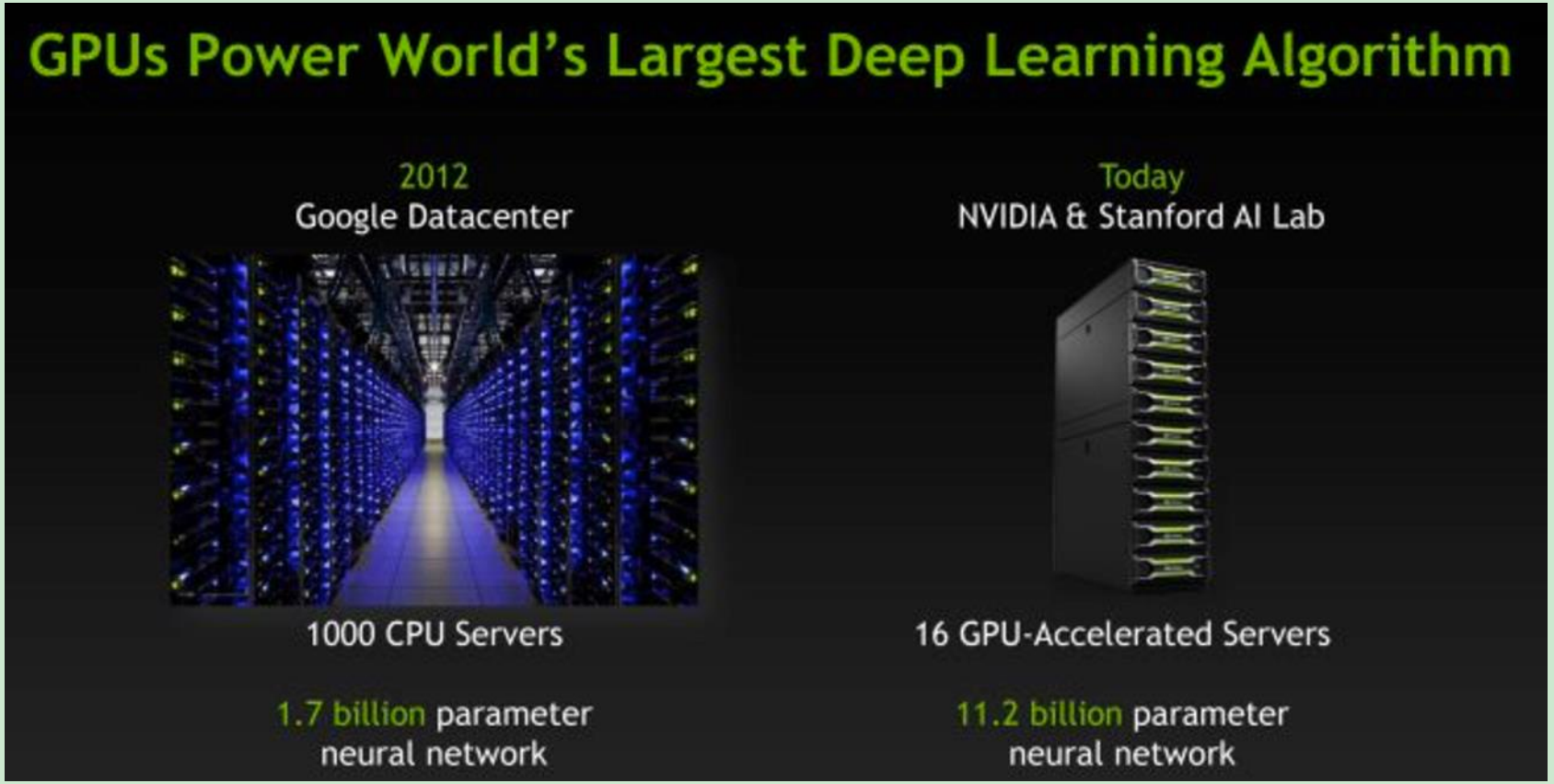


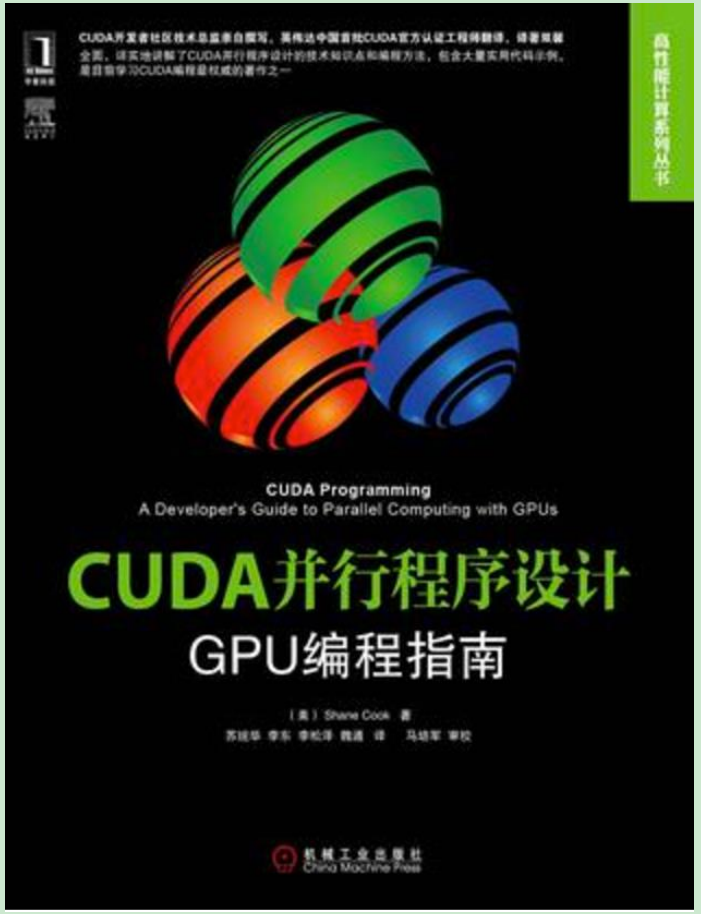
GPU并行计算与CUDA编程 第1课

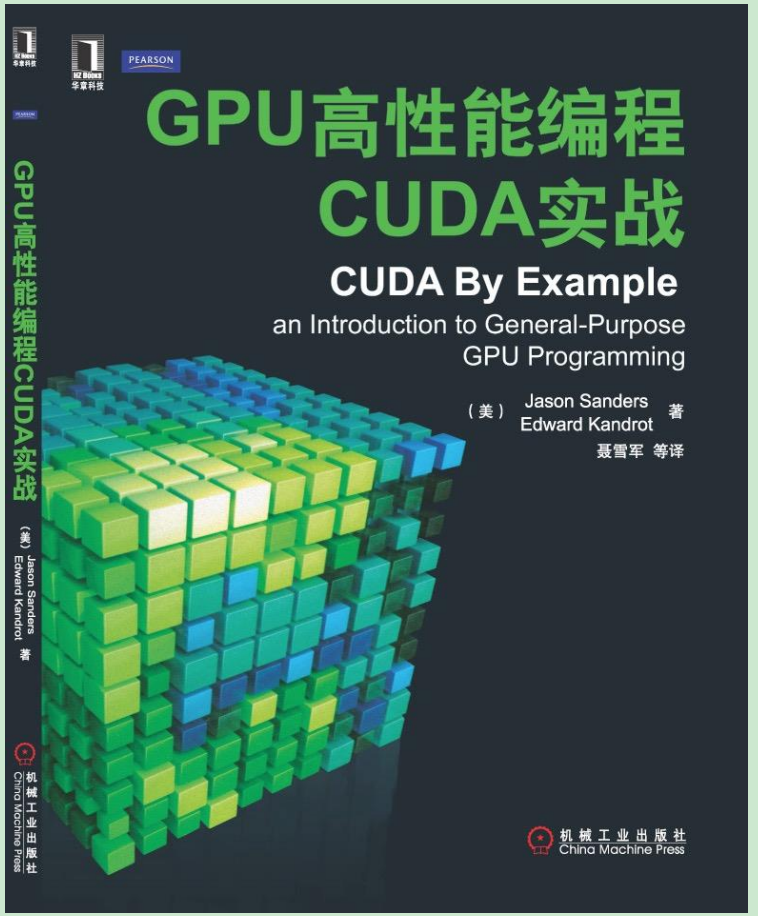
第一版 讲师 罗韵（WeChat：LaurenLuoYun）

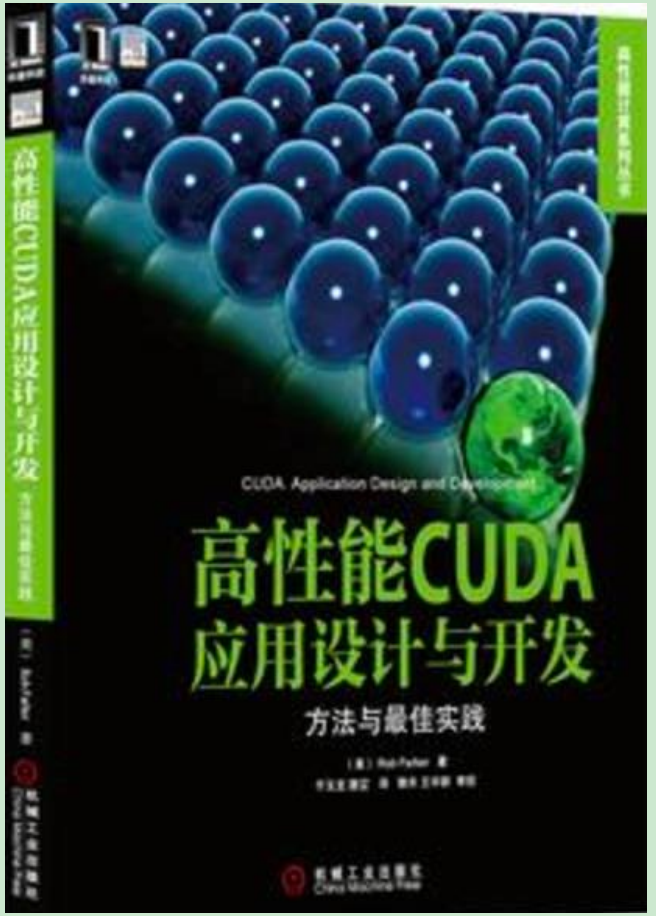
本周介绍内容

- 0. 课程参考资料
- 1. GPU并行计算的原理与意义
- 2. CUDA硬件环境，体系结构，常见的显卡型号与性能，显卡的选择与显存需求估计
- 3. CUDA软件环境介绍，包括平台、架构、开发工具和热点技术
- 4. 租用AWS云服务的环境搭建步骤
- 5. 本地机器的环境搭建步骤

0.课程参考资料


CUDA Programming
A Developer's Guide to Parallel Computing with GPUs
CUDA并行程序设计
GPU编程指南


GPU高性能编程
CUDA实战
CUDA By Example
an Introduction to General-Purpose
GPU Programming
(美) Jason Sanders
Edward Kandrot 著
聂雪军 等译
机械工业出版社
China Machine Press


