



江南大学  
JIANGNAN UNIVERSITY

# 太湖之光并行计算开发入门

尹达恒  
江南大学超级计算俱乐部

2019年7月

# **An Introduction to Parallel Programming on Sunway System**

**Tutorial and tips for parallel programming beginners on Sunway**

**Based on porting and optimizing of mlpack**

**By**

**Howard Yin**

**Supercomputer Club, Jiangnan University**

**July 2019**

## 前 言

- 本书实际上是一个教程形式的学习笔记；
- 本书主要的行文线索是 2018 年 12 月左右启动的机器学习算法库 darknet 在太湖之光系统中的移植和并行优化项目；
- 本书是超算中心下发的官方教程《“神威·太湖之光”计算机编译系统用户手册 Version 0.9》（本书简称《编译手册》）和《神威·太湖之光并行程序设计与优化（第一版）Ver1.2》（本书简称《优化手册》）中重要知识点的实践总结；
- 比起 Fortran 作者更擅长也更常用 C 语言，C 语言也能胜任超算系统中的大部分工作，因此本书所有代码均使用 C 语言编写，关于编译器也只涉及 Sunway 的 C 语言编译器；
- 本书不区分“‘神威·太湖之光’计算机系统”、“‘神威·太湖之光’系统”、“Sunway 系统”、“神威系统”和“太湖之光系统”，以及处理器名称“SW26010”、“Sunway26010”和“申威 26010”，请读者注意；
- 本书将作为每周的俱乐部工作报告持续更新。

### 学习本教程前需要掌握的知识

- C 语言，不仅是 C 语言的使用，编译器的运行流程和 CMake 都要了解一点。本书所有代码都是 C 语言，神威编译器也只介绍 C 语言编译器；
- 用命令行操作 Linux 系统（Sunway 的操作系统没有图形界面）；
- 看过某些并行计算教程开头的介绍章节（比如 CUDA 教程），大概知道并行计算是什么意思。

### 国家超级计算中心简介

神威·太湖之光超级计算机是由国家并行计算机工程技术研究中心研制、部署于国家超级计算无锡中心的超级计算机。类似的超级计算中心在国内共有 7 个，它们分别是：

- 天津中心：2009 年 6 月，滨海新区与国防科技大学签署合作协议共建国家超级计算天津中心。天津中心由国家科技部、滨海新区、开发区、国防科技大

学共同投资 6 亿元建设，于 2009 年底开始投入运营。定位为国家重大科技服务平台、产业技术创新平台、人才聚集培养平台，主要承接国家“863”重大科技专项，是重要的大规模集成电路设计中心和基础软件工程中心及产业化基地；

- 深圳中心：国家超级计算深圳中心又称深圳云计算中心，总投资 12.3 亿元。主机系统由中国科学院计算技术研究所研制、曙光信息产业（北京）有限公司制造，2010 年 5 月经世界超级计算机组织实测确认运算速度达每秒 1271 万亿次。深圳中心立足深圳、主要面向全国、服务华南、港、澳、台及东南亚地区，承担各种大规模科学计算和工程计算任务，同时以其强大的数据处理和存储能力为社会提供云计算服务；

- 长沙中心：国家超级计算长沙中心位于湖南大学校区内，其依托的计算设备是国防科技大学研制的“天河一号”超级计算机。长沙中心总投资 7.2 亿元，运算能力达每秒 300 万亿次，由湖南大学负责运营，国防科技大学提供计算设备和技术支持；

- 济南中心：国家超级计算济南中心主要依托的计算设备是“神威·蓝光”超级计算机，其持续性能可达 0.796PFlops。济南中心秉持立足山东、辐射周边、服务全国的工作思路，积极服务于山东省“两区一圈一带”发展战略需求，支持国家重大科技创新和战略性新兴产业，服务地方经济发展。2018 年 8 月，神威 E 级超算原型机在国家超级计算济南中心完成部署。

- 广州中心：国家超级计算广州中心由广东省人民政府、广州市人民政府、国防科技大学、中山大学共同建设，主要依托设备是“天河二号”超级计算机；

- 无锡中心：国家超级计算无锡中心是无锡市政府直属事业单位，由清华大学与无锡市政府共同建设，并委托清华大学管理运营。国家超级计算无锡中心依托的“神威·太湖之光”计算机系统是我国“十二五”期间“863 计划”的重大科研成果，由国家并行计算机工程技术研究中心研制，运算系统全面采用了由国家高性能集成电路设计中心通过自主核心技术研制的国产“申威 26010”众核处理器，是我国第一台全部采用国产处理器构建的世界排名第一的超级计算机；

- 郑州中心：2019 年 5 月，国家超级计算郑州中心获得科技部批复筹建，成为全国第 7 家批复建设的国家超级计算中心，也是科技部出台认定管理办法后批复筹建的首家国家超级计算中心。郑州中心拟依托郑州大学建设运营，计划于 2020 年上半年建设完成，峰值计算能力预期将达到 100PFlops。

## 神威·太湖之光简介

“神威·太湖之光”计算机系统 (Sunway TaihuLight) 采用全机水冷、直流供电、高密度组装，配有精确的资源调度管理算法、丰富的并行编程语言和开发环境。2016 年 6 月 20 日国际 TOP500 公布的排名数据中，“神威·太湖之光”计算机系统以每秒 12.54 亿亿次的峰值运算速度和 9.3 亿亿次的持续运算速度高居榜首。“神威·太湖之光”是中国第一台全部采用国内自主研发技术构建的超级计算机。详细的架构描述可见《优化手册》第一章。

## 申威 26010 简介

“神威·太湖之光”计算机系统采用的处理器是申威 26010(Sunway 26010)，该处理器由上海高性能集成电路设计中心自主研制，其指令集为自主研发的 64 位申威指令集，工作频率 1.5Ghz。该处理器不同于现有的纯 CPU、CPU-MIC、CPU-GPU 架构，而是采用了主-从核架构，每个申威 26010 中有 4 个处理器，每个处理器包含 1 个主核（运算控制核心）和 64 个从核（8x8 核心阵列），主从核的关系类似于 CPU-GPU 架构。详细的架构描述可见《优化手册》1.2 节。

## 同构计算和异构计算 [1]

同构计算是使用相同类型指令集和体系架构的计算单元组成系统的计算方式。。同构超算只单纯使用一种处理器，例如“神威·蓝光”只采用了 8704 片申威 1600。而异构计算主要是指使用不同类型指令集和体系架构的计算单元组成系统的计算方式。常见的计算单元类别包括 CPU、GPU 等协处理器、DSP、ASIC、FPGA 等。例如天河 2 号有 16000 个计算节点，每个节点由 2 片 Intel 的 E5 2692 和 3 片 Xeon PHI 组成，共使用了 32000 片 Intel 的 E5 2692 和 48000 片 Xeon PHI；天河 1A 使用了 14336 片 Intel Xeon X5670 处理器和 7168 片 NVIDIA Tesla M2050 高性能计算卡。相同功耗的情况下，异构超算能获得非常高的理论双精浮点性能和更高的性能功耗比，这也使异构处理器更受超级计算机系统偏爱。

“神威·太湖之光”计算机系统所采用的申威 26010 属于异构众核处理器。不同于“神威·蓝光”的同构申威 1600 处理器和天河 2 号的 CPU-GPU 架构，申威 26010 的异构模式相当于将 CPU 和 GPU 集成在一个芯片上，从“神威·太湖之光”整体上看，这相当于将 GPU 分散放置到了 CPU 的周围，使得单个处理器

同时具有 CPU 核 GPU 的功能。和 CPU-GPU 体系相比，这种架构的优势在于减少了计算过程中的数据传输时间，从而提高整体的运算速度。

## 目 录

<b>第 1 章 Sunway 系统的基本操作 .....</b>	<b>1</b>
1.1 登录 .....	1
1.1.1 登录 Sunway 系统 SSL VPN .....	1
1.1.2 SSH 登录超算系统 .....	1
1.2 第一个程序：矩阵相加 .....	3
1.2.1 单个 SW26010 处理器并行计算流程 .....	3
1.2.2 相关 Athread 函数 .....	3
1.2.3 并行编程思路 .....	5
1.3 程序的编译 .....	6
1.3.1 Sunway 程序编译简介 .....	6
1.3.2 用命令行编译程序 .....	7
1.4 程序的运行 .....	7
1.4.1 Sunway 程序运行简介 .....	7
1.4.2 用命令行运行程序 .....	8
1.5 使用 Linux make 进行编译和运行 .....	8
1.5.1 Linux make 和 Makefile 介绍其一 .....	8
1.5.2 Makefile 的格式 .....	9
1.5.3 编写 Makefile .....	10
1.6 总结 .....	11
1.6.1 知识点概括 .....	11
1.6.2 练习 .....	11
<b>第 2 章 Darknet 移植 .....</b>	<b>12</b>
2.1 Darknet 介绍 .....	12
2.2 Darknet 源码结构 .....	12
2.3 Darknet 的移植 .....	13
2.3.1 Linux make 和 Makefile 介绍其二 .....	13
2.3.2 Darknet 的 Makefile .....	16
2.3.3 数学公式 .....	18
2.3.4 数学环境 .....	18
2.3.5 表格 .....	19
2.3.6 图片插入 .....	19
2.3.7 算法 .....	21

---

2.3.8 参考文献引用 .....	21
2.4 常见使用问题 .....	22
<b>附录 A 第一个程序 .....</b>	<b>24</b>
A.1 主核 .....	24
A.2 从核 .....	25
<b>附录 B Darknet 的 Makefile .....</b>	<b>27</b>
<b>附录 C 中国科学院大学学位论文撰写要求 .....</b>	<b>30</b>
C.1 论文无附录者无需附录部分 .....	30
C.2 测试公式编号 $\Lambda, \lambda, \theta, \bar{\Lambda}, \sqrt{S_{NN}}$ .....	30
C.3 测试生僻字 .....	31
<b>参考文献 .....</b>	<b>33</b>
<b>图形列表 .....</b>	<b>35</b>
<b>表格列表 .....</b>	<b>36</b>
<b>作者简历及攻读学位期间发表的学术论文与研究成果 .....</b>	<b>37</b>
<b>致谢 .....</b>	<b>38</b>

# 第1章 Sunway 系统的基本操作

## 1.1 登录

未来 Sunway 官网和计算系统的登录方式可能变化，此处的登录方法仅供参考。超算中心下发了两套用户名和密码，一套用于登录 VPN，一套用于登录 SSH。

### 1.1.1 登录 Sunway 系统 SSL VPN

1. 使用 IE 浏览器<sup>1</sup>进入神威·太湖之光官网 [www.nsccwx.cn](http://www.nsccwx.cn);
2. 在右上角“登录”处选择一个运营商进行登录（图1.1a）;
3. 在弹出的警告页面<sup>2</sup>中选择“详细信息”→“转到此页面”（图1.1b）;
4. 在登录界面输入俱乐部下发的 VPN 用户名密码进行登录（图1.1c）;
5. 第一次登录完成后浏览器会自动下载 EasyConnect 工具，此工具为后续登录使用（图1.1d和图1.1e）。

