# 一种基于 CPU-GPU 异构计算的混合编程模型

王　伟[1]，郭绍忠[1]，王　磊[1]，冯　颖[2]

（1. 信息工程大学 信息工程学院，河南 郑州 450002；2. 总后勤部档案馆，北京 100842）

**摘要**：在分析基于 CPU-GPU 异构计算模式程序开发面临的主要挑战和当前解决途径的基础上，设计了一种适用于 GPU 集群环境的、综合利用 MPI、OpenMP、CUDA 以及 OpenGL API 开发技术的混合编程模型，重点分析了混合编程模型的两种编程方式的实现原理，搭建了相应的实验环境并进行了对比测试。实验结果表明，使用混合编程模型设计的程序具有良好的性能表现和扩展潜力。

**关键词**：GPU；异构计算；混合编程

**中图分类号**：TP311.52　　**文献标识码**：A　　**文章编号**：1671－0673(2010)06－0674－05

## Hybrid Programming Model Based on CPU-GPU Heterogeneous Computing

WANG Wei[1], GUO Shao-zhong[1], WANG Lei[1], FENG Ying[2]

(1. Institute of Information Engineering, Information Engineering University, Zhengzhou 450002, China;
2. Headquarters archives of General logistics, Beijing 100842, China)

**Abstract**: The paper analyzes the main difficulties of programming based on CPU-GPU heterogeneous computing, summarizes the main approaches available, and designs a hybrid programming model for GPU cluster environment which comprehensively uses MPI, OpenMP, CUDA and OpenGL API. It focuses on analyzing two methods that implement the programming model, then builds experimentation environment and makes a comparison experiment. The result shows that programs built with hybrid programming model have good performance and scalability.

**Key words**: GPU; heterogeneous computing; hybrid programming

近年来，基于 CPU-GPU 的混合异构计算系统逐渐成为国内外高性能计算领域的热点研究方向。在实际应用中，很多基于 CPU-GPU 的混合异构计算机系统纷纷涌现，并且表现出良好的性能。但是，由于历史和现实原因的制约，异构计算仍面临着诸多问题，其中最突出的是程序开发困难，尤其是扩展到集群规模级别时，问题更为突出。本文在分析 CPU-GPU 异构计算模式程序开发现状的基础上设计了一种综合利用 MPI、OpenMP、CUDA 以及图形 API 编程方法的混合编程模型，并进行了系统实现和实验验证。

## 1　CPU-GPU 异构计算模式研究现状

### 1.1　CPU-GPU 异构计算模式编程面临的挑战

基于 CPU-GPU 的混合异构计算系统是指在传统计算机系统中加入 GPU 作为加速部件并配合 CPU 共同完成计算任务的新型系统。相比于传统的单纯以 CPU 为计算部件的系统，异构计算系统在浮点运算

收稿日期：2010－04－27；修回日期：2010－06－23

基金项目：国家 863 计划资助项目(2009AA012201)；上海市科委重大科技攻关项目(08dz501600)

作者简介：王　伟(1983－)，男，硕士生，主要研究方向为分布式计算、GPU 通用计算；

郭绍忠(1964－)，女，副教授，主要研究方向为分布式系统、海量信息处理。

能力、功耗、可扩展性等方面都具有明显的优势[1]。近年来,针对 CPU-GPU 混合异构计算的研究发展迅速,但是由于历史和现实因素的制约,CPU-GPU 异构计算仍面临诸多问题,其中最突出的是程序开发困难。究其原因,一是 GPU 最初设计目的是专业图形处理而非通用计算,这导致了 GPU 本身的体系架构对通用计算存在许多硬件制约,例如缺少数据校验机制、双精度性能偏低等,这使得程序开发人员在使用 GPU 进行通用计算时不得不专门考虑这些问题;二是 GPU 软件开发的编程模型及编程方式还不成熟,尽管 NVIDIA 公司推出的 CUDA 技术已经大大降低了 GPU 通用计算开发的难度,但是要程序员改变 CPU 模式下的 X86 编程习惯并非易事,而如何处理以往应用中的遗留代码也是很大的挑战;三是异构计算标准 OpenCL(Open Computing Language,开放计算语言)推出时间比较短,其应用还不广泛。