高性能CUDA
应用设计与开发
方法与最佳实践

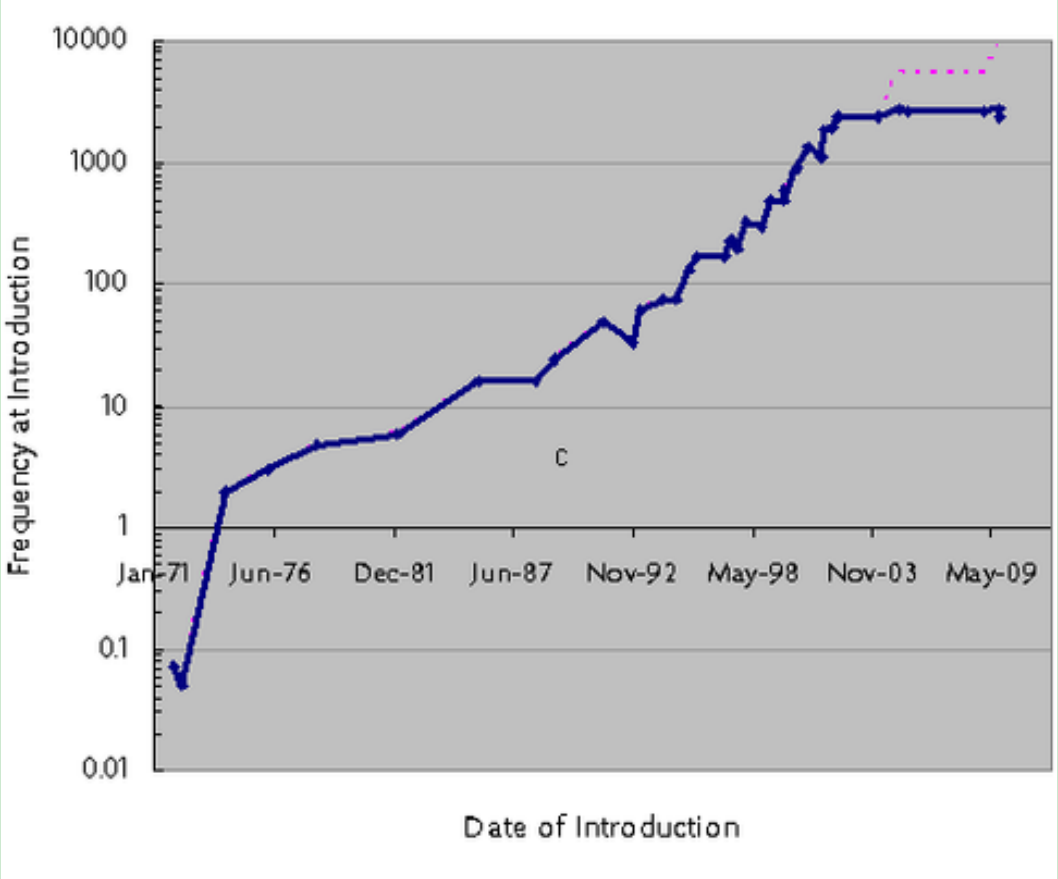
1. GPU并行计算的原理与意义

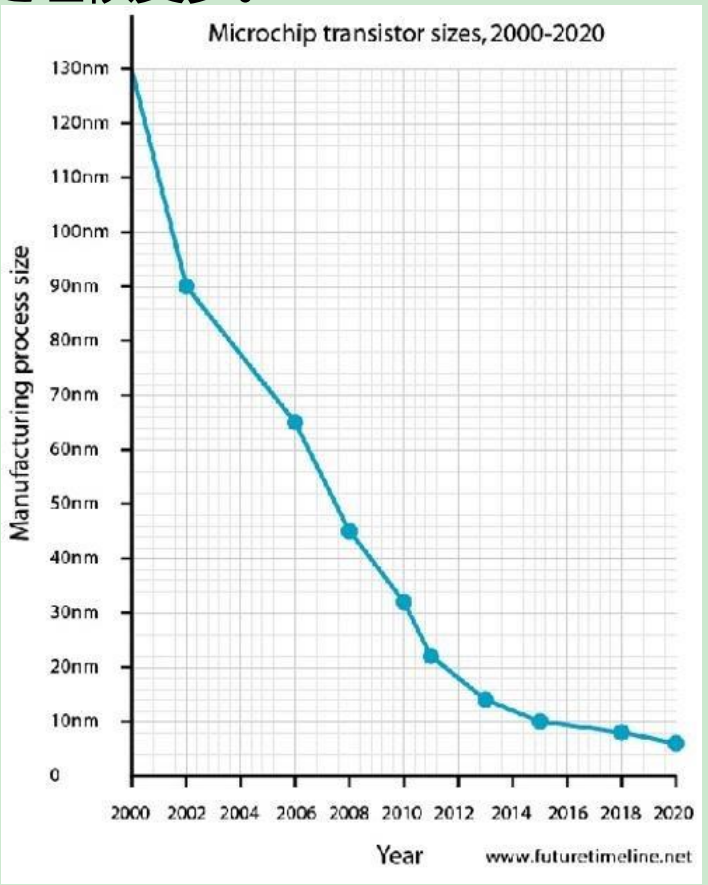
- **CPU和GPU的区别**

面积代表了实际的硬件占用面积

- 图片来自NVIDIA CUDA文档。其中绿色的是计算单元，橙红色的是存储单元，橙黄色的是控制单元。
- GPU采用了数量众多的计算单元和超长的流水线，但只有非常简单的控制逻辑并省去了Cache。而CPU不仅被Cache占据了大量空间，而且还有有复杂的控制逻辑和诸多优化电路，相比之下计算能力只是CPU很小的一部分