再次登录时，打开第一次登录完成后自动下载的 EasyConnect 工具输入俱乐部下发的 VPN 用户名密码即可。

### 1.1.2 SSH 登录超算系统

1. 登录 VPN;
2. 在弹出的页面找到可用资源的 IP 地址（图1.1f）;
3. 在命令行窗口输入“ssh [俱乐部下发的用户名]@[可用资源的 IP 地址]”命令进行登录<sup>3</sup>，登录时会提示输入密码<sup>4</sup>，此处密码为 SSH 登录密码（图1.1g）;
4. 出现 bash 界面说明登录成功（图1.1h）。

<sup>1</sup>亲测 Edge 和 Chrome 在第一次登录时无法下载 EasyConnect 工具，卡在加载界面

<sup>2</sup>截至本章作成日 2018 年 12 月 1 日，太湖之光的官方网站服务器在使用自主生成的 SSL 证书进行 https 连接，这类非信任机构生成的证书会被现在的浏览器识别为不安全

<sup>3</sup>这里的用户名是 SSH 用户名

<sup>4</sup>多次登录有时会出现 WARNING 和登录失败，大部分情况下这都是神威系统为了安全对 SSH 登录验证的公钥进行定时修改所致（猜测）。这时只要删除用户目录下的 C:/用户/用户名/.ssh/known\_hosts 文件重新登录 SSH 即可（这个文件记录了所有登录过的 SSH 服务器的公钥，再次登录时会进行公钥比对，如果不符则登录失败）

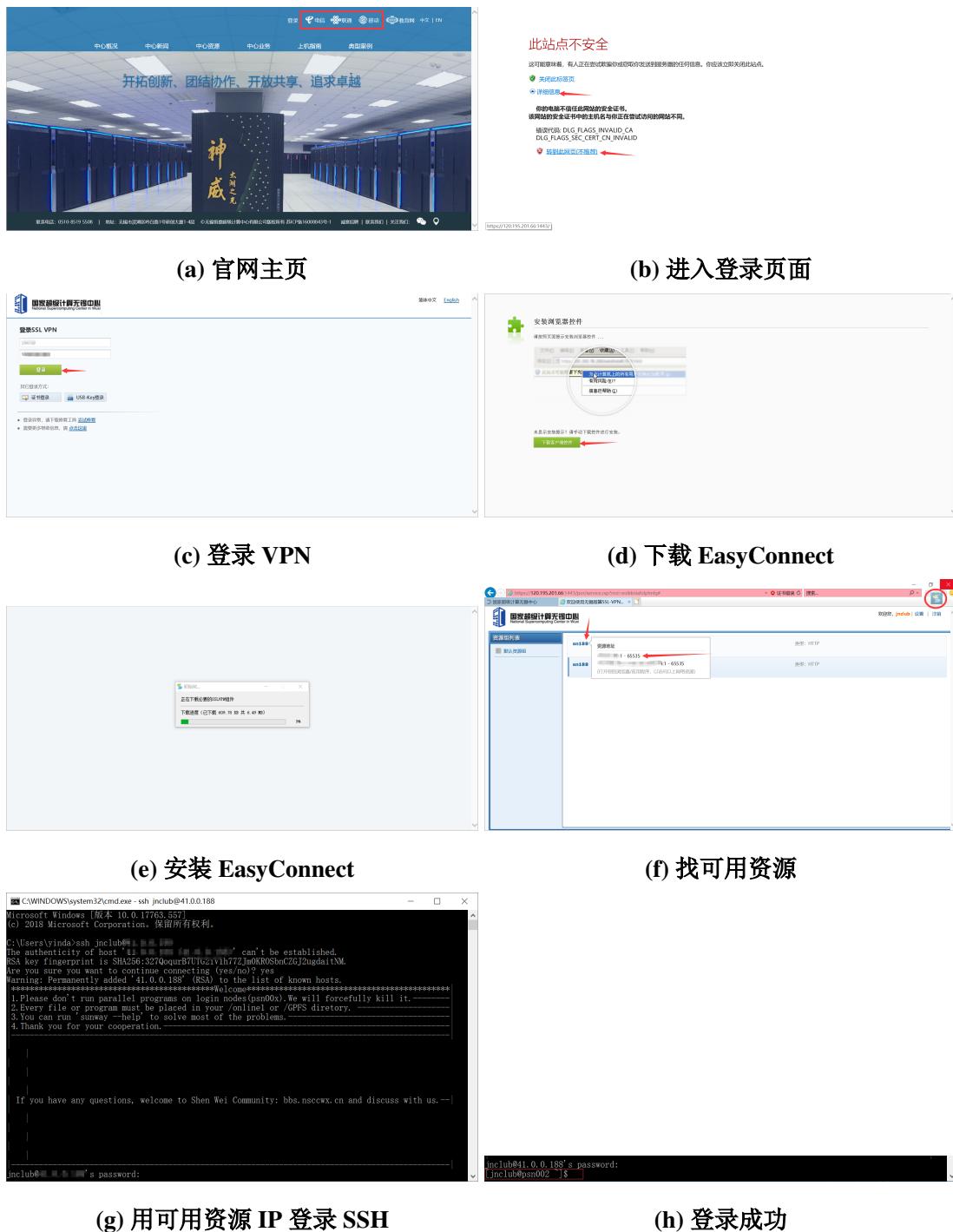


图 1.1 超算系统登录流程

## 1.2 第一个程序：矩阵相加

矩阵计算是所有并行计算的基础，第一个并行程序以矩阵相加程序为例，介绍在单个 SW26010 处理器上编程时 Sunway 编译器和 Athread 的一些基本操作。矩阵相加程序的要求非常简单：将两个  $64 \times 2048$  的矩阵相加。

### 1.2.1 单个 SW26010 处理器并行计算流程

正如前言中所说，每个 SW26010 芯片中都包含 4 个处理器，每个处理器有 1 个主核和 64 个从核，第一个程序的编写将着眼于在一个处理器上的编程，程序只使用 1 个主核和 64 个从核。

1. 运行于主核上的主程序启动  $N$  个并行执行的线程 ( $N \leq 64$ )，每个线程都在一个从核上执行；
2. 在并行执行的线程中，每个线程都从主存储中**不同的位置**取出数据放入从核各自的局部存储中<sup>56</sup>；
3. 从核在自己的局部存储上进行计算；
4. 从核计算完成后，都分别将计算结果放回到主存储中后结束自身的线程；
5. 在从核运行过程中主程序进行其他其他操作并等待所有从核线程结束；
6. 所有的从核线程结束（从核函数退出）之后，主程序进行下一步并行操作或结束。

### 1.2.2 相关 Athread 函数

下面是矩阵相加程序中使用到的并行函数相关必要知识的介绍，更加详细的描述见《编译手册》第五章“加速线程库”。

#### 1.2.2.1 在主核中调用的 Athread 函数

1. `athread_init()` 核组初始化，在所有 `athread` 操作之前都要进行调用；
2. `athread_set_num_threads()` 设置下一次并行操作启动的线程总数，如果不设置此项值则下一次并行操作 (`athread_spawn`) 启动 64 个线程；

---

<sup>5</sup>在一般的并行编程中都有在并行线程中获取线程编号的方法，并行线程即通过线程编号计算出自己要从哪个位置取数

<sup>6</sup>从主存储中取出数据放入从核的局部存储使用称作 DMA(Direct Memory Access，直接内存存取) 的方法，DMA 运行独立于主核和从核，可以在主核和从核进行计算的同时传输数据，在基于传输的并行优化方面尤其有用

3. `athread_spawn([函数指针],[传入参数])` 在从核上启动并行线程, [函数指针] 为指向从核函数入口的指针 (从核函数名), 且该指针需要在文件开头使用一个 Athread 库中的宏 “`extern SLAVE_FUN(从核函数名)()`” 进行声明; [传入参数] 为向每个从核函数传递的实参该参数会在运行时传递到每个并行从核函数线程中;
4. `athread_join()` 等待所有从核线程结束, 程序执行到此函数时会阻塞直到所有从核线程结束才会执行下一步;
5. `athread_halt()` 关闭线程组流水线, 在所有并行操作完成之后才能调用此函数, 在程序结束时调用, 调用后运算核心将无法在本进程中再次使用。

### 1.2.2.2 在从核中调用的 Athread 函数

1. `athread_get_id()` 获取线程逻辑标号, 一般在从核程序中通过此函数的值判断应该从主存储的那个位置取数据;
2. `athread_get([传输模式],[源地址],[目的地址],[数据量],[回答字地址],[],[],[])` 通过 DMA(见注6) 方法从主存中读取数据写入从核局部存储。各形参含义为:
  - 传输模式: DMA 传输命令模式, 本例程中只涉及使用 `PE_MODE`;
  - 源地址: 要传输的数据在主核中的地址 (在主核程序中的变量名)。对于数组形式的数据, 此处填数组变量的首地址, 就像下面这样<sup>7</sup>。多维数组的取址方法以此类推。

```

1 &源数组名[取数位置]//一维数组首地址位置
2 &源数组名[取数位置1][取数位置2]//二维数组首地址位置
3

```

上面写出的源数组名数组名需要在从核程序开头进行“`extern`”声明, 如下所示。

```
extern 数据类型 源数组名; //文件开头声明主核中的某个数组
```

2

- 目的地址: 从主存中取得的数据放入局部存储的那个地址 (变量) 中。对于数组形式的数据其写法和源地址相同, 且数组名需要在从核函数文件开头进

<sup>7</sup>这里的取数位置即是从核函数要从何处取数的标志 (见注5), 该值一般由 `athread_get_id()` 获取到的线程逻辑标号计算得出

行“`__thread_local`”声明，如下所示。

```
1 extern 数据类型 源数组名; //文件开头声明主核中的某个数组
```

- 数据量：从源地址开始读取多少字节数据到局部存储（目的地址）中。对于数组形式的数据，此参数的值为 [要读多少个数]\*[此数组的数据类型占多少字节]<sup>8</sup>。
- 回答字地址：当 `athread_get` 数据传输完成时，回答字地址中的值加 1。多个 `athread_get` 同时运行可以共用一个回答字，每个 `athread_get` 运行结束时都会使回答字地址中的值加 1。一般在等待 `athread_get` 传输完成的地方会有“`while([回答字]!=[共用此回答字的 athread_get 函数个数]);`”。回答字变量在从核函数文件开头以“`__thread_local volatile`”声明。
- 本例程不涉及最后三个参数。

