Let’s get started!

Filippo Spiga, HPCS, University of Cambridge

<fs395@cam.ac.uk>

Outline

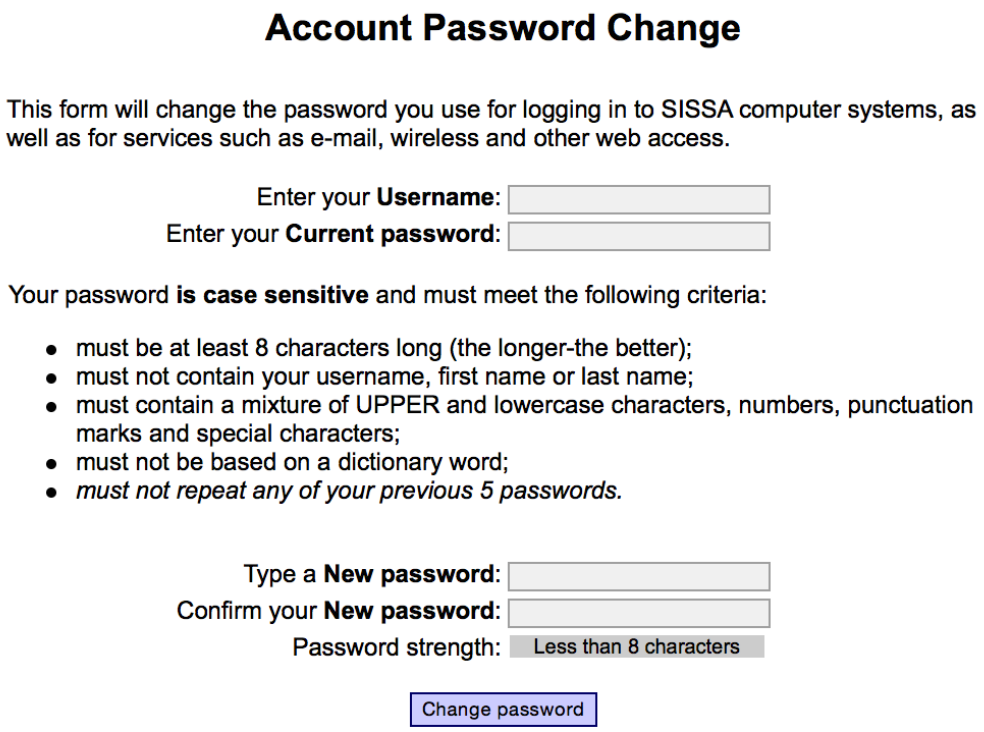
- SISSA HPC
- HPC environment (file-systems, scheduler, modules)
- How to compile an application
- GPU, hardware and software
- How to compile a GPU application
- Standard HPC workflow in a multi-tenant environment

… all mixed by live examples and few hands-on!

SISSA HPC credentials…

First of all, good practice to change default password!!!

https://password.sissa.it/



Account Password Change

This form will change the password you use for logging in to SISSA computer systems, as well as for services such as e-mail, wireless and other web access.

Enter your **Username**:

Enter your **Current password**:

Your password **is case sensitive** and must meet the following criteria:

- must be at least 8 characters long (the longer-the better);
- must not contain your username, first name or last name;
- must contain a mixture of UPPER and lowercase characters, numbers, punctuation marks and special characters;
- must not be based on a dictionary word;
- *must not repeat any of your previous 5 passwords.*

Type a **New password**:

Confirm your **New password**:

Password strength: Less than 8 characters

Change password

Connecting to SISSA HPC

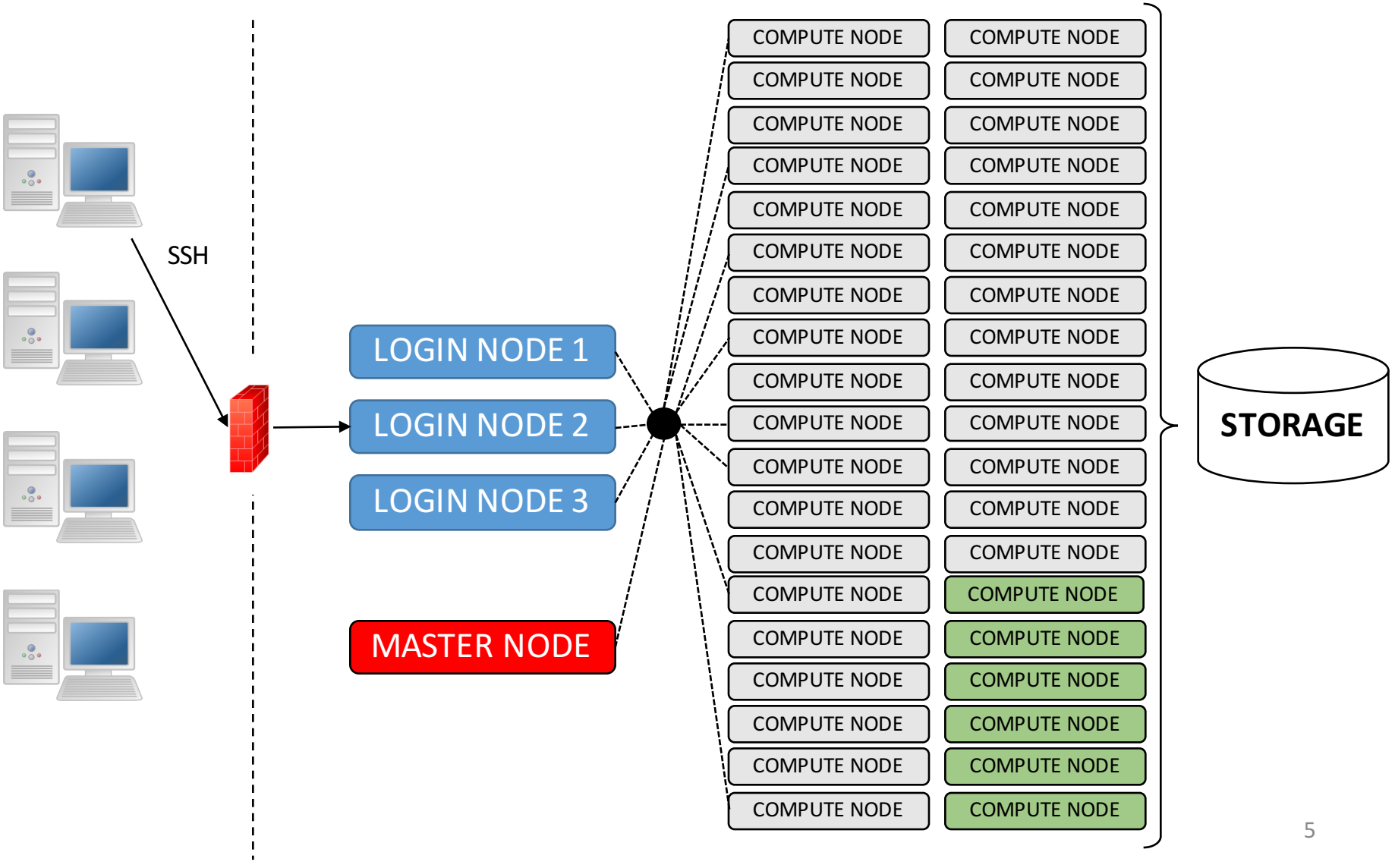
UNIX environment, interacting via Linux shell…

```
$ ssh id004914@frontend2.hpc.sissa.it
id004914@frontend2.hpc.sissa.it's password:
```

```
$ ssh <...>@frontend2.hpc.sissa.it
<...>@frontend2.hpc.sissa.it's password:
```

```
$ ssh -Y <...>@frontend2.hpc.sissa.it
<...>@frontend2.hpc.sissa.it's password:
```

(simplified) HPC cluster



SISSA HPC system

Relevant details…

- You have a HOME directory: /home/<your-login>
- You have a SCRATCH directory: /scratch/<your-login>
- There are LOGIN nodes
- There are COMPUTE nodes of various type
- There are software packages installed “somewhere”
- There is a SCHEDULER who help coordinate resources

All HPC systems have these components but in different HW/SW configurations or locations!