- **CPU的发展：**处理器越来越小，处理速度越来越快，处理核变多。



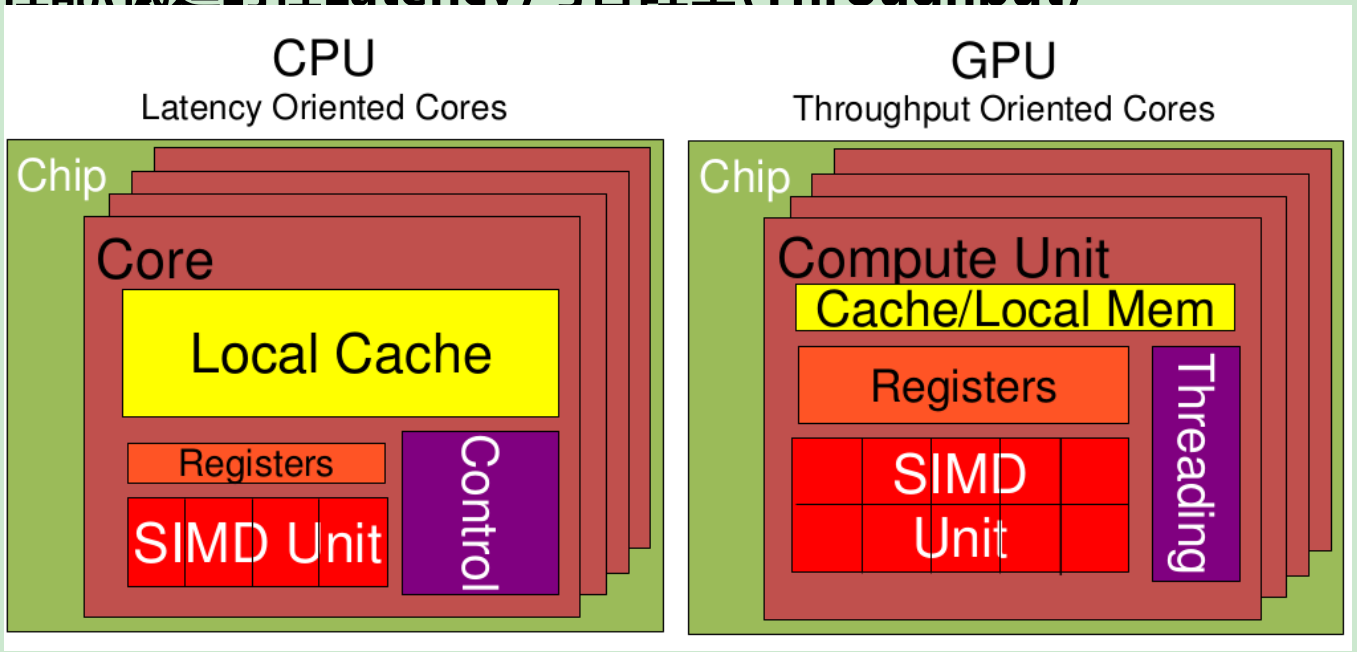


说明了就是cpu处理器的大小是可预知的，因此很难增强了cpu的能力了。

- **为什么CPU不可以一直沿着趋势发展下去？**

总结就是，gpu是低延时的，但是总共的吞吐量太小了。
gpu的每个处理单元是高延时的，但是数目众多，这样加和之后速度就快了。

- **性能(低延时性Latency)与吞吐量(Throughput)**



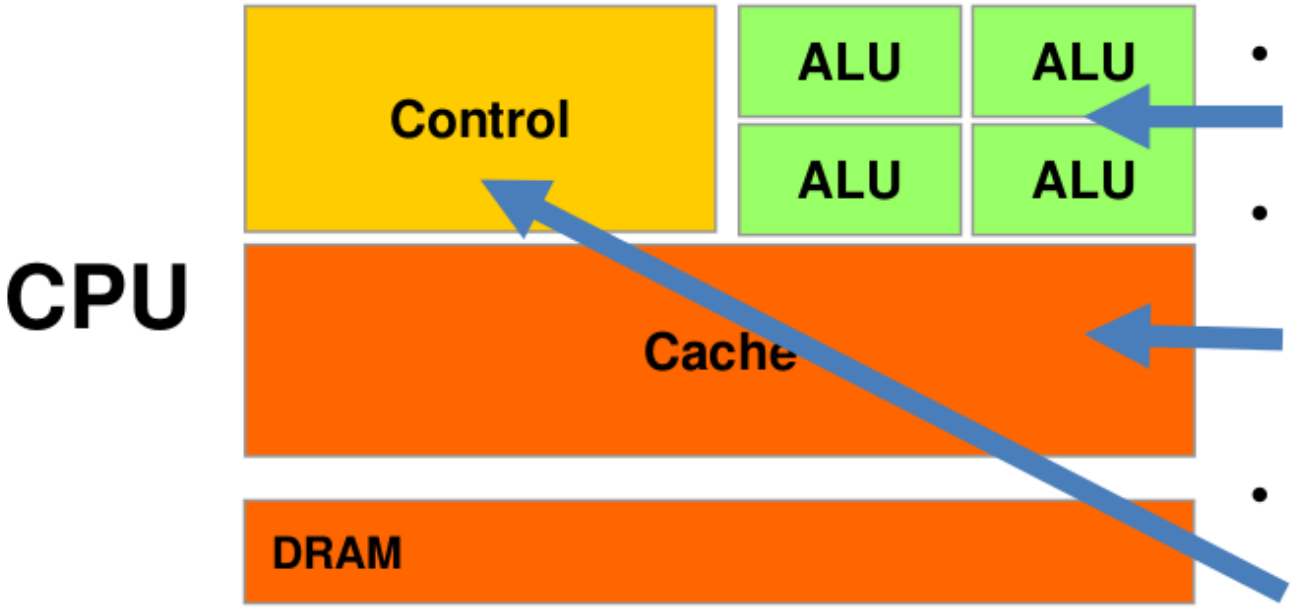
举一个挖掘机的例子：
cpu有一个大的挖掘机，但是每次挖掘的数量是有限的。
而gpu有很多小的挖掘机，尽管每个挖掘机的能力比不上cpu的大挖掘机，但是综合所有的小挖掘机，能力还是比大挖掘机强的。

这就说明了gpu的吞吐量大。

- Cache, local memory： CPU > GPU
- Threads(线程数): GPU > CPU
- Registers: GPU > CPU 多寄存器可以支持非常多的Thread,thread需要用到register,thread数目大，register也必须得跟着很大才行。

CPU：基于低延时性设计

CPUs: Latency Oriented Desigr

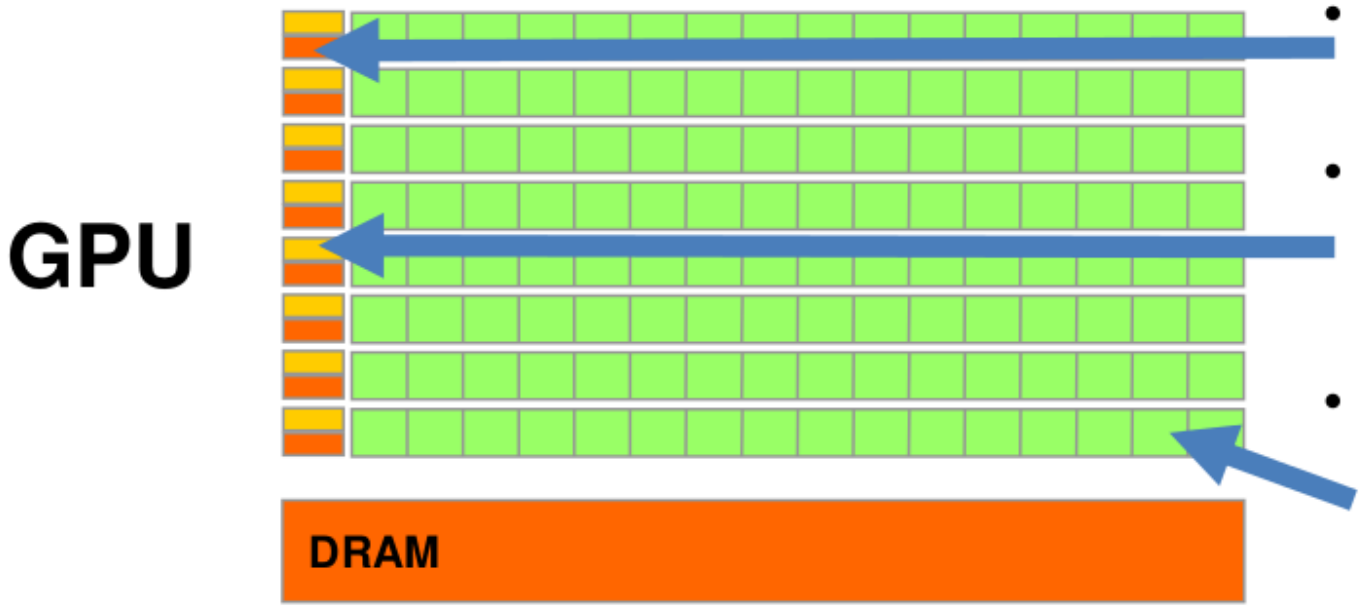


- Powerful ALU
  - Reduced operation latency
- Large caches
  - Convert long latency memory accesses to short latency cache accesses
- Sophisticated control
  - Branch prediction for reduced branch latency
  - Data forwarding for reduced data latency

- **ALU**：CPU有强大的ALU（算术运算单元）,它可以在很少的时钟周期内完成算术计算。
  - 当今的CPU可以达到64bit 双精度。执行双精度浮点源算的加法和乘法只需要1～3个时钟周期。
  - CPU的时钟周期的频率是非常高的，达到1.532～3gigahertz(千兆HZ, 10的9次方).
- **Cache**：大的缓存也可以降低延时。保存很多的数据放在缓存里面，当需要访问的这些数据，只要在之前访问过的，如今直接在缓存里面取即可。 硬盘>内存>缓存
- **Control**：复杂的逻辑控制单元。
  - 当程序含有多个分支的时候，它通过提供分支预测的能力来降低延时。
  - 数据转发。 当一些指令依赖前面的指令结果时，数据转发的逻辑控制单元决定这些指令在pipeline中的位置并且尽可能快的转发一个指令的结果给后续的指令。这些动作需要很多的对比电路单元和转发电路单元。

GPU：基于吞吐量设计

GPUs: Throughput Oriented Design



- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding
- Energy efficient ALUs
  - Many, long latency but heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies

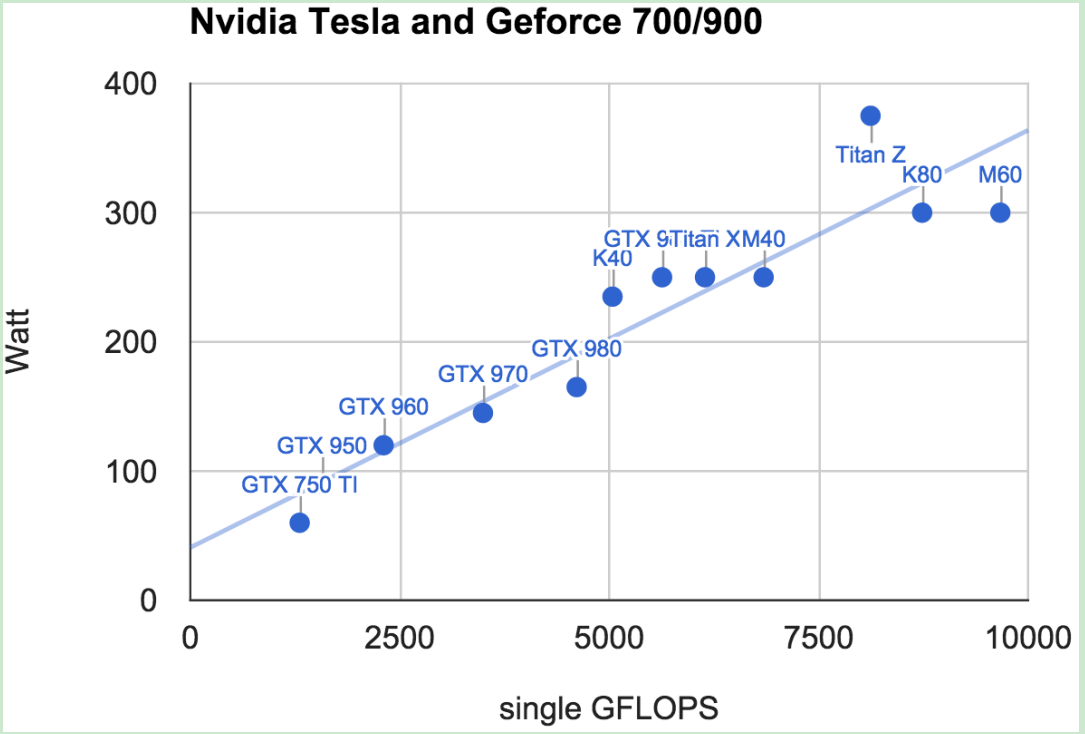
- **ALU，Cache：**GPU的特点是有很多的ALU和很少的cache. 缓存的目的不是保存后面需要访问的数据的，这点和CPU不同，而是为thread提高服务的。如果有很多线程需要访问同一个相同的数据，缓存会合并这些访问，然后再去访问dram（因为需要访问的数据保存在dram中而不是cache里面），获取数据后cache会转发这个数据给对应的线程，这个时候是数据转发的角色。但是由于需要访问dram，自然会带来延时的问题。
- **Control：**控制单元（左边黄色区域块）可以把多个的访问合并成少的访问。
- GPU的虽然有dram延时，却有非常多的ALU和非常多的thread. 为了平衡内存延时的问题，我们可以中充分利用多的ALU的特性达到一个非常大的吞吐量的效果。尽可能多的分配多的Threads.通常来看GPU ALU会有非常重的pipeline就是因为这样。
- CPU擅长逻辑控制，串行的运算。和通用类型数据运算不同，GPU擅长的是大规模并发计算，这也正是密码破解等所需要的。所以GPU除了图像处理，也越来越多的参与到计算当中来。

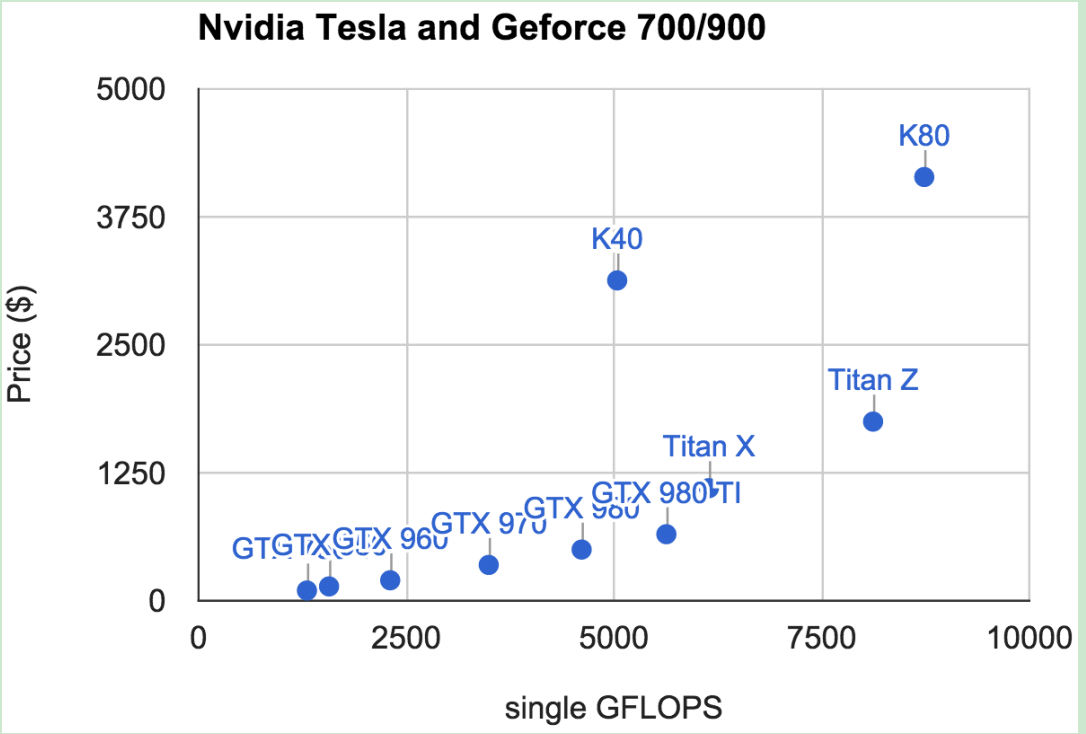
- **什么类型的程序适合在GPU上运行？**
- 1. 计算密集型的程序。
- 2. 易于并行的程序。GPU其实是一种SIMD(Single Instruction Multiple Data)架构。