### 1.2.3 并行编程思路

并行编程程序分主核和从核两个程序流程，请理解上文所述的各函数的作用并理解下列流程后自行编写程序。参考程序见附录A。

主核程序：

1. 生成两个 64x2048 矩阵；
2. `athread_init()` 初始化；
3. `athread_spawn()` 启动 64 个从核线程，每个线程处理每个 64X2048 矩阵中的 2048 个数；
4. `athread_join()` 等待线程全部结束；
5. `athread_halt()` 关闭线程流水线；
6. 输出结果，退出程序。

从核程序：

1. `athread_get_id()` 获取线程逻辑标号；
2. `athread_get()` 根据线程逻辑标号从主核程序变量中取得要用的数据，每个线程取在两个 64x2048 矩阵中各取 2048 个数；

<sup>8</sup>实测在太湖之光的编译系统中，int 型和 float 型占 4 个字节，long 型和 double 型占 8 个字节

3. 计算相加结果；
4. `athread_put()` 将结果写回主核程序变量中；
5. 退出程序。

### 1.3 程序的编译

“神威·太湖之光”系统的编译环境有两种，一种是服务于高速计算系统的编译环境，即 SW26010 搭建的神威主系统；另一种服务于辅助计算系统，该系统是常见的 Intel X86 CPU + GPU 结构服务器。这里只介绍高速计算系统环境下的程序编译。本节主要内容来自《优化手册》2.5 节“编译环境”。

#### 1.3.1 Sunway 程序编译简介

目前的“神威·太湖之光”系统支持的编程语言主要包括 C 语言 (C99)、C++ 语言 (C++03 和 C++11) 和 Fortran(Fortran2003)，其中 C++ 目前还不能编译从核程序。

神威系统的 C 语言编译器指令为“sw5cc”，编译模式分为主核和从核两种<sup>9</sup>，编译主核程序的指令为：

```
1 sw5cc -host [选项] 文件名.c
```

编译从核的指令为：

```
1 sw5cc -slave [选项] 文件名.c
```

主核程序和从核程序编译完成后，则使用下面这个命令进行混合链接生成可执行程序：

```
1 sw5cc -hybrid [选项] 主核文件名.o 从核文件名.o
```

本章只介绍编译器的基本使用方法，不过多涉及编译器的编译选项，详细了解编译选项可见《优化手册》表格 2-2。

<sup>9</sup>正如前言中所说，异构计算使用不同类型指令集和体系架构的计算单元组成系统，因此在编译程序时主核和从核使用的编译器并不相同

### 1.3.2 用命令行编译程序

假设在1.2.3节编写的主核程序源文件名“master\_arrAdd.c”、从核程序源文件名“slave\_arrAdd.c”，则可以输入下面这三条指令对源文件进行编译和链接：

1. 编译主核，-c选项表示为每个源文件生成一个.o文件但不进行链接

```
1 sw5cc -host -c master_arrAdd.c
```

2. 编译从核

```
1 sw5cc -slave -c slave_arrAdd.c
```

3. 链接，-o arrAdd选项表示输出的可执行文件名为“arrAdd”

```
1 sw5cc -hybrid master_arrAdd.o slave_arrAdd.o -o arrAdd
```

编译完成后即可在当前目录下看到一个可执行文件“arrAdd”。接下来介绍运行这个可执行文件的方法。

## 1.4 程序的运行

和上一节介绍的编译环境一样，神威系统的运行也分高速计算系统运行和辅助计算系统运行两种，这里也只介绍程序在高速计算系统中的运行，主要内容来自《优化手册》2.6节“作业提交”。

### 1.4.1 Sunway 程序运行简介

“神威·太湖之光”的高速计算系统计算资源由一个作业管理系统进行管理，其本质是一个队列。在高速计算系统上运行的程序称为“作业”，要运行某个作业时，将这个作业提交到作业管理系统的队列中，作业管理系统按照先进先出方式取出队列中的作业放到计算节点上运行。

俱乐部目前持有的账号可以向免费的开发调试计算队列提交作业：高速计算系统的q\_sw\_expr和辅助计算系统的q\_x86\_expr。这两个队列中每个作业的最长运行时间为60分钟，最大并行规模为64。

### 1.4.2 用命令行运行程序

神威系统提交作业的指令为“`bsub`”，其指令选项内容丰富，此处不一一展开，详细的选项可见《优化手册》2.6.2 节，这里只介绍几个常见的选项。

使用“`bsub`”指令运行1.3.2中编译完成的可执行程序“`arrAdd`”，可键入如下指令：

```
1 bsub -I -b -q q_sw_expr -n 1 -cgsp 64 ./arrAdd
```

指令选项解释：

- `-I`: 使作业输出在当前窗口；
- `-q q_sw_expr`: 将作业提交到 `q_sw_expr` 计算队列；
- `-b`: 指定从核栈位于局部存储，该选项使得从核程序先全部读入从核局部再运行，可以减少主从核数据传输的次数，对于并行计算的速度提升非常重要；
- `-n 1`: 指定程序要用的核组个数为 1；
- `-cgsp 64`: 指定每个核组内要用的从核个数为 64，由于一个 SW26010 一个核组只有 64 个从核，因此该值不能大于 64。该选项即影响程序中 `athread_spawn()` 生成的从核线程的个数；
- `./arrAdd`: 要执行的可执行文件位置。

若使用附录A中的程序，在程序执行完成后可得到并行和非并行计算的结果与时间对比。

## 1.5 使用 Linux make 进行编译和运行

### 1.5.1 Linux make 和 Makefile 介绍其一

在装有 GNU Make 的 Linux 系统上，`make` 是一条可执行指令，该指令读入一个名为 `Makefile` 的文件，通过执行这个文件中指定的指令完成程序的编译、安装和运行。该指令有如下功能：

1. 使最终用户可以在不知道构建细节的情况下构建和安装软件，构建细节都记录在所提供的 `Makefile` 中，用户只需要在 `Makefile` 文件所在目录下键入 `make` 指令即可；
2. `make` 可以通过读取和记录文件修改时间自动知道那些文件需要更新（重

新编译), 并且它也会通过 Makefile 中所记录的文件依赖关系自动决定文件更新的适当顺序, 使得相关的文件一并更新;

3. make 本质上只是一个系统指令的执行器, 只要是能通过键入指令完成构建和运行的程序都可以通过 make 完成构建和运行, 因此 make 的使用不限于任何一种特定的语言。对于程序中任何一种非源文件, makefile 文件中可以指定 shell 指令去生成它。shell 指令可以执行编译器生成目标文件, 执行链接器生成可执行文件, 执行 ar 更新库文件, 执行 TeX (一个文本排版软件) 或 Makeinfo 去格式化文档等。

### 1.5.2 Makefile 的格式

Makefile 文件的语法可以看作是一系列的任务定义, 定义任务的文本格式非常简单:

```
1 target:[dependencies \dots]
      command
3   command
  \dots
```

其各部分的用处如下:

- target: 定义任务名称。每个 Makefile 文件都包含有多个任务, 每个任务都有一个唯一的 target 指定这个任务的名称。任务的名称只是这个任务的一个代号, 但是为了可读性, 任务的编写和命名规则一般有如下约定:

- 将生成中间文件、编译、清理生成文件、运行等任务分开编写;
- 如果这个任务的作用是生成一个文件, 则任务名是要生成的文件名;
- 如果这个任务的作用是编译整个工程, 则任务名是“build”或是最终的可执行文件名;
- 如果这个任务是清理生成文件, 则任务名为“clean”;
- 如果这个任务是运行, 则任务名为“run”。

当在 Makefile 中定义了一个任务后, 可以使用“make target”指令 Makefile 中定义的某个特定任务; 而若直接键入“make”, 则默认运行 Makefile 中定义的第一个任务。

dependencies ... : 依赖项。该项指定当前任务与哪些文件有关或是在哪些任

务完成之后才能运行，dependencies 是文件名称或其他任务的任务名称（target），多个名称以空格分隔。在 make 指令调用此任务时，make 程序对 dependencies 中的每个名称的处理如下：

1. 如果有该名称对应的文件，则检查文件的修改日期，将其与上一次执行时存储的修改日期对比，若不同则说明文件被修改。如果在 dependencies 对应的文件中检测到任何一个文件被修改，则先执行该名称对应的任务（如果有的话，否则只执行本任务<sup>10</sup>）再执行本任务；
2. 如果有该名称对应的任务，则检查该任务的 dependencies，规则同上。

递归式地运行以上两步，直到检查的任务中没有名称对应任务，从而能使得一个文件被修改时执行且只执行相关的任务进行中间文件和输出文件的编译更新；

- **command:** 完成任务要执行的指令。该项指定当前任务要完成哪些指令，每个指令前必须要以 tab 缩进，指令间以换行分隔。当调用一个任务时，该任务下定义的指令将被按顺序执行。

### 1.5.3 编写 Makefile

以1.2.3中编写的程序和1.3.2中的编译过程为例编写一个简单的 Makefile：

```

1 arrAdd:master_arrAdd.o slave_arrAdd.o
2   sw5cc -hybrid master_arrAdd.o slave_arrAdd.o -o arrAdd
3 master_arrAdd.o:master_arrAdd.c
4   sw5cc -host -c master_arrAdd.c
5 slave_arrAdd.o:slave_arrAdd.c
6   sw5cc -slave -c slave_arrAdd.c
7 clean:
8   -rm master_arrAdd.o slave_arrAdd.o arrAdd
9 run:arrAdd
10  bsub -I -b -q q_sw_expr -n 1 -cgsp 64 ./arrAdd

```

按照前面所讲的文件格式和命名规则，该 Makefile 定义了 5 个任务，其中默认任务为“arrAdd”，该文件生成可执行文件“arrAdd”；此外还有一个运行任务“run”和清理任务“clean”。该文件放在和源文件同一目录下，则可在该目录下运行“make”编译程序或运行“make run”执行程序等操作。

<sup>10</sup>想想看这就是以要生成的文件名作为任务名的好处

## 1.6 总结

### 1.6.1 知识点概括

本章主要介绍了在神威系统编写程序的流程和必须要知道的一些基本操作，内容可大致概括为图1.2。

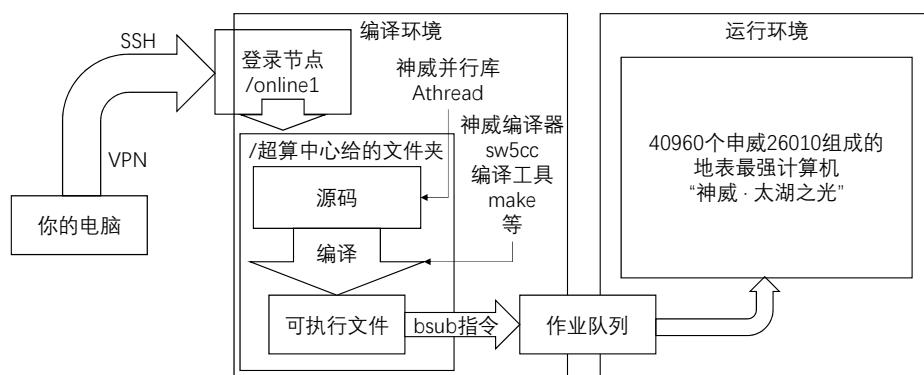


图 1.2 第 1 章的知识点概括

### 1.6.2 练习

- 用 Athread 实现并行的矩阵减法；
- 用 Athread 实现并行的矩阵乘法。

## 第 2 章 Darknet 移植

### 2.1 Darknet 介绍

Darknet 是一个开源的 C 语言神经网络框架，由著名的机器视觉专家 Joseph Redmon 开发 [2]。Darknet 性能优异，安装简单，在目标识别领域有广泛应用的 YOLO 模型 [3] 的最初版本就是在这个框架下开发的。作为“神威·太湖之光”并行编程的入门练习，Darknet 有如下优点 [4]：