### 1.2 目前可用的程序开发方法及适用场合

目前,CPU-GPU 混合异构计算系统程序开发工具可以大致分为以下 4 类[2]:①基于底层图形 API 的开发方法,即使用 CG、GLSL 等图形绘制语言将算法映射到图形处理过程,适用于积累了大量基于图形 API 方法编写的遗留程序的开发者;②基于低层次抽象的轻量级 GPU 编程工具,即使用 CUDA、OpenCL 等编程工具编写 GPU 内核程序,适用于那些需要编写少量专业领域算法应用程序的开发者;③基于高层次抽象的函数库或模板库,即直接调用已经封装好的诸如 CUBLAS、CUFFT、CUDPP 之类的函数库进行开发,适用于编写大部分运行时间都用于执行标准函数的应用程序的开发者;④基于高层次抽象的使用编译器的方法,即通过使用指示语句、算法模板以及复杂的代码分析技术,由编译器或语言运行时系统自动生成 GPU 内核程序,如 Portland Group 编译器、HMPP[3] 等,适用于开发大量使用专业领域算法应用程序的开发者。

## 2 混合编程模型的设计

基于 CPU-GPU 的混合异构计算系统按规模可以划分为工作站级别、集群级别两类。在高性能计算领域使用 GPU 作为加速部件时一般采用集群方式,即将 CPU-GPU 异构系统作为节点构造大规模集群,这时会涉及到多个异构节点的管理。在 GPU 集群环境下进行程序开发时,除了上面列出的 4 种 GPU 编程方式,还需要其它一些必要的辅助技术。本文针对 GPU 集群环境提出一种综合使用 MPI、OpenMP、CUDA 及 OpenGL API 开发技术的混合编程模型。为叙述简洁清晰,下文简称混合编程模型。

### 2.1 主从模型设计思想

混合编程模型的设计理念基于异构模式下的主从模型(Master/Worker)设计思想[4-6]。主从模型设计思想大致描述如下:给定一个问题或算法,在求解时,将问题或算法划分为多个子任务,这些子任务遵循一定的流程执行。子任务的结构包括两部分,即数据和在数据上所执行的运算。具体到 CPU-GPU 协作计算,可以将不同的子任务映射到 GPU 与 CPU 上分别完成,即通过任务划分将不同性质的子模块调度到不同的适合的硬件上执行。由主控节点的 CPU 负责任务调度并处理指令复杂、分支循环多的子任务,计算节点的 GPU 负责处理指令简单、高度并行的计算子任务。

混合编程模型包括一个主控节点和多个计算节点。在只有一个计算节点时的计算执行流程如图 1 所示,描述如下:数据初始化任务交给主控节点的 CPU 使用 OpenMP 并行处理。数据初始化完毕,主控节点将计算任务划分为适合 CPU 计算的子任务和适合 GPU 计算的子任务,然后将 CPU 计算子任务留下处理使用 MPI 将适合 GPU 计算的子任务分发给计算节点,并由计算节点的 CPU 将子任务程序和数据映射到 GPU 进行计算。计算节点的 GPU 采用 CUDA 或 OpenGL API 方式执行计算任务。计算结束后,计算节点的执行结果通过 MPI 传回主控节点,主控节点的 CPU 负责结果收集和生成结果文件。

在整个计算过程中,程序经过 3 个不同粒度的并行加速:节点级并行、线程级并行和 GPU 硬件并行。三级并行分别用到的并行计算技术为 MPI,OpenMP,CUDA 或 OpenGL API。混合编程模型支持 2 种编程方式:MPI + OpenMP + OpenGL API(以下简称 MOO)和 MPI + OpenMP + CUDA(以下简称 MOC)。MPI 是一种消息传递型并行通信的程序设计标准,具体定义了一组消息传递函数。OpenMP 是支持共享存储并行编程的工业标准,提供了一种与语言平台无关的编写消息传递程序的标准。

### 2.2 MPI + OpenMP + OpenGL API 编程模型

混合编程模型针对的 GPU 集群一般具有共享存储和分布存储两级存储结构。为了充分利用这种结

构特征,集群节点内通信使用共享内存的通信方式,集群节点间使用消息传递的通信方式,通过两级并行提高并行计算性能。具体技术上,混合编程模型采用 MPI 和 OpenMP。在 GPU 硬件并行级别,使用图形 API 实现是一种选择。这种实现方式中,开发者需要将程序用图形绘制语言重新表达,将程序步骤设法映射到图形绘制流程。常用的图形绘制语言有 OpenGL 的 GLSL、NVIDIA 的 CG 等。混合编程模型使用 OpenGL 的 GLSL。