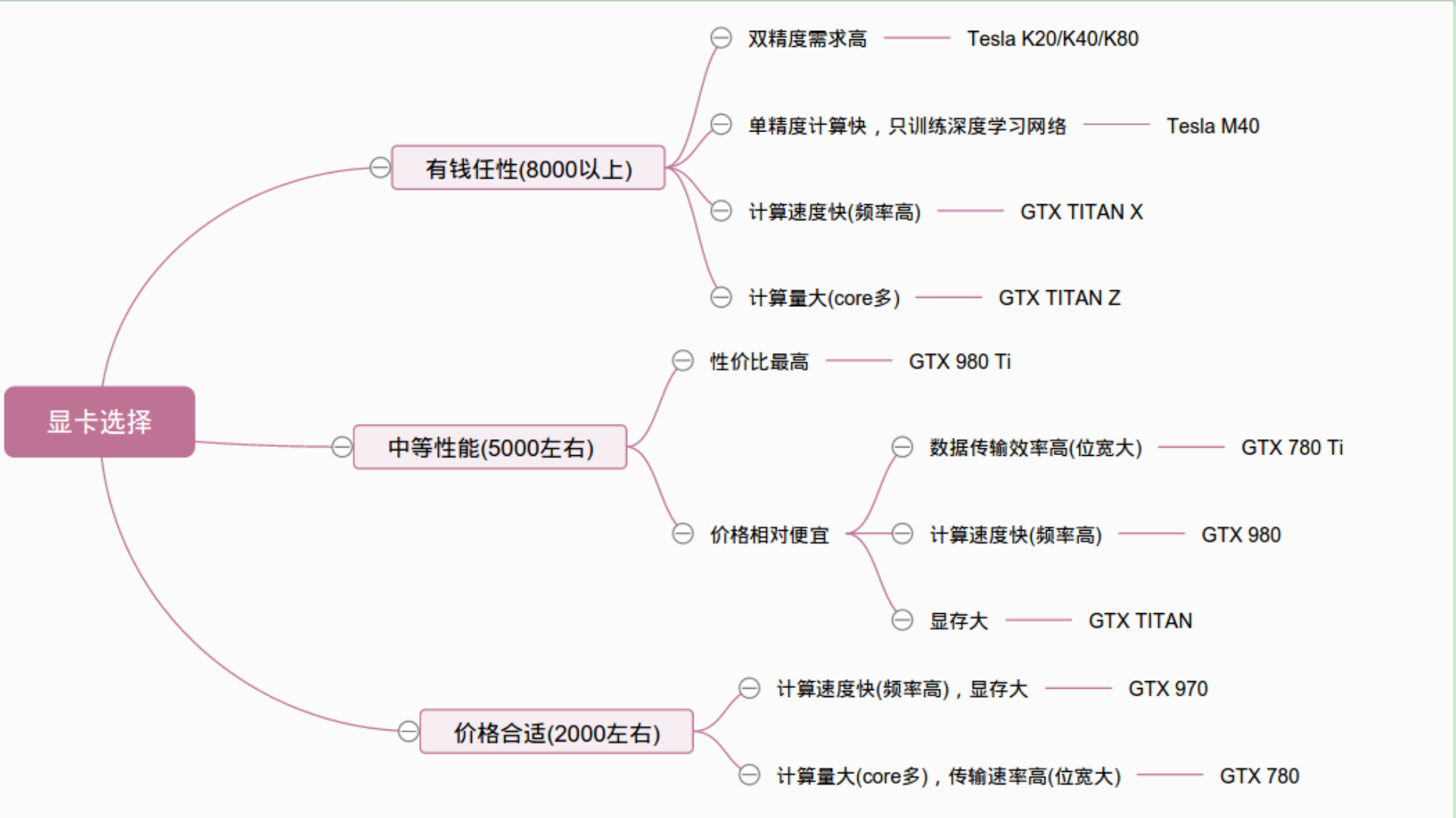
2.CUDA硬件环境

- **一些常用的名词：**
- gpu架构：Tesla、Fermi、Kepler、Maxwell、Pascal
- 芯片型号：GT200、GK210、GM104、GF104等
- 显卡系列：GeForce、Quadro、Tesla
- GeForce显卡型号：G/GS、GT、GTS、GTX

- 处理器：


Nvidia Tesla and Geforce 700/900
Watt
400
300
200
100
0
0
2500
5000
7500
10000
single GFLOPS
Titan Z
K80
M60
K40
GTX 980
GTX 970
GTX 960
GTX 950
GTX 750 TI


Nvidia Tesla and Geforce 700/900
Price ($)
5000
3750
2500
1250
0
0
2500
5000
7500
10000
single GFLOPS
K80
K40
Titan Z
Titan X
GTX 980 TI
GTX 960


显卡选择
有钱任性(8000以上)
双精度需求高
Tesla K20/K40/K80
单精度计算快，只训练深度学习网络
Tesla M40
计算速度快(频率高)
GTX TITAN X
计算量大(core多)
GTX TITAN Z
中等性能(5000左右)
性价比最高
GTX 980 Ti
价格相对便宜
数据传输效率高(位宽大)
GTX 780 Ti
计算速度快(频率高)
GTX 980
显存大
GTX TITAN
价格合适(2000左右)
计算速度快(频率高)，显存大
GTX 970
计算量大(core多)，传输速率高(位宽大)
GTX 780

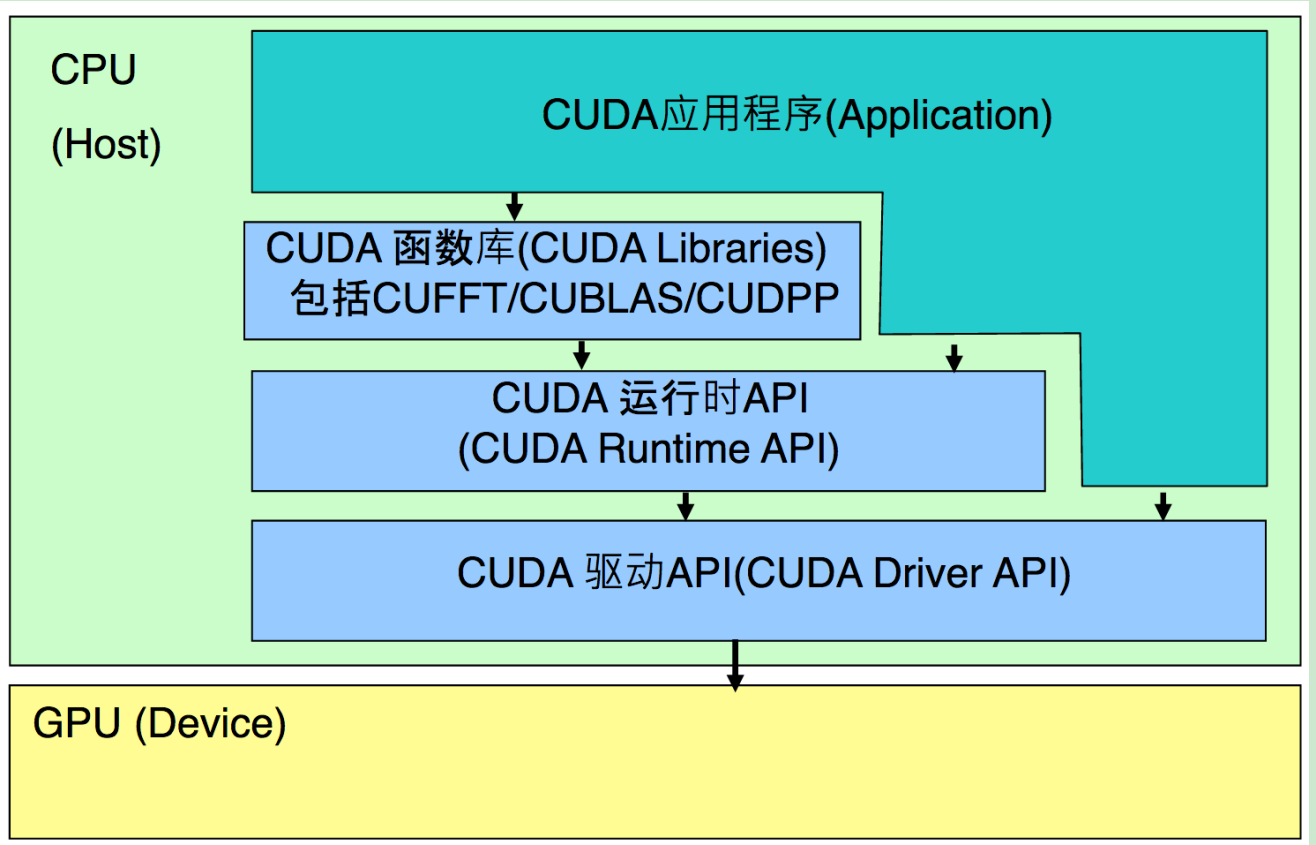
- **内存评估：（以卷积神经网络的内存需求评估为例）**
- 1. 激活和误差
- 2. 计算的规模
- 3. 输入维度
- 4. 分类数量

- **多GPU的集群**
- 并行难度大，特别是对深度学习，速度提升不明显。
- 参考一个国外的案例：http://timdettmers.com/2014/08/14/which-gpu-for-deep-learning/