- 易于安装：在 makefile 里面选择自己需要的附加项（cuda, cudnn, opencv 等）直接 make 即可；
- 没有任何依赖项：整个框架都用 C 语言进行编写，可以不依赖任何库，连 opencv 作者都编写了可以对其进行替代的函数；
- 结构清晰：其框架的基础文件都在 src 文件夹，而定义的一些检测、分类函数则在 example 文件夹，可根据需要直接对源代码进行查看和修改；
- 友好 python 接口：虽然 darknet 使用 c 语言进行编写，但是也提供了 python 的接口，通过 python 函数，能够使用 python 直接对训练好的.weight 格式的模型进行调用；
- 易于移植：该框架部署十分简单，且可以根据机器情况，使用 cpu 和 gpu，特别是检测识别任务的本地端部署，darknet 会显得异常方便。

此外，由于 Darknet 是一个神经网络框架，其中的主要运算任务都可以以矩阵计算的形式所表述，这使得 Darknet 的移植和优化有助于读者对矩阵和并行更加深入的理解。最后，Darknet 框架整体结构比较成熟完整，读者能在移植和优化过程中了解到优秀的机器学习框架的架构与编写方式，对软件架构能力的提升有一定帮助。

### 2.2 Darknet 源码结构

从 Joseph Redmon 的个人网站或 Github 上下载得到的 Darknet 源码包解压后可得到如图所示的文件和文件夹，各文件和文件夹的作用如下 [4]：

- cfg 文件夹内是一些模型的架构，每个 cfg 文件类似于 caffe 的 prototxt 文件，通过该文件定义的整个模型的架构；

- data 文件夹内放置了一些 label 文件，如 coco9k 的类别名等，和一些样例图，主要在训练和测试时使用；
- src 文件夹内是最底层的框架定义文件，所有层的定义等最基本的函数全部在该文件夹内，是 Darknet 框架的源码所在；
- examples 文件夹是更为高层的一些函数，如检测函数，识别函数等，这些函数直接调用了底层的函数，是对底层函数的封装；
- include 文件夹，顾名思义，存放头文件的地方；
- python 文件夹里是使用 python 对模型的调用方法，基本都在 darknet.py 中。要实现 python 的调用，还需要用到 darknet 的动态库 libdarknet.so；
- scripts 文件夹中是一些脚本，如下载 coco 数据集，将 voc 格式的数据集转换为训练所需格式的脚本等；
- 一系列和功能无关的 license 文件；
- 本章的重点 Makefile 文件，用于框架的编译和运行。

## 2.3 Darknet 的移植

### 2.3.1 Linux make 和 Makefile 介绍其二

在1.5.1节中已经介绍过 Makefile 的文件结构和 Linux make 的基本使用方法，在本节中将进一步介绍 Makefile 文件中的变量定义和计算，为理解 Darknet 的编译过程打下基础。

在 Makefile 文件中，如果有某些指令和名称比较长或是需要在文件不同位置多次使用，为了保证可读性和修改的方便则需要用类似 C 语言宏定义的方式，用一些“宏”（更准确地说应该是变量）代表这些指令或名称。Linux make 和 Makefile 即提供了这样的功能，这种功能的存在也使得 Makefile 被称为“make 脚本”<sup>1</sup>。这一功能可以概况为如下几种语法：

- 变量定义：

变量名 = 值；

<sup>2</sup>

<sup>1</sup> 为啥叫“脚本”？“脚本”的英文是“script”，可以联想两种解释型编程语言：js 和 python。js 全称是“JavaScript”，python 的源代码文件也叫“python script”，它们都是以解释器解释脚本文件中源代码的方式运行的。make 也可以看作是一个解释型编程语言的解释器，Makefile 就是它读入并解释的脚本文件，1.5.1节和本节就是在介绍这个编程语言的语法

- 变量调用:

```
1 $(变量名)
```

- 变量修改:

```
1 变量名+=$(变量名)或值
```

```
1 变量名=$(变量名)或值+$(变量名)或值+...
```

- 变量比较和 if 语句:

```
1 ifeq ($(变量名), $(变量名)或值)
[一些操作]
3 endif
```

- 内嵌函数:

```
1 $(函数名 变量1, 变量2,...)
```

- 静态模式: 可以将多个任务合并于一个任务定义中, 如下任务运行前 \$@ 将被文件路径%.o 替代、\$< 将被文件路径%.cpp 替代, 生成每个.o 和每个.cpp 一一对应的多个指令:

```
1 [文件路径]%.o: [文件路径]%.cpp [其他 dependences]
[一部分指令] $< [一部分指令] $@
3
```

而下面这种的 `$@` 将被 `target` 替代、`$^` 被 `dependences` 替代，但没有上面那种多重替代的效果：

```
[target]: [dependences]
2      [一部分指令] $^ [一部分指令] $@
```

`Makefile` 中的所有变量均是字符串，变量修改中的“+”表示的就是字符串的连接，变量比较就是字符串的比较；字符串的调用可以在文件的任何地方，Linux `make` 在运行时会自动将 `Makefile` 中的变量调用替换为变量对应的字符串<sup>2</sup>。例如1.5.3中的 `Makefile`：

```
arrAdd:master_arrAdd.o slave_arrAdd.o
2      sw5cc -hybrid master_arrAdd.o slave_arrAdd.o -o arrAdd
master_arrAdd.o:master_arrAdd.c
4      sw5cc -host -c master_arrAdd.c
slave_arrAdd.o:slave_arrAdd.c
6      sw5cc -slave -c slave_arrAdd.c
clean:
8      -rm master_arrAdd.o slave_arrAdd.o arrAdd
run:arrAdd
10     bsub -I -b -q q_sw_expr -n 1 -cgsp 64 ./arrAdd
```

把其中的输出文件名“`arrAdd`”、两个.o 文件和 `bsub` 的运行选项用变量表示，可将 `Makefile` 改写如下：

```
EXEC=arrAdd
2 OBJS=master_arrAdd.o slave_arrAdd.o

4 OPT=-I -b
OPT+=-q q_sw_expr
6 OPT+=-n 1
OPT+=-cgsp 64
8
$(EXEC):$(OBJS)
10    sw5cc -hybrid $(OBJS) -o $(EXEC)
master_arrAdd.o:master_arrAdd.c
12    sw5cc -host -c master_arrAdd.c
slave_arrAdd.o:slave_arrAdd.c
14    sw5cc -slave -c slave_arrAdd.c
clean:
```

<sup>2</sup>就像 C 语言的宏一样

```

16      -rm $(OBJS) $(EXEC)
run:$(EXEC)
18      bsub $(OPT) ./$(EXEC)

```

静态模式语法可以以一个任务定义同时定义多个任务，例如当上面的 Makefile 要编译多个主核和从核源文件时，可以用内嵌函数和静态模式语法进一步缩写：

```

MASTER=master/
2 SLAVE=slave/
OBJDIR=obj/
4
EXEC=arrAdd
6 OBJ=master_arrAdd1.o master_arrAdd2.o slave_arrAdd1.o slave_arrAdd2.o
OBJS=$(addprefix $(OBJDIR), $(OBJ))
8
OPT=-I -b
10 OPT+=-q q_sw_expr
OPT+=-n 1
12 OPT+=-cgsp 64

14 $(EXEC):$(OBJS)
    sw5cc -hybrid $^ -o $@
16 $(OUTDIR)%.o:$(MASTER)%.c
    sw5cc -host -c $< -o $@
18 $(OUTDIR)%.o:$(SLAVE)%.c
    sw5cc -slave -c $< -o $@
20 clean:
    -rm $(OBJS) $(OUTPUT)
22 run:$(EXEC)
    bsub $(OPT) ./$@

```

在此 Makefile 目录下运行 Linux make 时，行 16~19 的两个静态模式语法会使得 make 从 MASTER 变量和 SLAVE 变量所指目录下找到所有.c 文件编译后放入 OUTDIR 所指目录中，即将多个任务用一个任务完成定义。在大型项目中善用此方法可以大大减少 Makefile 文件的长度，使项目更易于维护。

### 2.3.2 Darknet 的 Makefile

官网下载的 Darknet 源码中的 Makefile 文件如附录B所示，本节将对这个 Makefile 文件及其所定义的编译过程进行解析。

### 2.3.2.1 变量定义

Makefile 文件 1~30 行是变量的定义，其各部分作用如下：

- 行 1~5：整体的编译设定。指定编译过程中是否包含 GPU、CUDNN、OpenCV、OpenMP 和 DEBUG 模式。这些选项会在第 32~59 行被一系列 ifeq 调用，根据其值改变编译选项；
- 行 7~12：NVCC 的编译选项。这个变量在第 92 行编译.cu 文件时调用。NVCC 是 Nvidia C Compiler，编译 CUDA 的.cu 文件的编译器，这里的这些编译选项表示编译的目标设备的算力<sup>3</sup>；
- 行 16~20：SLIB 和 ALIB 是生成的链接库的.so 和.a 文件名称、EXEC 是生成的可执行文件的名称、OBJDIR 是中间文件的存储路径，它们都在后面的编译过程中被调用。VPATH 是 Makefile 的一个特殊变量，VPATH 中指代的目录下的文件在后面的过程中不需要再输完整路径<sup>4</sup>，多个目录以冒号分隔。VPATH 实际发挥作用的地方是行 85~92 三个静态模式语法中没有路径的.c、.cpp、.cu 源文件；
- 行 22~30：CC、CPP、NVCC、AR 都是 GCC 里的编译器，ARFLAGS、OPTS、LDFLAGS、COMMON、CFLAGS 都是这些编译器的编译选项，这些变量也都在后面的编译过程中被调用。

### 2.3.2.2 编译选项的设置

Makefile 文件 32~73 行是编译选项的设置，其主要作用有二：

- 用 ifeq 根据行 1~5 定义的整体编译设定修改行 22~30 定义的编译选项；
- 确定输出文件的文件位置，以供 make 检查文件更新使用（行 68 和 69 的内嵌函数 addprefix 是加前缀，行 70 的 \$(wildcard src/\*.h) 是返回 src 目录下的所有.h 文件路径）。这些文件位置都会在后面的编译过程被调用。

