分布	通信、同步
CUDA	Kernel 函数	显式	显式
OpenCL	Kernel 函数	显式	显式
自动编译系统	函数前缀标记	隐式	隐式

6 应用实例

实验的主机操作系统平台为 Ubuntu12.10,自动编译系统使用的格式如下:

```
Auto_compiler_sys proj_dir prefix_lab M_compiler_dir S_compiler_dir
```

其中 Auto_compiler_sys 指自动编译系统应用程序名,proj_dir 指异构多核源程序的工程路径,prefix_lab 指运行在从核上函数的前缀标记,M_compiler_dir 指主核编译器路径,S_compiler_dir 指从核编译器路径.二维向量加法的 C 语言实例如下:

```
p0_param.len = len;
p0_param.x = malloc(len * sizeof(int));
p0_param.y = malloc(len * sizeof(int));
```

```

p0_param.z = malloc(len * sizeof(int));
printf(" vadd: \n\t");
for (i=0; i<len; i++) {
    p0_param.x[i] = x;
    p0_param.y[i] = y;
    printf(" (%d + %d) ", p0_param.x[i], p0_param.y
[i]);
}
AFI_vadd((void *)&p0_param);
printf(" vadd result: \t");
for (i=0; i<len; i++)
    printf(" %d ", p0_param.z[i]);

```

函数前缀标记 AFI_表示了函数 AFI_vadd() 在从核上执行以实现加速. 经过自动编译系统的处理后,函数 AFI_vadd() 实现处会在从核编译器上编译得到从核执行体,函数 AFI_vadd() 调用处会被替换成运行库加载启动从核执行体的调用.

7 结束语

本文根据异构多核处理器的特点,提出了一种异构多核系统的编译方法,设计自动编译系统. 该方法通过简单函数前缀标记的异构并行编程接口来区分主核程序与从核程序,实现了异构多核代码的自动分离和编译,屏蔽处理器异构带来的差异,允许用户采用同构视角的编程方法,有效提升了异构系统的可用性,提高了程序员开发效率,同时也便于应用程序从同构平台上移植到异构平台上.

参考文献:

- [1] Xiaojing Feng. Study and implementation of service oriented heterogeneous multi processor system-on-chip [D]. Wuhan, University of Science and Technology of China, 2013.
- [2] Tang Tao, Yang Xuejun. A survey of the programming methods for heterogeneous systems [J]. Computer Engineering & Science, 2012, 34(3): 29-31.
- [3] Chen Fangyuan, Zhang Dongsong, Wang Zhiying. Research of the heterogeneous multi-core processor architecture design [J]. Computer Engineering & Science, 2011, 33(12): 27-28.
- [4] NVIDIA CUDAC Programming Guide, Version 2. 2. 0 [EB/OL]. [2014-01-17]. http://developer.download.nvidia.com/GPU_Programming_Guide/GPU_Programming_Guide_Chinese.pdf.
- [5] Dong Luo, Ge Wancheng, Chen Kangli. Study on application of parallel computation on CUDA [J]. Information Technology, 2010, 17(4): 11-12.
- [6] Khronos Group. Open computing language [EB/OL]. [2014-07-10]. <http://www.khronos.org/>.
- [7] Stone J E, Gohara D, Shi G. OpenCL: a parallel programming standard for heterogeneous computing systems [J]. Computing in science & engineering, 2010, 12(3): 66.
- [8] Zhai wenzheng. Study on the model-based development of heterogeneous multi-core embedded system [D]. Shanghai: Shanghai University, 2013.
- [9] Li Chunjiang, Yang Xuejun. The programming model and compilation framework for master_slave one-sided heterogeneous multi-core processors [J]. Computer Engineering & Science, 2009, 31(8): 67-68.
- [10] Liu Ying, Lu Fang, Wang Lei, et al. Research on heterogeneous parallel programming model [J]. Journal of Software, 2014, 25(7): 1462-1467.

作者简介:

刘丹丹 男, (1990-), 硕士研究生. 研究方向为嵌入式软件开发. E-mail: liudandan14118@126.com

杨灿美 男, (1965-), 博士, 研究员. 研究方向为通信数字基带系统设计、低功耗与高密度 SoC 架构设计.

倪素萍 女, (1977-), 硕士, 工程师. 研究方向为系统架构设计.

杜学亮 男, (1982-), 博士, 高级工程师. 研究方向为处理器架构及 SoC 设计.