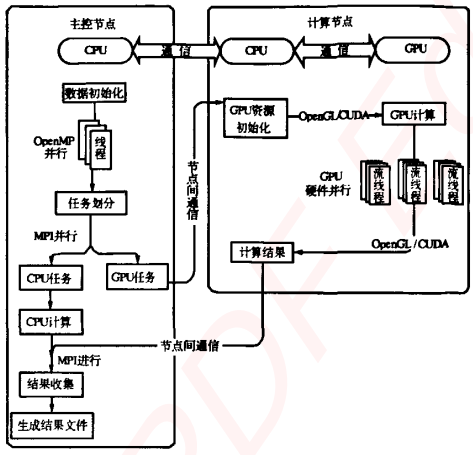


图 1 混合编程模型计算执行流程图

MOO 编程方式并行模型如图 2 所示。混合编程模型可以充分利用 3 种编程方式的优点:MPI 解决各个多处理器计算机间的粗粒度通信,OpenMP 提供的轻量级线程解决每个多处理器计算机内部各处理器间的交互,OpenGL API 编程将计算任务进一步分发给 GPU 各个流处理器执行。该编程方式是一种层次模型,MPI 并行位于顶层,OpenMP 位于中层,OpenGL API 位于底层。

MOO 编程方式的缺点在于开发者必须熟悉 GPU 底层图形 API,编程难度大。它的优点是:首先兼容性较好,各生产厂商的 GPU 一般都能适用;其次,可处理遗留代码,在 CUDA 技术出现之前,利用图形 API 编程一直是 GPU 通用计算的主流方法,曾遗留大量的算法代码,将这些代码的关键部分使用 OpenMP 和 MPI 并行化,就可以直接使用 MOO 编程方式在 GPU 集群上实现移植,这体现了混合编程模型具有良好的向后兼容性。

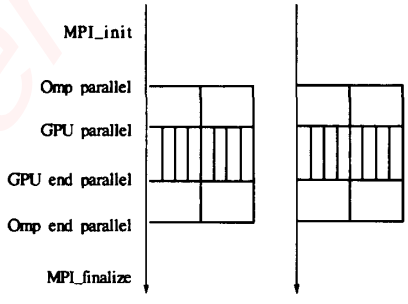


图 2 MOO 编程方式并行模型图

### 2.3 MPI + OpenMP + CUDA 编程模型

在实现 GPU 硬件级别并行时,使用 CUDA 实现是另一种选择。CUDA 是 NVIDIA 公司为自家生产的 GPU 产品开发的一种软件架构。作为基于 GPU 通用计算的开发平台,CUDA 非常适合于多个线程高度并行的数据计算。开发者可以直接将 GPU 作为数据并行计算设备进行开发,而无需将其映射到图形 API。CUDA 编程模型如图 3 所示[7]:CUDA 包含一个 C 扩展语言和一个相应的编译器,用户编写的 C 语言代码经过编译器编译后转换成 GPU 代码和 CPU 代码。适合 GPU 执行的代码交给 CUDA 驱动,控制 GPU 执行相关数据操作,而适合 CPU 执行的代码则交给标准 C 编译器生成机器码对 CPU 进行控制。