### 2.3.2.3 编译任务

从行 76 开始的所有部分均是和编译相关的任务定义。其各部分作用如下：

- 行 72：默认 make 任务，生成 obj、backup、results 三个目录（行 94~99 的任务），生成静态链接库和动态链接库（行 79~83 ALIB 和 SLIB 变量值对应的任

<sup>3</sup>对于不同算力的 Nvidia 卡 NVCC 有不同的优化策略

<sup>4</sup>就像 Windows 和 Linux 里面的环境变量

务), 生成可执行文件 (EXEC 变量值对应的任务);

- 行 76~77: 生成可执行文件。该任务依赖于 obj 目录下的一系列.o 文件 (变量 EXECOBJ) 和静态链接库 (变量 ALIB);
- 行 79~83: 静态模式语法, 用 obj 目录下的一系列.o 文件 (变量 OBJS) 生成静态链接库和动态链接库;
- 行 85~92: 静态模式语法, 用源代码生成 obj 目录下一系列.o 文件;
- 行 94~104: 编译相关的其他任务。.PHONY: clean 使 clean 任务必被执行。

### 2.3.3 开始移植 Darknet

做了这么多前期学习, 终于可以正式开始移植 Darknet 了。但是有了前面那些分析感觉移植 Darknet 的这一节没啥好讲的了。移植主要是改 Makefile 文件, 这里简单记录一下我改了哪吧。读者请自己下载 Darknet 在神威上编译, 按照编译错误信息一步步进行修改, 下面这个修改过程也是对着错误信息一步步出来的, 仅供参考。

#### 2.3.3.1 Makefile 文件修改

1. 编译器首先要修改的肯定是编译器。要把原来的 C 语言编译器 (变量 CC) 和 C++ 编译器 (变量 CPP) 的值改成神威里的编译器 sw5cc 和 sw5CC;
2. 编译选项其次是要改编译选项 (变量 CFLAGS)。原来的一些编译选项在 sw5cc 里面是不支持的, 然后因为目前只是移植所以编译选项设置为-host 全部在主核编译运行;
3. 连接选项: 在连接任务中, sw5cc 连接时要加上 -hybrid 选项;
4. 其他修改
  - 生成动态链接库有点问题, 目前无法解决, 故先删去生成动态链接库的任务;
    - 为了之后方便优化和维护继续往 Makefile 里加东西时不会和原来不支持的编译模式弄混, 最好把所有神威系统不支持的编译模式删掉, 包括 CUDA、OpenMP 和 OpenCV 编译, 只保留没有任何依赖的普通模式和 debug 编译模式。

#### 2.3.3.2 源代码文件修改

1. include/darknet.h 这个文件在编译时报类型重定义错误, 原因是里面的 network 和 layer 类型嵌套方式的问题, 改成能正常编译的 struct 嵌套就行了。

### 2.3.4 数学公式

比如 Navier-Stokes 方程（方程 (2.1)）：

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0 & \text{times math test : 1, 2, 3, 4, 5, 1, 2, 3, 4, 5} \\ \frac{\partial(\rho \mathbf{V})}{\partial t} + \nabla \cdot (\rho \mathbf{V} \mathbf{V}) = \nabla \cdot \boldsymbol{\sigma} & \text{times text test: 1, 2, 3, 4, 5} \\ \frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\rho E \mathbf{V}) = \nabla \cdot (k \nabla T) + \nabla \cdot (\boldsymbol{\sigma} \cdot \mathbf{V}) \end{cases} \dots (2.1)$$

$$\frac{\partial}{\partial t} \int_{\Omega} u \, d\Omega + \int_S \mathbf{n} \cdot (u \mathbf{V}) \, dS = \dot{\phi} \dots (2.2)$$

$$\mathcal{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, dt, \quad \mathcal{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, dt$$

$$\mathcal{F}(f(x + x_0)) = \mathcal{F}(f(x)) e^{2\pi i \xi x_0}, \quad \mathcal{F}(f(x + x_0)) = \mathcal{F}(f(x)) e^{2\pi i \xi x_0}$$

数学公式常用命令请见 [WiKibook Mathematics](#)。artracom.sty 中对一些常用数据类型如矢量矩阵等进行了封装，这样的好处是如有一天需要修改矢量的显示形式，只需单独修改 artracom.sty 中的矢量定义即可实现全文档的修改。

### 2.3.5 数学环境

**公理 2.1.** 这是一个公理。

**定理 2.2.** 这是一个定理。

**引理 2.3.** 这是一个引理。

**推论 2.4.** 这是一个推论。

**断言 2.5.** 这是一个断言。

**命题 2.6.** 这是一个命题。

证明. 这是一个证明。 □

**定义 2.1.** 这是一个定义。

**例 2.1.** 这是一个例子。

注. 这是一个注。

**表 2.1 这是一个样表。****Table 2.1 This is a sample table.**

行号	跨多列的标题							
Row 1	1	2	3	4	5	6	7	8
Row 2	1	2	3	4	5	6	7	8
Row 3	1	2	3	4	5	6	7	8
Row 4	1	2	3	4	5	6	7	8

### 2.3.6 表格

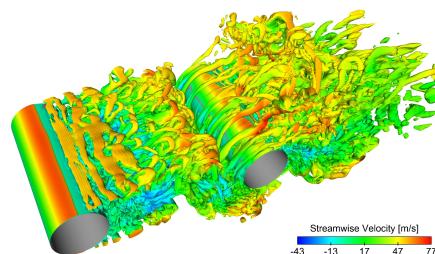
请见表 2.1。

制图制表的更多范例，请见 [ucasthesia 知识小站](#) 和 [WiKibook Tables](#)。

### 2.3.7 图片插入

论文中图片的插入通常分为单图和多图，下面分别加以介绍：

单图插入：假设插入名为tc\_q\_criteria（后缀可以为.jpg、.png、.pdf，下同）的图片，其效果如图2.1。



**图 2.1 Q 判据等值面图，同时测试一下一个很长的标题，比如这真的是一个很长很长很长很长很长很长很长很长很长很长的标题。**

**Figure 2.1 Isocontour of Q criteria, at the same time, this is to test a long title, for instance, this is a really very long very long very long very long very long title.**

如果插图的空白区域过大，以图片shock\_cyn为例，自动裁剪如图2.2。

多图的插入如图2.3，多图不应在子图中给文本子标题，只要给序号，并在主标题中进行引用说明。

### 2.3.8 算法

如见算法 1，详细使用方法请参见文档 [algorithmicx](#)。

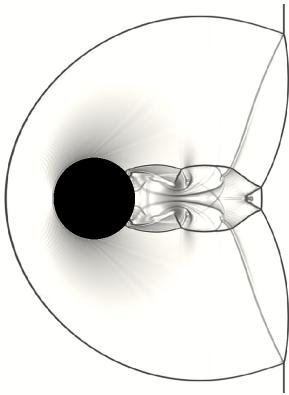


图 2.2 激波圆柱作用。

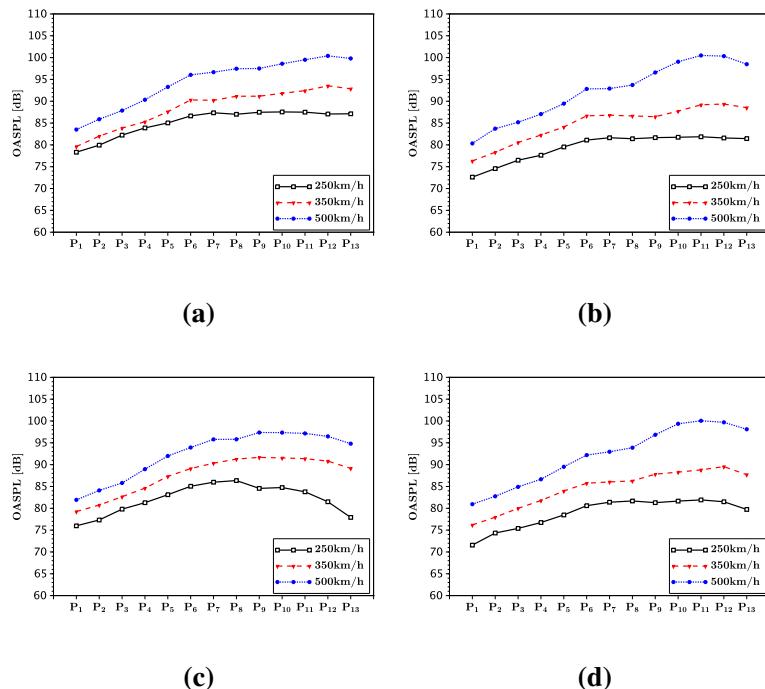
**Figure 2.2 Shock-cylinder interaction.**

图 2.3 总声压级。(a) 这是子图说明信息, (b) 这是子图说明信息, (c) 这是子图说明信息, (d) 这是子图说明信息。

**Figure 2.3 OASPL.(a) This is the explanation of subfig, (b) This is the explanation of subfig, (c) This is the explanation of subfig, (d) This is the explanation of subfig.**

---

**算法 1 Euclid's algorithm**

---

```

1: procedure Euclid( $a, b$ ) ▷ The g.c.d. of a and b
2:    $r \leftarrow a \bmod b$ 
3:   while  $r \neq 0$  do ▷ We have the answer if r is 0
4:      $a \leftarrow b$ 
5:      $b \leftarrow r$ 
6:      $r \leftarrow a \bmod b$ 
7:   end while
8:   return  $b$  ▷ The gcd is b
9: end procedure

```

---

### 2.3.9 参考文献引用

参考文献引用过程以实例进行介绍,假设需要引用名为”Document Preparation System”的文献, 步骤如下:

1) 使用 Google Scholar 搜索 Document Preparation System, 在目标条目下点击 Cite, 展开后选择 Import into BibTeX 打开此文章的 BibTeX 索引信息, 将它们 copy 添加到 ref.bib 文件中 (此文件位于 Biblio 文件夹下)。

2) 索引第一行 @article{lamport1986document, 中 lamport1986document 即为此文献的 label (中文文献也必须使用英文 label, 一般遵照: 姓氏拼音 + 年份 + 标题第一字拼音的格式), 想要在论文中索引此文献, 有两种索引类型:

文本类型: \citet{lamport1986document}。正如此处所示 Lampert [5];

括号类型: \citep{lamport1986document}。正如此处所示 [5]。

**多文献索引用英文逗号隔开:**

\citep{lamport1986document, chu2004tushu, chen2005zhulu}。正如此处所示 [5–7]

更多例子如:

Walls 等 [8] 根据 Betts 和 Taylor [9] 的研究, 首次提出...。其中关于... [8, 9], 是当前中国... 得到迅速发展的研究领域 [10, 11]。引用同一著者在同一年份出版的多篇文献时, 在出版年份之后用英文小写字母区别, 如: [12–14] 和 袁训来等 [12, 13, 14]。同一处引用多篇文献时, 按出版年份由近及远依次标注。例如 [10, 15–17]。

使用著者-出版年制(authoryear)式参考文献样式时,中文文献必须在 BibTeX 索引信息的 **key** 域(请参考 ref.bib 文件)填写作者姓名的拼音,才能使得文献列表按照拼音排序。参考文献表中的条目(不排序号),先按语种分类排列,语种顺序是:中文、日文、英文、俄文、其他文种。然后,中文按汉语拼音字母顺序排列,日文按第一著者的姓氏笔画排序,西文和俄文按第一著者姓氏首字母顺序排列。如中 [17]、日 [18]、英 [15]、俄 [19]。

如此,即完成了文献的索引,请查看下本文档的参考文献一章,看看是不是就是这么简单呢?是的,就是这么简单!