The Scheduler: TORQUE

- TORQUE is a open-source workload management system for Linux clusters derivate from PBS
- It manages *resources* (e.g. compute nodes), *queues* (a group of resources with attribute) and *work-flows* (a set of operations with dependencies)
- It ingests scripts (called JOBS) that are executed on compute nodes. JOBS have properties (e.g. number of nodes, maximum wall-time)
- It allows multiple users to share resources on the same system, enforce *policies* based on permissions or priorities

TORQUE top useful commands

Check status of JOBS and QUEUES

```
$ qstat -q
$ qstat -u $USER
```

Submit a JOB script

```
$ qsub my_script.sh
114235.master1
```

Check status of JOBS and QUEUES

```
$ qdel 114235
```

Manage software: MODULES

- MODULES is a set of scripts that help setting environment right
- It allows to manage several versions of the same library or compiler transparently
- It is *semi-automatic*, module scripts can control dependencies and set environment right (e.g. a library needs a specific compiler version)

MODULES can help you but they can also be a source of confusion. Use them properly!

MODULES top useful commands

Check installed modules

```
$ module avail
```

Check current load modules

```
$ module avail
```

Load/unload modules

```
$ module load ...
$ module unload ...
```

Purge the environment (= *start from clean scratch*)

```
$ module purge
```

Compile a CPU application

Few essential steps…

- Load desired compiler using module (default GCC 4.4)
- Load additional libraries using module (e.g. MPI)

Configure, make,…. GO!

Ok but.. How do I run on a HPC?

- Login nodes allow users to access the system and compile their own program… not to run those programs!
- Parallel jobs (jobs that run programs able to exploit distributed memory systems) require more than one compute node
- Nodes are shared across several users, users sometimes need to wait for availability of them

By creating **a JOB SCRIPT** it is possible to interact with the scheduler, accessing resources (policies) and get stuff done :-)

Write your own submission script

```
#!/bin/bash
#
#   Set a job name (as reporte in qstat)
#PBS -N my_first_job
#
#   Request to run on GPU queue
#PBS -q reserved1
#
#   Request 1 processors on 1 node
#PBS -l nodes=1:ppn=1
#
#   Request 5 minutes of walltime
#PBS -l walltime=0:05:00
#
#   Request that stdout & stderr  go to the same file
#PBS -j oe
```

Write your own submission script

By default, the scripts are executed in your home directory, not the directory from which they were submitted. It is usually good practice to move in the directory from which the job was submitted.

```
...

cd $PBS_O_WORKDIR

module purge
module load cudatoolkit/6.5

uname -a
```

Understanding the GPU

- GPU is a throughput processors
  - Thousands of floating-point units working together
- GPU sits on the PCI-E bus
  - GPU memory is separated from CPU memory
- GPU is controlled (*driven*) by the CPU
- Each GPU has a specific hardware architecture
  - Current architecture is KEPLER (compute capability 3.5)
- GPU with the same architecture may have different HW variation (e.g. amount of device memory)

GPU K20

SMX = Streaming Multi-processor

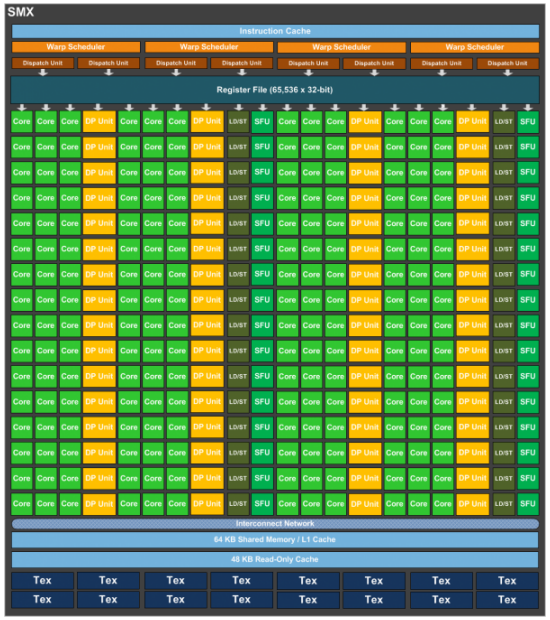
- 192 CUDA cores
- 64k registers
- 64Kb of shared memory
- up to 2K active threads

Each K20 has 13 SMX

- 2496 CUDA cores
- ~208 GB/s bandwidth
- 5 GByte memory
- ~1.1 TFlop/s DP (peak)
- ~225 Watt (peak)

2 versions:

- K20m, passive cooled (for servers)
- K20c, active cooled (for workstations)



What do I need to use a GPU?

Q: I have a GPU at home! How can I use it?