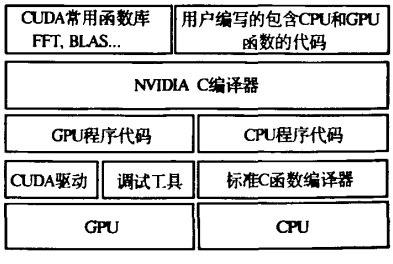


图 3 CUDA 编程模型图

MOC 编程方式的关键代码实现过程可描述为:首先使用 MPI 并行化程序,将问题划分为多个子任务,每个进程负责一个子任务,它们之间以消息传递方式来进行同步和通信。程序通过调用 MPI_Init( )和 MPI_Finalize( )来开始和结束并行程序,消息的发送和接收由 MPI_Send( )和 MPI_Recv( )分别完成。在每个 MPI 进程中,OpenMP 并行化可用于 2 种情形,一是可以进行常规并行化的程序(如数据的初始化操作)用 OpenMP 标准中的并行化指示语句完成并行化;二是在存在多个 GPU 设备时,可以用多线程控制,每个 OpenMP 线程控制一个 GPU 设备,在每个线程内编写 CUDA 内核程序,由 GPU 实现进一步硬件级并行。伪代码如下:

```
#include "omp.h"
#include "mpi.h"
int main(int argc,char * * argv)
```

```
{
    MPI_Init(&argc,&argv);/* MPI 初始化 */
    MPI_Comm_size ( MPL COMM_WORLD,num_processor);
    MPI_Comm_rank ( MPL COMM_WORLD,&myid);
    /* 若存在多个 GPU 设备数,则一般设置 OpenMP 线程数为设备数 */
    omp_set_num_threads(n);
    MPI_XXX ( ) /* MPI 操作 */
    #pragma omp parallel
    DataInit( );
    #pragma omp parallel
    Cuda_function( ); /* CUDA 运算 */
    MPI_XXX ( ) /* MPI 操作 */
    MPI_Finalize( ); /* MPI 进程结束 */
}
```

CUDA 技术作为一种新型 GPU 编程技术,在现实中已经出现了大量的应用案例,在未来拥有广阔的应用前景。混合编程模型的 MOC 编程方式对 CUDA 技术进行支持,体现了其具有良好的向前兼容性。

## 3 实验测试

### 3.1 实验环境的搭建

下面设置一个实验环境对混合编程模型进行性能测试。

为全面测试,本实验分别在 Windows 和 Linux 系统环境下搭建了 2 个环境。为便于比较,每个实验环境分别设置 2 个运行串行程序的比较测试环境。鉴于条件限制,每个实验环境只包括 1 个主控节点和 1 个计算节点,主控节点负责发送任务和接收结果,计算节点通过高速网络与主控节点连接,GPU 作为协处理器配置于计算节点,并通过 PCI-E ×16 接口与节点主板相连。下文称实验环境为算法加速器。算法加速器软件结构如图 4 所示。

测试过程中,在数据规模依次递增的情况下进行测试。不同测试环境下所取的计算时间是运行主体算法的时间(不包含读写文件时间),即主控节点开始向计算节点发送数据至主控节点接收到计算节点的计算结果之间的时间。实验数据采用 30 次数据中去掉最大值和最小值后剩余实验数据的平均值。

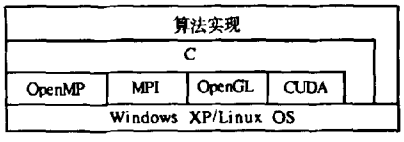


图 4 算法加速器软件结构图

### 3.2 MPI + OpenMP + OpenGL API 实验分析

MOO 编程方式在 Windows 环境下测试。主要配置为:主控节点(Intel Mobile DualCore Intel Core Duo T2300 CPU,1G 内存),计算节点(Intel Pentium E2160 CPU,1G 内存,NVIDIA GeForce 8800 GS GPU,768M 显存);比较测试环境:① PC 机(Intel Pentium 4 CPU,1G 内存,Windows XP OS);②服务器(Intel Xeon MP CPU,1G 内存)。

实验用 MOO 编程模型实现了 BLAS 库的 SSYMV 子函数(α * 向量 + β * 矩阵 * 向量)。不同测试环境下 SSYMV 算法时间分布如图 5 所示。

分析测试结果:采用 CPU 串行计算 SSYMV 算法时,随着矩阵数据规模的扩大,CPU 计算时间不断增加,当运算数据量达到一定规模时,整体运算时间增长到让人难以接受的程度;在服务器上运行同样算法,性能有所提升,但提升空间非常有限;使用 MOO 编程模型实现的并行算法执行时,当数据规模较小时由于通信和 GPU 初始化时间较长,其性能远不如其它 2 种测试环境,但随着数据规模逐渐增大,混合编程的性能优势逐渐体现,在数据规模达到 512 以上时,其执行效率远高于比较测试环境。

### 3.3 MPI + OpenMP + CUDA 实验分析

MOC 编程模型在 Linux 环境下测试。主要配置为:主控节点(Intel Pentium 4.3GHz CPU,2G 内存),计

算节点(Intel Pentium 4.3GHz CPU,2G 内存,NVIDIA GeForce GTX260 GPU,896M 显存)。比较测试环境:①PC 机(Intel Pentium 4.3GHz CPU,2G 内存);②服务器(Intel Core Quad CPU Q8200 2.33GHz,4G 内存)。