不同文献样式和引用样式,如著者-出版年制(authoryear)、顺序编码制(numbers)、上标顺序编码制(super)可在 Thesis.tex 中对 artratex.sty 调用实现,详见 [ucasthesia 知识小站之文献样式](#)

参考文献索引的更多知识,请见 [WiKibook Bibliography](#)。

## 2.4 常见使用问题

1. 模板每次发布前,都已在 Windows, Linux, MacOS 系统上测试通过。下载模板后,若编译出现错误,则请见 [ucasthesia 知识小站](#) 的 [编译指南](#)。

2. 模板文档的编码为 UTF-8 编码。所有文件都必须采用 UTF-8 编码,否则编译后生成的文档将出现乱码文本。若出现文本编辑器无法打开文档或打开文档乱码的问题,请检查编辑器对 UTF-8 编码的支持。如果使用 WinEdt 作为文本编辑器(**不推荐使用**),应在其 Options -> Preferences -> wrapping 选项卡下将两种 Wrapping Modes 中的内容:

TeX;HTML;ANSI;ASCII|DTX...

修改为: TeX;UTF-8|ACP;HTML;ANSI;ASCII|DTX...

同时,取消 Options -> Preferences -> Unicode 中的 Enable ANSI Format。

3. 推荐选择 xelatex 或 lualatex 编译引擎编译中文文档。编译脚本的默认设定为 xelatex 编译引擎。你也可以选择不使用脚本编译,如直接使用 L<sup>A</sup>T<sub>E</sub>X 文本编辑器编译。注: L<sup>A</sup>T<sub>E</sub>X 文本编辑器编译的默认设定为 pdflatex 编译引擎,若选择 xelatex 或 lualatex 编译引擎,请进入下拉菜单选择。为正确生成引用链接和参考文献,需要进行**全编译**。

4. Texmaker 使用简介

- (a) 使用 Texmaker “打开 (Open)” Thesis.tex。
- (b) 菜单“选项 (Options)”->“设置当前文档为主文档 (Define as Master Document)”
- (c) 菜单“自定义 (User)”->“自定义命令 (User Commands)”->“编辑自定义命令 (Edit User Commands)”->左侧选择“command 1”，右侧“菜单项 (Menu Item)”填入 Auto Build -> 点击下方“向导 (Wizard)”->“添加 (Add)”: xelatex + bibtex + xelatex + xelatex + pdf viewer -> 点击“完成 (OK)”
- (d) 使用 Auto Build 编译带有未生成引用链接的源文件，可以仅使用 xelatex 编译带有已经正确生成引用链接的源文件。
- (e) 编译完成，“查看 (View)” PDF，在 PDF 中“ctrl+click”可链接到相对应的源文件。

5. 模版的设计可能地考虑了适应性。致谢等所有条目都是通过最为通用的  
`\chapter{item name}` and `\section*{item name}`  
 来显式实现的(请观察 Backmatter.tex)，从而可以随意添加，放置，和修改，  
 如同一般章节。对于图表目录名称则可在 ucastheses.cfg 中进行修改。

6. 设置文档样式: 在 artratex.sty 中搜索关键字定位相应命令，然后修改

- (a) 正文行距: 启用和设置 `\linespread{1.5}`，默认 1.5 倍行距。
- (b) 参考文献行距: 修改 `\setlength{\bibsep}{0.0ex}`
- (c) 目录显示级数: 修改 `\setcounter{tocdepth}{2}`
- (d) 文档超链接的颜色及其显示: 修改 `\hypersetup`

7. 文档内字体切换方法:

- 宋体: 国科大论文模板 `ucastheses` 或 国科大论文模板 `ucastheses`
- 粗宋体: **国科大论文模板 `ucastheses`** 或 **国科大论文模板 `ucastheses`**
- 黑体: 国科大论文模板 `ucastheses` 或 国科大论文模板 `ucastheses`
- 粗黑体: **国科大论文模板 `ucastheses`** 或 **国科大论文模板 `ucastheses`**
- 仿宋: 国科大论文模板 `ucastheses` 或 国科大论文模板 `ucastheses`
- 粗仿宋: 国科大论文模板 `ucastheses` 或 国科大论文模板 `ucastheses`
- 楷体: 国科大论文模板 `ucastheses` 或 国科大论文模板 `ucastheses`
- 粗楷体: **国科大论文模板 `ucastheses`** 或 **国科大论文模板 `ucastheses`**

## 附录 A 第一个程序

### A.1 主核

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <pthread.h>
4 #include <sys/time.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>

9 extern SLAVE_FUN(func)();

11 #define X 64
12 #define Y 2048
13
14 int A[X][Y], B[X][Y], C[X][Y],CC[X][Y];
15
16 void init()
17 {
18     int i, j;
19     for (i = 0; i < X; i++)
20         for (j = 0; j < Y; j++)
21         {
22             A[i][j] = i + j;
23             B[i][j] = i + j + 1;
24             C[i][j] = 0;
25             CC[i][j]=0;
26         }
27     printf("Init finished\n");
28 }
29
30 int main(void)
31 {
32     int i, j;
33     struct timeval start, end;
34     init();
35     double time_use;
36     printf("No boost proceed:\n");
37     gettimeofday(&start, NULL);
38     for (i = 0; i < X; i++)
39         for (j = 0; j < Y; j++)
40             C[i][j] = A[i][j] + B[i][j];
41     gettimeofday(&end, NULL);
42     time_use = 1000000 * (end.tv_sec - start.tv_sec) + end.tv_usec - start.tv_usec;

```