3. CUDA软件环境

- **CUDA是什么？**
- CUDA，全称是Compute Unified Device Architecture，英伟达在2007年推出这个统一计算架构，为了让gpu有可用的编程环境，从而能通过程序控制底层的硬件进行计算。CUDA提供host-device的编程模式以及非常多的接口函数和科学计算库，通过同时执行大量的线程而达到并行的目的。CUDA也有不同的版本，从1.0开始到现在的8.0，每个版本都会有一些新特性。CUDA是基于C语言的扩展，例如扩展了一些限定符device、shared等，从3.0开始也支持c++编程，从7.0开始支持c++11。

- CUDA软件体系可以分为三层结构
  - CUDA函数库（CUDA Library）
  - CUDA运行时API（Runtime API）
  - CUDA驱动API（Driver API）



- CUDA应用程序可以通过直接调用底层的CUDA驱动来调用GPU硬件进行并行计算
- 也可以使用对底层驱动进行了一定封装的CUDA运行时库来简化编程过程（二者不可混合使用）
- 对一些标准的数学函数计算，也可以直接调用CUFFT、CUBLAS以及CUDPP等标准函数库进一步简化编程过程。

CUDA API

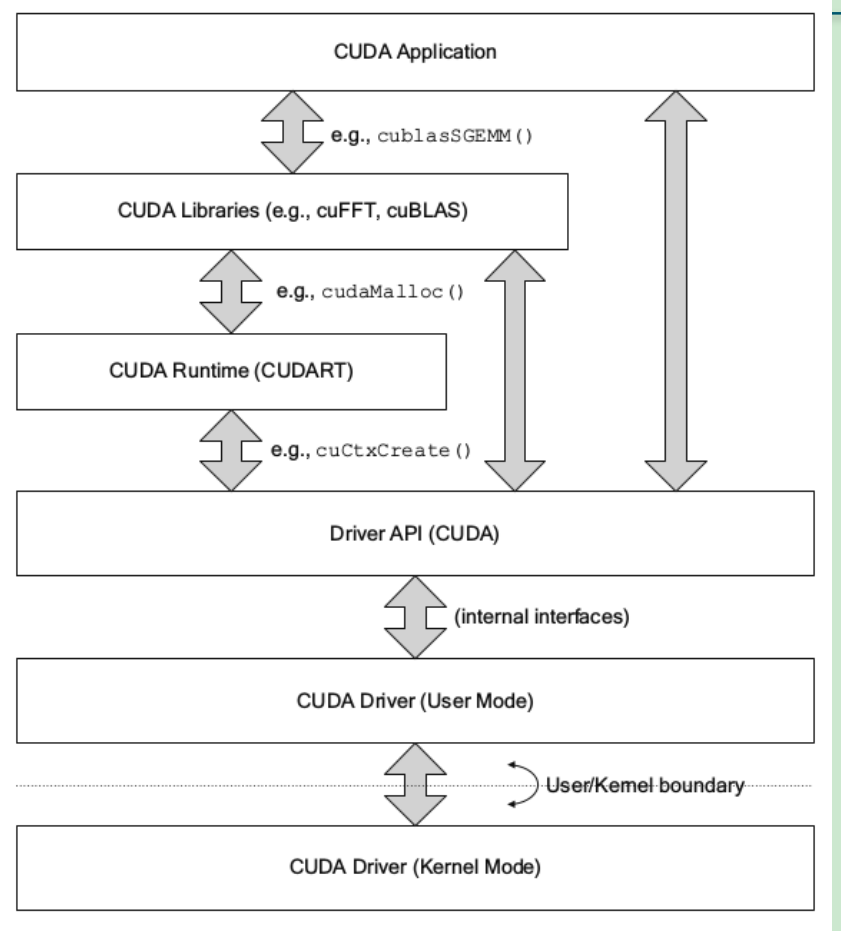
- CUDA可以通过两种方式调用底层GPU硬件：驱动API和运行API

驱动API

- 驱动API是一种基于句柄的底层接口,大多数对象通过句 柄被引用,其函数前缀均为cu。
- 在调用驱动API前必须进行初始化,然后创建一个CUDA上下文,该上下文关联到特定设备并成为主机线程的当前上下文。
- 通过加载PTX汇编形式或二进制对象形式的内核并指定 参数,就可以启动内核进行计算。
- CUDA驱动API可以通过直接操作硬件执行一些复杂的功 能而获得更高的性能,但其编程较为复杂,使用难度较 大。

运行API

- 运行时API对驱动API进行了一定的封装,隐藏了其部分 实现细节,因此使用起来更为方便,简化了编程的过程 ,因此我们实际中更多使用的是运行时API。
- 运行时API没有专门的初始化函数,它将在第一次调用运 行时函数时自动完成初始化。
- 使用运行时API的时候,通常需要包含头文件cuda_runtime.h,其函数前缀均为cuda。



CUDA软件层(图片来源《The cuda handbook》)

CUDA函数库

CUDA提供了几个较为成熟的高效函数库,程序员可以 直接调用这些库函数进行计算,因而大大简化了程序员 的工作量。其中最常用的包括:

- CUFFT （利用CUDA进行傅里叶变换的函数库 ）
- CUBLAS （利用CUDA进行加速版本的完 整标准矩阵与向量的运算库 ）
- CUDPP （常用的并行操作函数库）
- CUDNN （利用CUDA进行深度卷积神经网络，深度学习常用）
- …

- 全部库：https://developer.nvidia.com/gpu-accelerated-libraries

- **NVCC编译流程**
- 由于程序是要经过编译器编程成可执行的二进制文件，而cuda程序有两种代码，一种是运行在cpu上的host代码，一种是运行在gpu上的device代码，所以NVCC编译器要保证两部分代码能够编译成二进制文件在不同的机器上执行。

文件后缀	意义
.cu	cuda源文件，包括host和device代码
.cup	经过预处理的cuda源文件，编译选项--preprocess/-E
.c	c源文件
.cc/.cxx/.cpp	c++源文件
.gpu	gpu中间文件，编译选项--gpu
.ptx	类似汇编代码，编译选项--ptx
.o/.obj	目标文件，编译选项--compile/-c
.a/.lib	库文件，编译选项--lib/-lib
.res	资源文件
.so	共享目标文件，编译选项--shared/-shared
.cubin	cuda的二进制文件，编译选项-cubin

3. AWS上安装CUDA环境

1. 创建AWS的GPU服务器

- 1. 注册并登陆AWS网站 https://aws.amazon.com/
- 2. 创建一台虚拟服务器（EC2 instance），建议选择操作系统Ubuntu14.04 64bit。
- 3. 选择服务器的类型，选择带GPU的服务器即可

☑	GPU instances	g2.2xlarge	8	15	1 x 60 (SSD)	Yes	High	-
☐	GPU instances	g2.8xlarge	32	60	2 x 120 (SSD)	-	10 Gigabit	-
☐	GPU compute	p2.xlarge	4	61	EBS only	Yes	High	-
☐	GPU compute	p2.8xlarge	32	488	EBS only	Yes	10 Gigabit	-
☐	GPU compute	p2.16xlarge	64	732	EBS only	Yes	20 Gigabit	-