A: You need 2 components…

1. The NVIDIA DRIVER is responsible of the interaction between OS and HW)
2. The NVIDIA SDK includes GPU compiler (NVCC), CUDA tools (nvprof, nvvp, cuda-gdb), CUDA libraries
   - libcudart.so : CUDA run-time
   - libcublas.so, libcufft.so, libcurand.so, … : CUDA math libraries

Plus NVIDIA examples, a wide set of examples to help developers become familiar with CUDA ecosystem!

What do I need to use a GPU?

https://developer.nvidia.com/cuda-downloads

Home > CUDA ZONE > Tools & Ecosystem > CUDA Toolkit > CUDA 7 Downloads

CUDA 7 Downloads

Check out:

- CUDA 7 Performance Report and Webinar Recording
- An informative webinar by Ujval Kapasi, NVIDIA's CUDA Product Manager CUDA 7 Features and Overview
- The Power of C++11 in CUDA 7, another technical blog on Parallel Forall.

If you find any issues please file a bug (requires membership of the CUDA Registered Developer Program).

Please Note: In CUDA 7.0, the cuFFT library has a known issue that can lead to incorrect results for certain inputs sizes less than or equal to 1920 in any dimension when cufftSetStream() is passed a non-blocking stream (e.g., one created using the cudaStreamNonBlocking flag of the CUDA Runtime API or the CU_STREAM_NON_BLOCKING flag of the CUDA Driver API).

Windows | Linux x86 | Linux POWER8 | Mac OSX

Version	Network Installer	Local Package Installer	Runfile Installer
Fedora 21	Coming Soon	Coming Soon	Coming Soon
OpenSUSE 13.2	RPM (3KB)	RPM (1GB)	RUN (1.1GB)
OpenSUSE 13.1	RPM (3KB)	RPM (1GB)	RUN (1.1GB)
RHEL 7 CentOS 7	RPM (10KB)	RPM (1GB)	RUN (1.1GB)
RHEL 6 CentOS 6	RPM (18KB)	RPM (1GB)	RUN (1.1GB)
SLES 12	RPM (3KB)	RPM (1.1GB)	RUN (1.1GB)
SLES 11 (SP3)	RPM (3KB)	RPM (1.1GB)	RUN (1.1GB)
SteamOS 1.0-beta			RUN (1.1GB)
Ubuntu 14.10	DEB (3KB)	DEB (1.5GB)	RUN (1.1GB)
Ubuntu 14.04*	DEB (10KB)	DEB (902MB)	RUN (1.1GB)
Ubuntu 12.04	DEB (3KB)	DEB (1.3GB)	RUN (1.1GB)
GPU Deployment Kit	Included in Installer	Included in Installer	RUN (4MB)

Linux Getting Started Guide

Documentation
Release Notes
End User License Agreement
Online Documentation
CUDA Toolkit Overview
Checksums

nvidia-smi

nvidia-smi - NVIDIA System Management Interface program – is a tool part of the NVIDIA driver that provides:

- Real-time monitoring of GPU status
  - Memory usage, active processes associated to a GPU, fan speed, temperature, instantaneous power,…
- Detailed information of GPU HW
  - Clocks speeds, ECC errors, PCI-E information, firmware versions
- Ability to tweak and change GPU properties (need root privileges)

NVSMI is also a library for GPU management (python binds exist)

What GPU I am using?

There are various ways to check which GPU is installed in a system and gather all HW characteristics. NVIDIA provides an example for this…

```
…

cd $PBS_O_WORKDIR

module purge
module load cudatoolkit/6.5

deviceQuery
```