```

43     printf("Time usage is %lf us\n", time_use);
44     printf("(C[32][0], C[63][999]) = (%d, %d)\n", C[32][0], C[63][999]);
45     init();
46     athread_init();
47     printf("Boosted proceed:\n");
48     gettimeofday(&start, NULL);
49     athread_spawn(func, 0);
50     athread_join();
51     gettimeofday(&end, NULL);
52     time_use = 1000000 * (end.tv_sec - start.tv_sec) + end.tv_usec - start.tv_usec;
53     printf("Time usage is %lf us\n", time_use);
54     printf("(C[32][0], C[63][999]) = (%d, %d)\n", C[32][0], C[63][999]);
55     athread_halt();
56     return 0;
57 }
```

## A.2 从核

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <string.h>
4 #include "slave.h"
5 #define X 64
6 #define Y 2048
7
8 #define Y0 64
9 #define N 32
10 //Y0*N=Y
11
12 #define L0 256
13 //L0=Y*4
14
15 __thread_local int my_id;
16 __thread_local volatile unsigned long get_reply[N], put_reply = 0;
17 extern int A[X][Y], B[X][Y], C[X][Y], CC[X][Y];
18
19 __thread_local int A_slave[Y], B_slave[Y], C_slave[Y];
20
21 void func()
22 {
23     int i, j, t, tt;
24     put_reply=0;
25     my_id = athread_get_id(-1);
26
27     for (i = 0; i < N; i++)//输入数据
28         for (j = 0; j < N; j++)
29             for (t = 0; t < N; t++)
30                 for (tt = 0; tt < N; tt++)
31                     if (my_id == i)
32                         if (i == j)
33                             if (j == t)
34                                 if (t == tt)
35                                     if (A[i][j] > B[i][j])
36                                         C[i][j] = A[i][j];
37                                     else
38                                         C[i][j] = B[i][j];
39
40         if (put_reply > 0)
41             if (put_reply < N)
42                 if (put_reply == my_id)
43                     if (put_reply == i)
44                         if (put_reply == j)
45                             if (put_reply == t)
46                                 if (put_reply == tt)
47                                     if (put_reply == A_slave[j])
48                                         if (put_reply == B_slave[t])
49                                         if (put_reply == C_slave[tt])
50                                             if (put_reply == CC[i][j])
51                                                 if (put_reply == 0)
52
53         if (put_reply > 0)
54             if (put_reply < N)
55                 if (put_reply == my_id)
56                     if (put_reply == i)
57                         if (put_reply == j)
58                             if (put_reply == t)
59                                 if (put_reply == tt)
60                                     if (put_reply == A_slave[j])
61                                         if (put_reply == B_slave[t])
62                                         if (put_reply == C_slave[tt])
63                                         if (put_reply == CC[i][j])
64                                         if (put_reply == 0)
65
66         if (put_reply > 0)
67             if (put_reply < N)
68                 if (put_reply == my_id)
69                     if (put_reply == i)
70                         if (put_reply == j)
71                             if (put_reply == t)
72                                 if (put_reply == tt)
73                                     if (put_reply == A_slave[j])
74                                         if (put_reply == B_slave[t])
75                                         if (put_reply == C_slave[tt])
76                                         if (put_reply == CC[i][j])
77                                         if (put_reply == 0)
78
79         if (put_reply > 0)
80             if (put_reply < N)
81                 if (put_reply == my_id)
82                     if (put_reply == i)
83                         if (put_reply == j)
84                             if (put_reply == t)
85                                 if (put_reply == tt)
86                                     if (put_reply == A_slave[j])
87                                         if (put_reply == B_slave[t])
88                                         if (put_reply == C_slave[tt])
89                                         if (put_reply == CC[i][j])
90                                         if (put_reply == 0)
91
92         if (put_reply > 0)
93             if (put_reply < N)
94                 if (put_reply == my_id)
95                     if (put_reply == i)
96                         if (put_reply == j)
97                             if (put_reply == t)
98                                 if (put_reply == tt)
99                                     if (put_reply == A_slave[j])
100                                    if (put_reply == B_slave[t])
101                                    if (put_reply == C_slave[tt])
102                                    if (put_reply == CC[i][j])
103                                    if (put_reply == 0)
104
105         if (put_reply > 0)
106             if (put_reply < N)
107                 if (put_reply == my_id)
108                     if (put_reply == i)
109                         if (put_reply == j)
110                             if (put_reply == t)
111                                 if (put_reply == tt)
112                                     if (put_reply == A_slave[j])
113                                         if (put_reply == B_slave[t])
114                                         if (put_reply == C_slave[tt])
115                                         if (put_reply == CC[i][j])
116                                         if (put_reply == 0)
117
118         if (put_reply > 0)
119             if (put_reply < N)
120                 if (put_reply == my_id)
121                     if (put_reply == i)
122                         if (put_reply == j)
123                             if (put_reply == t)
124                                 if (put_reply == tt)
125                                     if (put_reply == A_slave[j])
126                                         if (put_reply == B_slave[t])
127                                         if (put_reply == C_slave[tt])
128                                         if (put_reply == CC[i][j])
129                                         if (put_reply == 0)
130
131         if (put_reply > 0)
132             if (put_reply < N)
133                 if (put_reply == my_id)
134                     if (put_reply == i)
135                         if (put_reply == j)
136                             if (put_reply == t)
137                                 if (put_reply == tt)
138                                     if (put_reply == A_slave[j])
139                                         if (put_reply == B_slave[t])
140                                         if (put_reply == C_slave[tt])
141                                         if (put_reply == CC[i][j])
142                                         if (put_reply == 0)
143
144         if (put_reply > 0)
145             if (put_reply < N)
146                 if (put_reply == my_id)
147                     if (put_reply == i)
148                         if (put_reply == j)
149                             if (put_reply == t)
150                                 if (put_reply == tt)
151                                     if (put_reply == A_slave[j])
152                                         if (put_reply == B_slave[t])
153                                         if (put_reply == C_slave[tt])
154                                         if (put_reply == CC[i][j])
155                                         if (put_reply == 0)
156
157         if (put_reply > 0)
158             if (put_reply < N)
159                 if (put_reply == my_id)
160                     if (put_reply == i)
161                         if (put_reply == j)
162                             if (put_reply == t)
163                                 if (put_reply == tt)
164                                     if (put_reply == A_slave[j])
165                                         if (put_reply == B_slave[t])
166                                         if (put_reply == C_slave[tt])
167                                         if (put_reply == CC[i][j])
168                                         if (put_reply == 0)
169
170         if (put_reply > 0)
171             if (put_reply < N)
172                 if (put_reply == my_id)
173                     if (put_reply == i)
174                         if (put_reply == j)
175                             if (put_reply == t)
176                                 if (put_reply == tt)
177                                     if (put_reply == A_slave[j])
178                                         if (put_reply == B_slave[t])
179                                         if (put_reply == C_slave[tt])
180                                         if (put_reply == CC[i][j])
181                                         if (put_reply == 0)
182
183         if (put_reply > 0)
184             if (put_reply < N)
185                 if (put_reply == my_id)
186                     if (put_reply == i)
187                         if (put_reply == j)
188                             if (put_reply == t)
189                                 if (put_reply == tt)
190                                     if (put_reply == A_slave[j])
191                                         if (put_reply == B_slave[t])
192                                         if (put_reply == C_slave[tt])
193                                         if (put_reply == CC[i][j])
194                                         if (put_reply == 0)
195
196         if (put_reply > 0)
197             if (put_reply < N)
198                 if (put_reply == my_id)
199                     if (put_reply == i)
200                         if (put_reply == j)
201                             if (put_reply == t)
202                                 if (put_reply == tt)
203                                     if (put_reply == A_slave[j])
204                                         if (put_reply == B_slave[t])
205                                         if (put_reply == C_slave[tt])
206                                         if (put_reply == CC[i][j])
207                                         if (put_reply == 0)
208
209         if (put_reply > 0)
210             if (put_reply < N)
211                 if (put_reply == my_id)
212                     if (put_reply == i)
213                         if (put_reply == j)
214                             if (put_reply == t)
215                                 if (put_reply == tt)
216                                     if (put_reply == A_slave[j])
217                                         if (put_reply == B_slave[t])
218                                         if (put_reply == C_slave[tt])
219                                         if (put_reply == CC[i][j])
220                                         if (put_reply == 0)
221
222         if (put_reply > 0)
223             if (put_reply < N)
224                 if (put_reply == my_id)
225                     if (put_reply == i)
226                         if (put_reply == j)
227                             if (put_reply == t)
228                                 if (put_reply == tt)
229                                     if (put_reply == A_slave[j])
230                                         if (put_reply == B_slave[t])
231                                         if (put_reply == C_slave[tt])
232                                         if (put_reply == CC[i][j])
233                                         if (put_reply == 0)
234
235         if (put_reply > 0)
236             if (put_reply < N)
237                 if (put_reply == my_id)
238                     if (put_reply == i)
239                         if (put_reply == j)
240                             if (put_reply == t)
241                                 if (put_reply == tt)
242                                     if (put_reply == A_slave[j])
243                                         if (put_reply == B_slave[t])
244                                         if (put_reply == C_slave[tt])
245                                         if (put_reply == CC[i][j])
246                                         if (put_reply == 0)
247
248         if (put_reply > 0)
249             if (put_reply < N)
250                 if (put_reply == my_id)
251                     if (put_reply == i)
252                         if (put_reply == j)
253                             if (put_reply == t)
254                                 if (put_reply == tt)
255                                     if (put_reply == A_slave[j])
256                                         if (put_reply == B_slave[t])
257                                         if (put_reply == C_slave[tt])
258                                         if (put_reply == CC[i][j])
259                                         if (put_reply == 0)
260
261         if (put_reply > 0)
262             if (put_reply < N)
263                 if (put_reply == my_id)
264                     if (put_reply == i)
265                         if (put_reply == j)
266                             if (put_reply == t)
267                                 if (put_reply == tt)
268                                     if (put_reply == A_slave[j])
269                                         if (put_reply == B_slave[t])
270                                         if (put_reply == C_slave[tt])
271                                         if (put_reply == CC[i][j])
272                                         if (put_reply == 0)
273
274         if (put_reply > 0)
275             if (put_reply < N)
276                 if (put_reply == my_id)
277                     if (put_reply == i)
278                         if (put_reply == j)
279                             if (put_reply == t)
280                                 if (put_reply == tt)
281                                     if (put_reply == A_slave[j])
282                                         if (put_reply == B_slave[t])
283                                         if (put_reply == C_slave[tt])
284                                         if (put_reply == CC[i][j])
285                                         if (put_reply == 0)
286
287         if (put_reply > 0)
288             if (put_reply < N)
289                 if (put_reply == my_id)
290                     if (put_reply == i)
291                         if (put_reply == j)
292                             if (put_reply == t)
293                                 if (put_reply == tt)
294                                     if (put_reply == A_slave[j])
295                                         if (put_reply == B_slave[t])
296                                         if (put_reply == C_slave[tt])
297                                         if (put_reply == CC[i][j])
298                                         if (put_reply == 0)
299
299 }
```

```
29  {
30      get_reply[i]=0;
31      t = i * Y0; //当前位置
32      athread_get(PE_MODE, &A[my_id][t], &A_slave[t], L0,
33                  &get_reply[i], 0, 0, 0);
34      athread_get(PE_MODE, &B[my_id][t], &B_slave[t], L0,
35                  &get_reply[i], 0, 0, 0);
36  }
37
38  for (i = 0; i < N; i++)//计算
39  {
40      t = i * Y0; //当前起始位置
41      while (get_reply[i] != 2);
42      for (j = 0; j < Y0; j++)
43      {
44          tt=t+j;
45          C_slave[tt] = A_slave[tt] + B_slave[tt];
46      }
47      athread_put(PE_MODE, &C[my_id][t], &C_slave[t], L0, &put_reply, 0, 0);
48  }
49 }
```

## 附录 B Darknet 的 Makefile

```

1 GPU=0
2 CUDNN=0
3 OPENCV=0
4 OPENMP=0
5 DEBUG=0
6
7 ARCH= -gencode arch=compute_30,code=sm_30 \
     -gencode arch=compute_35,code=sm_35 \
8     -gencode arch=compute_50,code=[sm_50,compute_50] \
     -gencode arch=compute_52,code=[sm_52,compute_52]
9
10 #      -gencode arch=compute_20,code=[sm_20,sm_21] \ This one is deprecated?
11
12 # This is what I use, uncomment if you know your arch and want to specify
13 # ARCH= -gencode arch=compute_52,code=compute_52
14
15 VPATH=../src/../../examples
16
17 SLIB=libdarknet.so
18 ALIB=libdarknet.a
19 EXEC=darknet
20 OJDIR=../obj/
21
22 CC=gcc
23 CPP=g++
24 NVCC=nvcc
25 AR=ar
26 ARFLAGS=rcs
27 OPTS=-Ofast
28 LDFLAGS= -lm -pthread
29 COMMON= -Iinclude/ -Isrc/
30 CFLAGS=-Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC
31
32 ifeq ($(OPENMP), 1)
33 CFLAGS+= -fopenmp
34 endif
35
36 ifeq ($(DEBUG), 1)
37 OPTS=-O0 -g
38 endif
39
40 CFLAGS+= $(OPTS)
41
42 ifeq ($(OPENCV), 1)
43 COMMON+= -DOPENCV
44 CFLAGS+= -DOPENCV

```

```

45 LDFLAGS+= `pkg-config --libs opencv` -lstdc++
46 COMMON+= `pkg-config --cflags opencv`
47 endif

49 ifeq ($(GPU), 1)
50 COMMON+= -DGPU -I/usr/local/cuda/include/
51 CFLAGS+= -DGPU
52 LDFLAGS+= -L/usr/local/cuda/lib64 -lcuda -lcudart -lcublas -lcurand
53 endif

55 ifeq ($(CUDNN), 1)
56 COMMON+= -DCUDNN
57 CFLAGS+= -DCUDNN
58 LDFLAGS+= -lcudnn
59 endif

61 OBJ=gemm.o utils.o cuda.o deconvolutional_layer.o convolutional_layer.o list.o image.o
       activations.o im2col.o col2im.o blas.o crop_layer.o dropout_layer.o maxpool_layer.o
       softmax_layer.o data.o matrix.o network.o connected_layer.o cost_layer.o parser.o
       option_list.o detection_layer.o route_layer.o upsample_layer.o box.o
       normalization_layer.o avgpool_layer.o layer.o local_layer.o shortcut_layer.o
       logistic_layer.o activation_layer.o rnn_layer.o gru_layer.o crnn_layer.o demo.o
       batchnorm_layer.o region_layer.o reorg_layer.o tree.o lstm_layer.o l2norm_layer.o
       yolo_layer.o iseg_layer.o image_opencv.o
EXECOBJA=captcha.o lsd.o super.o art.o tag.o cifar.o go.o rnn.o segmenter.o regressor.o
       classifier.o coco.o yolo.o detector.o nightmare.o instance-segmenter.o darknet.o

63 ifeq ($(GPU), 1)
64 LDFLAGS+= -lstdc++
65 OBJ+=convolutional_kernels.o deconvolutional_kernels.o activation_kernels.o im2col_kernels.o
       col2im_kernels.o blas_kernels.o crop_layer_kernels.o dropout_layer_kernels.o
       maxpool_layer_kernels.o avgpool_layer_kernels.o
endif

67 EXECOBJ = $(addprefix $(OBJDIR), $(EXECOBJA))
68 OBJS = $(addprefix $(OBJDIR), $(OBJ))
DEPS = $(wildcard src/*.h) Makefile include/darknet.h

71 all: obj backup results $(SLIB) $(ALIB) $(EXEC)
72 #all: obj results $(SLIB) $(ALIB) $(EXEC)

75 $(EXEC): $(EXECOBJ) $(ALIB)
76     $(CC) $(COMMON) $(CFLAGS) $^ -o $@ $(LDFLAGS) $(ALIB)

79 $(ALIB): $(OBJS)
80     $(AR) $(ARFLAGS) $@ $^

81 $(SLIB): $(OBJS)