实验使用 MOC 编程模型实现了正方形矩阵乘算法,其中,OpenMP 只用于矩阵数据初始化。不同测试环境下矩阵相乘算法时间分布图如图 6 所示。

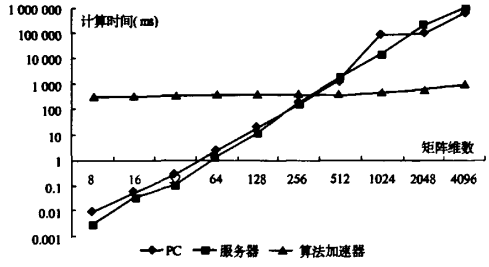


图 5 不同测试环境下 SSYMV 算法时间分布图

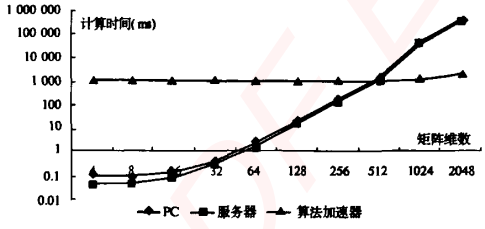


图 6 不同测试环境下矩阵乘算法时间分布图

分析测试结果:采用 CPU 串行计算矩阵乘法时,随着矩阵数据规模不断扩大,CPU 计算时间几乎线性增加;在服务器上运行同样算法,性能提升空间非常有限;使用 MOC 编程模型实现的并行算法执行时,当数据规模较小时其性能远不如比较测试环境,但随着数据规模逐渐增大,混合编程的性能优势逐渐体现,在数据规模达到 1 024 以上时,其执行效率远高于比较测试环境。

综合分析测试结果可以得出结论:混合编程模型在 MOO 和 MOC 2 种实现方式下性能都表现良好,从而验证了混合编程模型设计的合理性。在程序执行时,只有当计算和通信及初始化的比率较高时,通信和初始化的时间被隐藏,混合编程模型执行的性能优势才可以体现,这说明,基于 CPU-GPU 的混合异构计算模式适合运行计算强度大的算法与程序。

## 4 结束语

由于 GPU 在功耗、可扩展性方面具有的优势,使用 GPU 部件作为加速器的高性能计算机系统近年来发展迅速,但是程序开发困难是这种 CPU-GPU 异构计算模式目前面临的主要问题。本文针对集群环境下的 GPU 集群系统设计了一种基于主从模型思想的混合编程模型,从实现原理方面进行了理论分析,并且进行了相应的实验验证。混合编程模型综合运用了 MPI、OpenMP 以及 2 种 GPU 编程方法,具有较好的兼容性,对于集群环境下 GPU 异构系统程序开发具有重要意义。

**参考文献:**

[1] NVIDIA Corporation. CUDA Programming Guide 2.3[EB/OL]. [2009-07-10]. http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf.

[2] KINDRATENKO V V, ENOS J J, SHI G, et al. GPU Clusters for High-Performance Computing[EB/OL]. [2009-09-01]. http://www.ncsa.illinois.edu/kindr/papers/ppac09_paper.pdf.

[3] DOLBEAU R, STÉPHANEBIHAN, BODIN A F. HMPP: A Hybrid Multi-core Parallel Programming Environment[EB/OL]. [2009-10-20]. http://www.caps-entreprise.com/upload/ckfinder/userfiles/files/caps-hmpp-gpgpu-Boston-Workshop-Oct-2007.pdf.

[4] 陈华平,黄刘生,安虹,等.并行分布计算中的任务调度及其分类[J].计算机科学,2001,28(1):45-47.

[5] 冯高锋.GPU-CPU 集群上的动态规划算法[J].计算机应用,2007,27(12):281-282.

[6] 陈劲.SOC 软硬件协同设计自适应粒度算法研究[D].北京:清华大学,2004.

[7] NVIDIA Corporation. NVIDIA Tesla GPU Computing Technical Brief 1.0[EB/OL]. [2007-12-03]. http://www.nvidia.com/docs/IO/43395/Compute_Tech_Brief_v1-0-0_final_Dec07.pdf.

# 嵌入式资源免费下载

## 总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现

# VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植

15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法

# Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通

12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计

# Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计

2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现

# PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植

13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法

# ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计

26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植

# Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法

# Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案