Compile a GPU application

Few essential steps…

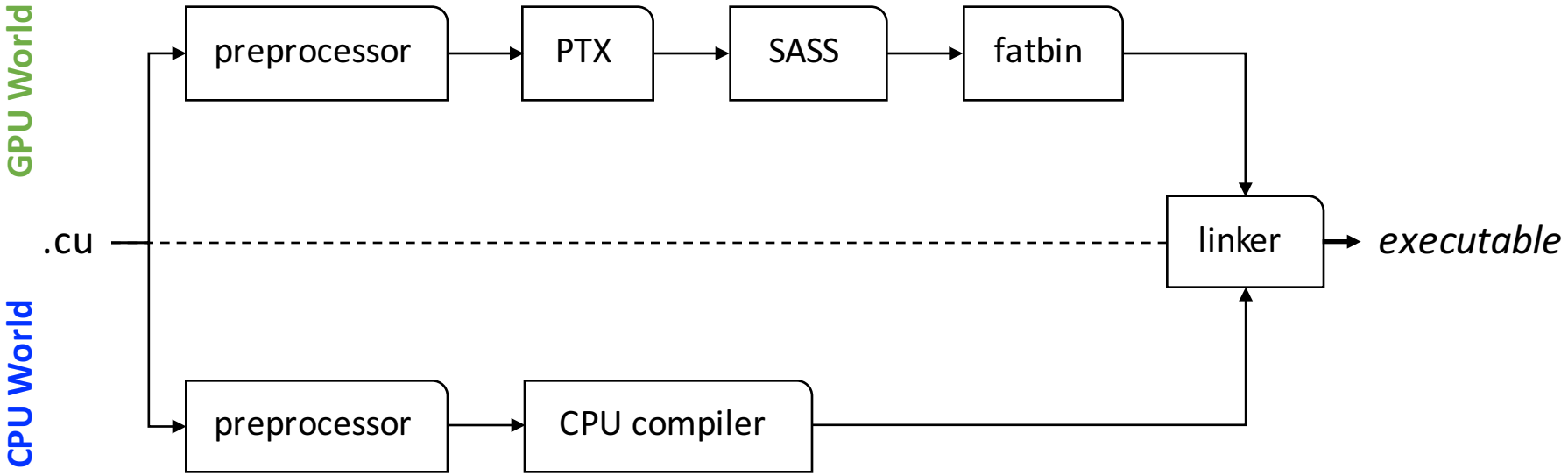
- Load desired compiler using module (default GCC 4.4)
- Load CUDA SDK module and CUDA compilers
- Load additional libraries using module (e.g. MPI)

```
$ module load cudatoolkit/6.5

$ which nvcc
/u/shared/programs/x86_64/cuda/6.5.14/bin/nvcc

$ nvcc -version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2014 NVIDIA Corporation
Built on Thu_Jul_17_21:41:27_CDT_2014
Cuda compilation tools, release 6.5, V6.5.12
```

NVCC top useful options

NVCC FLAG	Description
-O3	Enable compiler optimizations for CPU code (default CPU compiler is GCC)
-arch=sm_35	Specify the name of the class of NVIDIA GPU (*virtual*) architecture for which the CUDA input files must be compiled.
--ptxas-options=-v	Show resource usage such as registers and memory of the GPU code.
-pg	Generate debug & profile information for CPU code (gprof-style)
-G	Generate debug information for GPU code.

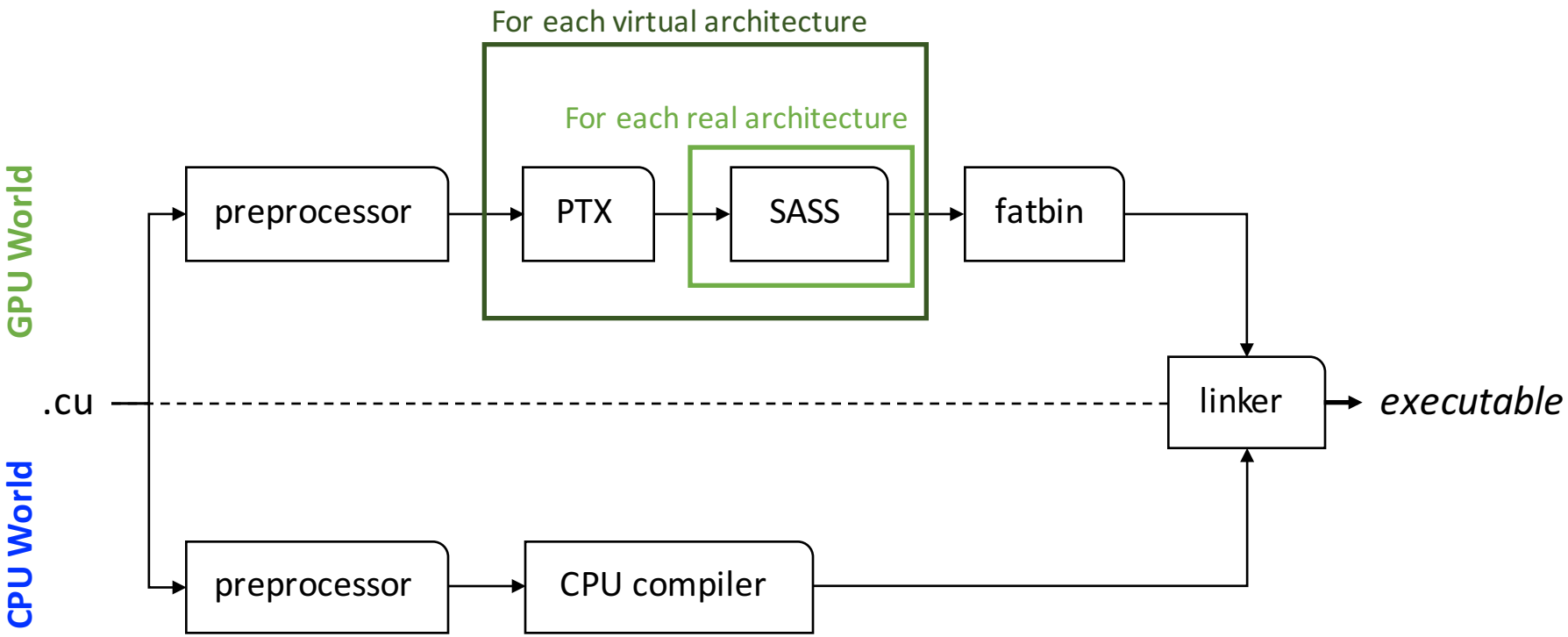
What NVCC does?