```

```
83      $(CC) $(CFLAGS) -shared $^ -o $@ $(LDFLAGS)

85 $(OBJDIR)%.o: %.cpp $(DEPS)
86     $(CPP) $(COMMON) $(CFLAGS) -c $< -o $@

87 $(OBJDIR)%.o: %.c $(DEPS)
88     $(CC) $(COMMON) $(CFLAGS) -c $< -o $@

89 $(OBJDIR)%.o: %.cu $(DEPS)
90     $(NVCC) $(ARCH) $(COMMON) --compiler-options "$(CFLAGS)" -c $< -o $@

91
92 obj:
93     mkdir -p obj
94 backup:
95     mkdir -p backup
96 results:
97     mkdir -p results

98 .PHONY: clean

99 clean:
100    rm -rf $(OBJS) $(SLIB) $(ALIB) $(EXEC) $(EXECOBJ) $(OBJDIR)/*
```

## 附录 C 中国科学院大学学位论文撰写要求

学位论文是研究生科研工作成果的集中体现，是评判学位申请者学术水平、授予其学位的主要依据，是科研领域重要的文献资料。根据《科学技术报告、学位论文和学术论文的编写格式》(GB/T 7713-1987)、《学位论文编写规则》(GB/T 7713.1-2006) 和《文后参考文献著录规则》(GB7714—87) 等国家有关标准，结合中国科学院大学（以下简称“国科大”）的实际情况，特制订本规定。

### C.1 论文无附录者无需附录部分

#### C.2 测试公式编号 $\Lambda, \lambda, \theta, \bar{\Lambda}, \sqrt{S_{NN}}$

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0 \\ \frac{\partial(\rho \mathbf{V})}{\partial t} + \nabla \cdot (\rho \mathbf{V} \mathbf{V}) = \nabla \cdot \boldsymbol{\sigma} \\ \frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\rho E \mathbf{V}) = \nabla \cdot (k \nabla T) + \nabla \cdot (\boldsymbol{\sigma} \cdot \mathbf{V}) \end{cases} \dots (C.1)$$

$$\frac{\partial}{\partial t} \int_{\Omega} u \, d\Omega + \int_S \mathbf{n} \cdot (u \mathbf{V}) \, dS = \dot{\phi} \dots (C.2)$$

$$\mathcal{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, dt, \quad \mathcal{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, dt$$

$$\mathcal{F}(f(x + x_0)) = \mathcal{F}(f(x)) e^{2\pi i \xi x_0}, \quad \mathcal{F}(f(x + x_0)) = \mathcal{F}(f(x)) e^{2\pi i \xi x_0}$$

mathtext:  $A, F, L, 2, 3, 5, \sigma$ , mathnormal:  $A, F, L, 2, 3, 5, \sigma$ , mathrm:  $A, F, L, 2, 3, 5, \sigma$ .

mathbf:  $\mathbf{A}, \mathbf{F}, \mathbf{L}, \mathbf{2}, \mathbf{3}, \mathbf{5}, \boldsymbol{\sigma}$ , mathit:  $A, F, L, 2, 3, 5, \sigma$ , mathsf:  $A, F, L, 2, 3, 5, \sigma$ .

mathtt:  $A, F, L, 2, 3, 5, \sigma$ , mathfrak:  $\mathfrak{A}, \mathfrak{F}, \mathfrak{L}, 2, 3, 5, \sigma$ , mathbb:  $\mathbb{A}, \mathbb{F}, \mathbb{L}, 2, 3, 5, \sigma$ .

mathcal:  $\mathcal{A}, \mathcal{F}, \mathcal{L}, 2, 3, 5, \sigma$ , mathscr:  $\mathcal{A}, \mathcal{F}, \mathcal{L}, 2, 3, 5, \sigma$ , boldsymbol:  $\mathbf{A}, \mathbf{F}, \mathbf{L}, 2, 3, 5, \sigma$ .

vector:  $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$ , unitvector:  $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$

matrix:  $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$ , unitmatrix:  $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$

tensor:  $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$ , unittensor:  $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$

### C.3 测试生僻字



## 参考文献

- [1] 为何中国超算偏爱异构计算[J/OL]. 知乎专栏[2019-07-09]. <https://zhuanlan.zhihu.com/p/20908218>.
- [2] REDMON J. Darknet: Open source neural networks in c[EB/OL]. 2013–2016. <http://pjreddie.com/darknet/>.
- [3] REDMON J, FARHADI A. Yolov3: An incremental improvement[J]. arXiv, 2018.
- [4] Darknet 概述 - 搬砖笔记 - CSDN 博客[EB/OL]. [2019-07-14]. <https://blog.csdn.net/u010122972/article/details/83541978>.
- [5] LAMPORT L. Document preparation system[M]. Addison-Wesley Reading, MA, 1986.
- [6] 初景利. 图书馆数字参考咨询服务研究[M]. 北京: 北京图书馆出版社, 2004.
- [7] 陈浩元. 著录文后参考文献的规则及注意事项[J]. 编辑学报, 2005, 17(6):413-415.
- [8] WALLS S C, BARICHIVICH W J, BROWN M E. Drought, deluge and declines: the impact of precipitation extremes on amphibians in a changing climate[J/OL]. Biology, 2013, 2(1):399-418[2013-11-04]. <http://www.mdpi.com/2079-7737/2/1/399>. DOI: 10.3390/biology2010399.
- [9] BETTS L R, TAYLOR C P. Aging reduces center-surround antagonism in visual motion processing[J]. Neuron, 2005, 45(3):361-366.
- [10] 陈晋镳, 张惠民, 朱士兴, 等. 蓟县震旦亚界研究[M]//中国地质科学院天津地质矿产研究所. 中国震旦亚界. 天津: 天津科学技术出版社, 1980: 56-114.
- [11] BRAVO H, OLAVARRIA J. Comparative study of visual inter and intrahemispheric cortico-cortical connections in five native chilean rodents[J]. Anatomy and embryology, 1990, 181(1): 67-73.
- [12] 袁训来, 陈哲, 肖书海. 蓝田生物群: 一个认识多细胞生物起源和早期演化的新窗口 – 篇一[J]. 科学通报, 2012, 57(34):3219.
- [13] 袁训来, 陈哲, 肖书海. 蓝田生物群: 一个认识多细胞生物起源和早期演化的新窗口 – 篇二[J]. 科学通报, 2012, 57(34):3219.
- [14] 袁训来, 陈哲, 肖书海. 蓝田生物群: 一个认识多细胞生物起源和早期演化的新窗口 – 篇三[J]. 科学通报, 2012, 57(34):3219.
- [15] STAMERJOHANNS H, GINEV D, DAVID C, et al. MathML-aware article conversion from LaTeX[J]. Towards a Digital Mathematics Library, 2009, 16(2):109-120.
- [16] 哈里森·沃尔德伦. 经济数学与金融数学[M]. 谢远涛, 译. 北京: 中国人民大学出版社, 2012: 235-236.
- [17] 牛志明, 斯温兰德, 雷光春. 综合湿地管理国际研讨会论文集[C]. 北京: 海洋出版社, 2013.

- [18] ボハンデ. 過去及び現在に於ける英國と会[J]. 日本時報, 1928, 17:5-9.
- [19] ДубровинА. И. Открытое письмо Председателя Главного Совета Союза Русского Народа Санкт-Петербургскому Антонию, Первенствующему члену Священного Синода[J]. Вече, 1906:1-3.
- [20] WIKIBOOK. <http://en.wikibooks.org/wiki/latex>[M]. On-line Resources, 2014.

## 图形列表

1.1 超算系统登录流程 .....	2
1.2 第 1 章的知识点概括 .....	11
2.1 Q 判据等值面图，同时测试一下一个很长的标题，比如这真的是一个 很长很长很长很长很长很长很长很长的标题。 .....	19
2.2 激波圆柱作用。 .....	20
2.3 总声压级。(a) 这是子图说明信息，(b) 这是子图说明信息，(c) 这是子 图说明信息，(d) 这是子图说明信息。 .....	20

## 表格列表

2.1 这是一个样表。 .....	19
-------------------	----

## 作者简历及攻读学位期间发表的学术论文与研究成果

本科生无需此部分。

### 作者简历

#### casthesis 作者

吴凌云，福建省屏南县人，中国科学院数学与系统科学研究院博士研究生。

#### ucasthesis 作者

莫晃锐，湖南省湘潭县人，中国科学院力学研究所硕士研究生。

### 已发表 (或正式接受) 的学术论文:

[1] ucasthesis: A LaTeX Thesis Template for the University of Chinese Academy of Sciences, 2014.

### 申请或已获得的专利:

(无专利时此项不必列出)

### 参加的研究项目及获奖情况:

可以随意添加新的条目或是结构。

## 致 谢

感激 casthesis 作者吴凌云学长, gbt7714-bibtex-style 开发者 zepinglee, 和 ctex 众多开发者们。若没有他们的辛勤付出和非凡工作, L<sup>A</sup>T<sub>E</sub>X 菜鸟的我是无法完成此国科大学位论文 L<sup>A</sup>T<sub>E</sub>X 模板 ucasthesis 的。在 L<sup>A</sup>T<sub>E</sub>X 中的一点一滴的成长源于开源社区的众多优秀资料和教程, 在此对所有 L<sup>A</sup>T<sub>E</sub>X 社区的贡献者表示感谢!

ucasthesis 国科大学位论文 L<sup>A</sup>T<sub>E</sub>X 模板的最终成型离不开以霍明虹老师和丁云云老师为代表的国科大学位办公室老师们制定的官方指导文件和众多 ucasthesis 用户的热心测试和耐心反馈, 在此对他们的认真付出表示感谢。特别对国科大的赵永明同学的众多有效反馈意见和建议表示感谢, 对国科大本科部的陆晴老师和本科部学位办的丁云云老师的细致审核和建议表示感谢。谢谢大家的共同努力和支持, 让 ucasthesis 为国科大学子使用 L<sup>A</sup>T<sub>E</sub>X 撰写学位论文提供便利和高效这一目标成为可能。