- 4. 根据自己的需要选择是否需要使用spot instance
- 5. 通过使用Key Pair（AWS控制台可以设置），ssh登陆服务器，

  ssh的命令为：ssh -i [你下载的key pair] ubuntu@[服务器的IPv4 address]

2.安装NVIDIA驱动

- 1. 下载NVIDIA驱动

  寻找自己显卡的驱动：http://www.nvidia.com/Download/Find.aspx

  AWS使用的显卡：

g2.2xlarge : G2 Instances

Product Type	GRID
Product Series	GRID Series
Product	GRID K520
Operating System	Linux 64-bit
Recommended/Beta	Recommended/Certified

g2.8xlarge : CG1 Instances

Product Type	Tesla
Product Series	M-Class
Product	M2050
Operating System	Linux 64-bit
Recommended/Beta	Recommended/Certified

  GRID K520 的驱动：

  http://us.download.nvidia.com/XFree86/Linux-x86_64/340.98/NVIDIA-Linux-x86_64-340.98.run

  赋权限：chmod +x NVIDIA-Linux-x86_64-340.98.run

- 2. 安装基础环境

  sudo apt-get update && sudo apt-get install build-essential

  sudo apt-get install linux-image-extra-virtual

- 3. 取消nouveau（会和NVIDIA kernel module冲突）

  sudo vim /etc/modprobe.d/blacklist-nouveau.conf

  输入以下信息，保存退出：

  blacklist nouveau

  blacklist lbm-nouveau

  options nouveau modeset=0

  alias nouveau off

  alias lbm-nouveau off

  - 执行生效：echo options nouveau modeset=0 | sudo tee -a /etc/modprobe.d/nouveau-kms.conf

- 4. 重启

  sudo update-initramfs -u

  sudo reboot

- 5. 获取Kernel Source

  sudo apt-get install linux-source

  sudo apt-get install linux-headers-3.13.0-37-generic

- 6. 运行驱动

  sudo apt-get install linux-headers-`uname -r`

  sudo ./NVIDIA-Linux-x86_64-340.98.run

- 7. 加载驱动

  sudo modprobe nvidia

3. 安装CUDA8

- 1. 下载CUDA8的deb安装包
  - 下载地址：https://developer.nvidia.com/compute/cuda/8.0/prod/local_installers/cuda-repo-ubuntu1404-8-0-local_8.0.44-1_amd64-deb
- 2. 安装CUDA8
  - sudo dpkg -i cuda-repo-ubuntu1404-8-0-local_8.0.44-1_amd64-deb
  - sudo apt-get update
  - sudo apt-get install cuda

- 3.设置环境变量
  - 在终端输入：

    export PATH=/usr/local/cuda-8.0/bin:$PATH

    export LD_LIBRARY_PATH=/usr/local/cuda-8.0/lib64:$LD_LIBRARY_PATH
  - sudo vim /etc/profile
  - 修改环境变量，输入以上export的两句，保存，退出
  - 使环境立即生效

    sudo ldconfig

- 4. 测试安装
  - cd /usr/local/cuda/samples/1_Utilities/deviceQuery
  - sudo make
  - ./deviceQuery

4. 本地安装CUDA

1. 安装NVIDIA驱动

- 1. 选择对应的驱动下载：http://www.nvidia.com/Download/Find.aspx
- 2. 执行命令：sudo service lightdm stop //关闭图形界面
- 3. 安装驱动：
  - 赋权限：sudo chmod 755 [刚下载下来的驱动]
  - 安装: sudo [驱动]
- 4.安装完成后重新启动图形界面：sudo service lightdm start

- 安装后可能遇到的问题：循环登陆界面，无法进入登陆系统。
- 解决方案：
- 通过ssh登陆系统（建议安装前都先安装ssh,以防图形界面无法进入）
- 执行以下命令：
- sudo add-apt-repository ppa:bumblebee/stable
- sudo apt-get update
- sudo apt-get install bumblebee bumblebee-nvidia
- sudo　reboot //重启

2.安装CUDA8

- 1. 下载CUDA8 ： https://developer.nvidia.com/cuda-release-candidate-download
  （需要注册，然后选择自己的系统对应的CUDA，可以选择.run文件或者deb文件，这个不影响）
- 2. 终端运行：sudo sh [下载的.run文件]
- 3. 按照以下的选择即可：

```
Do you accept the previously read EULA?
accept/decline/quit: accept

Install NVIDIA Accelerated Graphics Driver for Linux-x86_64 361.62?
(y)es/(n)o/(q)uit: n

Install the CUDA 8.0 Toolkit?
(y)es/(n)o/(q)uit: y

Enter Toolkit Location
[ default is /usr/local/cuda-8.0 ]:

Do you want to install a symbolic link at /usr/local/cuda?
(y)es/(n)o/(q)uit: y

Install the CUDA 8.0 Samples?
(y)es/(n)o/(q)uit: y

Enter CUDA Samples Location
[ default is /home/zhou ]:
```

- 3. 按照以下的选择即可：(直接回车是选择默认的)
  - Do you accept the previously read EULA?
  - accept/decline/quit: accept
  - Install NVIDIA Accelerated Graphics Driver for Linux-x86_64 361.62?
    (y)es/(n)o/(q)uit: n
  - Install the CUDA 8.0 Toolkit?
    (y)es/(n)o/(q)uit: y
  - Enter Toolkit Location
    [default is /usr/local/cuda-8.0]:
  - Do you want to install a symbolic link at /usr/local/cuda?
    (y)es/(n)o/(q)uit: y
  - Install the CUDA 8.0 Samples?
    (y)es/(n)o/(q)uit: y
  - Enter CUDA Samples Location
    [default is /home/luoyun]:

- 可能遇到的错误：

```
Driver: Not Selected

Toolkit: Installed in /usr/local/cuda-8.0

Samples: Installed in /home/zhou, but missing recommended libraries

Please make sure that

- PATH includes /usr/local/cuda-8.0/bin

- LD_LIBRARY_PATH includes /usr/local/cuda-8.0/lib64, or, add /usr/local/cuda-8.0/lib64 to /etc/ld.so.conf and run ldconf

To uninstall the CUDA Toolkit, run the uninstall script in /usr/local/cuda-8.0/bin

Please see CUDA_Installation_Guide_Linux.pdf in /usr/local/cuda-8.0/doc/pdf for detailed information on setting up CUDA.

***WARNING: Incomplete installation! This installation did not install the CUDA Driver. A driver of version at least 361.

To install the driver using this installer, run the following command, replacing with the name of this run file:

sudo .run -silent -driver

Logfile is /tmp/cuda_install_2961.log
```

- 解决方案：
- 缺少了一些库，安装上：
- sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev libxi-dev libglu1-mesa libglu1-mesa-dev

- 4. CUDA安装是否成功的检测方法和AWS的一样
- 测试安装
  - cd /usr/local/cuda/samples/1_Utilities/deviceQuery
  - sudo make
  - ./deviceQuery

本周作业

第一版 讲师 罗韵（WeChat：LaurenLuoYun）

- 安装好CUDA环境，并截图最后安装的画面即可。

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

第一版 讲师 罗韵（WeChat：LaurenLuoYun）

Thanks

FAQ时间