GPU code is *embedded* into the object file during the compile step.
Final linking to create final executable can be done using any linker.

http://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html

What NVCC does?



GPU code is *embedded* into the object file during the compile step.
Final linking to create final executable can be done using any linker.

http://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html

Execute a GPU application

To COMPILE a GPU application you need the CUDA SDK (libraries and compilers), not necessarily the physical hardware.

To EXECUTE a GPU application, you need the physical GPU :-)

You execute a GPU application in the same way you execute a CPU application, simple as that!

Q: What about if I have two GPUs in my system, how I can select the GPU I want?

A: There are multiple ways to do it, for example using CUDA_VISIBLE_DEVICES

Programming GPU: CUDA

CUDA (Compute Unified Device Architecture) is a programming model created by NVIDIA to leverage passive parallel GP-GPU architectures:

- CUDA C/C++ is an extension of industry-standard C/C++ (not yet fully compliant)
- GPU platforms are also accessible to software developers through CUDA-accelerated libraries, compiler directives (such as OpenACC)
- Straightforward APIs to manage devices, memory, ...
- FORTRAN support provided by compiler from PGI
- lots of example code and good documentation – 2~4 week initial learning curvelarge user community (forums, articles, books, publications, ...)

Best Practices

- **Never run your application on the login nodes!**
- Set the wall-time to a reasonable limit
  - For this training, GPU resources are limited… keep wall-time low
- Request only the resources you really need
  - For this training, GPU resources are limited… request one node only
- Make sure you know and you control where your application writes its output
- HOME is where you save your source code, important files, … run your JOB from SCRATCH (Lustre-based, faster I/O!)

NVIDIA TESLA for computing

	K10	K20	K20x	K40	K80
Chipset	GK104	GK110	GK110	GK110B	GK210
Compute Architecture	3.0	3.5	3.5	3.5	3.7
SMX	2 x 8	13	14	15	2 x 13
CUDA cores	3072	2496	2688	2880	2 x 2496
Memory	2 x 4 GByte	5 GByte	6 GByte	12 GByte	2 x 12 GByte
Core Freq.	745 MHz	706 MHz	732 MHz	745 MHz	562 MHz
Max Core freq.	745 MHz	706 MHz	732 MHz	875 MHz	875 MHz
Mem Bandwidth	320 GB/s	208 GB/s	250 GB/s	288 GB/s	480 GB/s
Max SP FLOP/s	4.56 TFlop/s	3.52 TFlop/s	3.95 TFlop/s	4.29 TFlop/s	8.74 TFlop/s
Max DP FLOP/s	--	1.17 TFlop/s	1.31 Tflop/s	1.43 TFlop/s	2.91 TFlop/s
PCI-e	Gen2	Gen2	Gen2	Gen3	Gen3
TDP	225 W	225 W	235 W	245 W	< 300 W