INTRODUCTION TO GPU COMPUTING FOR STATISTICIANS

Will Landau, Prof. Jarad Niemi

OUTLINE

- Why GPU computing?
- Computer processors
  - CPUs
  - GPUs
- Parallel computing
- Our GPU cluster

WHY GPU COMPUTING?

The whole point is to use new hardware (the GPU) in order to make long, repetitive calculations run faster.

On the Utility of Graphics Cards to Perform Massively Parallel Simulation of Advanced Monte Carlo Methods

Anthony LEE, Christopher YAU, Michael B. GILES,
Arnaud DOUCET, and Christopher C. HOLMES

Table 1. Running times for the population-based MCMC sampler for various numbers of chains M.

M	CPU (min)	8800 GT (sec)	Speedup	GTX 280 (sec)	Speedup
8	0.0166	0.887	1.1	1.083	0.9
32	0.0656	0.904	4	1.098	4
128	0.262	0.923	17	1.100	14
512	1.04	1.041	60	1.235	51
2048	4.16	1.485	168	1.427	175
8192	16.64	4.325	230	2.323	430
32,768	66.7	14.957	268	7.729	527
131,072	270.3	58.226	279	28.349	572

On the Utility of Graphics Cards to Perform Massively Parallel Simulation of Advanced Monte Carlo Methods

Anthony LEE, Christopher YAU, Michael B. GILES,
Arnaud DOUCET, and Christopher C. HOLMES

Table 2. Running times for the sequential Monte Carlo sampler for various values of N.

N	CPU (min)	8800 GT (sec)	Speedup	GTX 280 (sec)	Speedup
8192	4.44	1.192	223.5	0.597	446
16,384	8.82	2.127	249	1.114	475
32,768	17.7	3.995	266	2.114	502
65,536	35.3	7.889	268	4.270	496
131,072	70.6	15.671	270	8.075	525
262,144	141	31.218	271	16.219	522

On the Utility of Graphics Cards to Perform Massively Parallel Simulation of Advanced Monte Carlo Methods

Anthony LEE, Christopher YAU, Michael B. GILES,
Arnaud DOUCET, and Christopher C. HOLMES

Table 3. Running time (in seconds) for the sequential Monte Carlo method for various values of N.

N	CPU	8800 GT	Speedup	GTX 280	Speedup
8192	2.167	0.263	8	0.082	26
16,384	4.325	0.493	9	0.144	30
32,768	8.543	0.921	9	0.249	34
65,536	17.425	1.775	10	0.465	37
131,072	34.8	3.486	10	0.929	37

Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures

Marc A. Suchard[1], Quanli Wang[2,5], Cliburn Chan[3], Jacob Frelinger[4], Andrew Cron[5] & Mike West[5]

n	gpu 1	gpu 3	tesla	cpu 8	cpu 1	mac gpu	mac cpu
10^2	1.225	1.243	1.226	3.0	3.0	2.119	5.0
10^3	1.42	1.36	1.45	20.0	20.0	3.654	30.0
10^4	3.18	2.46	3.49	94.0	191.0	18.78	277.0
10^5	20.4	13.1	23.7	386.0	1,907.0	169.7	2,758.0
10^6	192.0	119.5	224.6	3,797.0	19,048.0	1,118.1	27,529.0
5×10^6	954.0	591.0	1,116.0	17,667.0	95,283.0	5,785.8	141,191.0

Table: Running times (in seconds) for 100 iterations of the MCMC analysis of TDP model

COMPUTER PROCESSORS

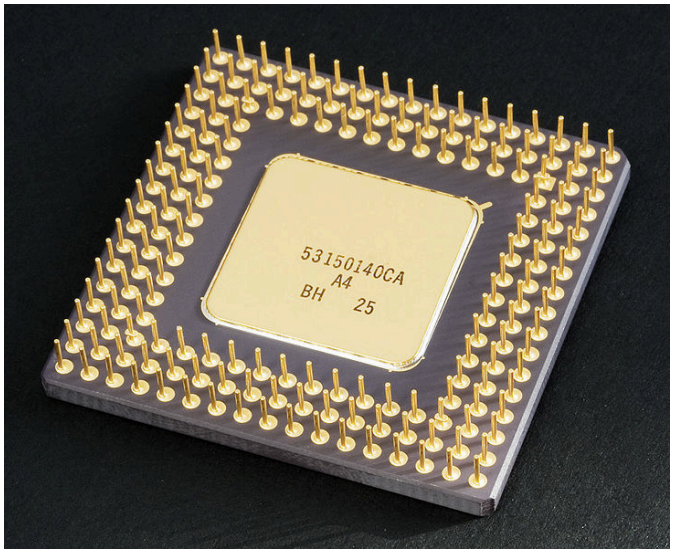
Processing Unit: a computer chip that performs executive functions: arithmetic, logic, etc.

Core: One of possibly many "sub-processors" placed on the same processing unit (chip). They work together, but each of them has the full functionality of a processing unit.

THE CPU

CPU = "Central Processing Unit"

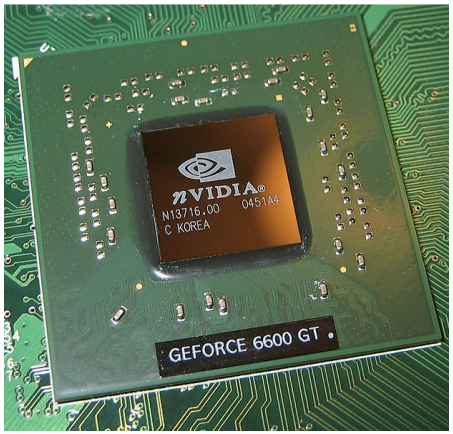
- The kind of processor you would find in a regular computer.
- Designed for general purpose computing.
- Does parallel computing too, but differently than a GPU.



THE GPU

GPU = "Graphics Processing Unit"

- The kind of processor that you would find in a graphics card or video card.
- Originally designed to speed up graphics throughput in video games, not to do general purpose computing.
- Performs massively parallel computing, able to run orders of magnitude more threads at a time than a CPU.
- Higher memory bandwidth than the CPU.

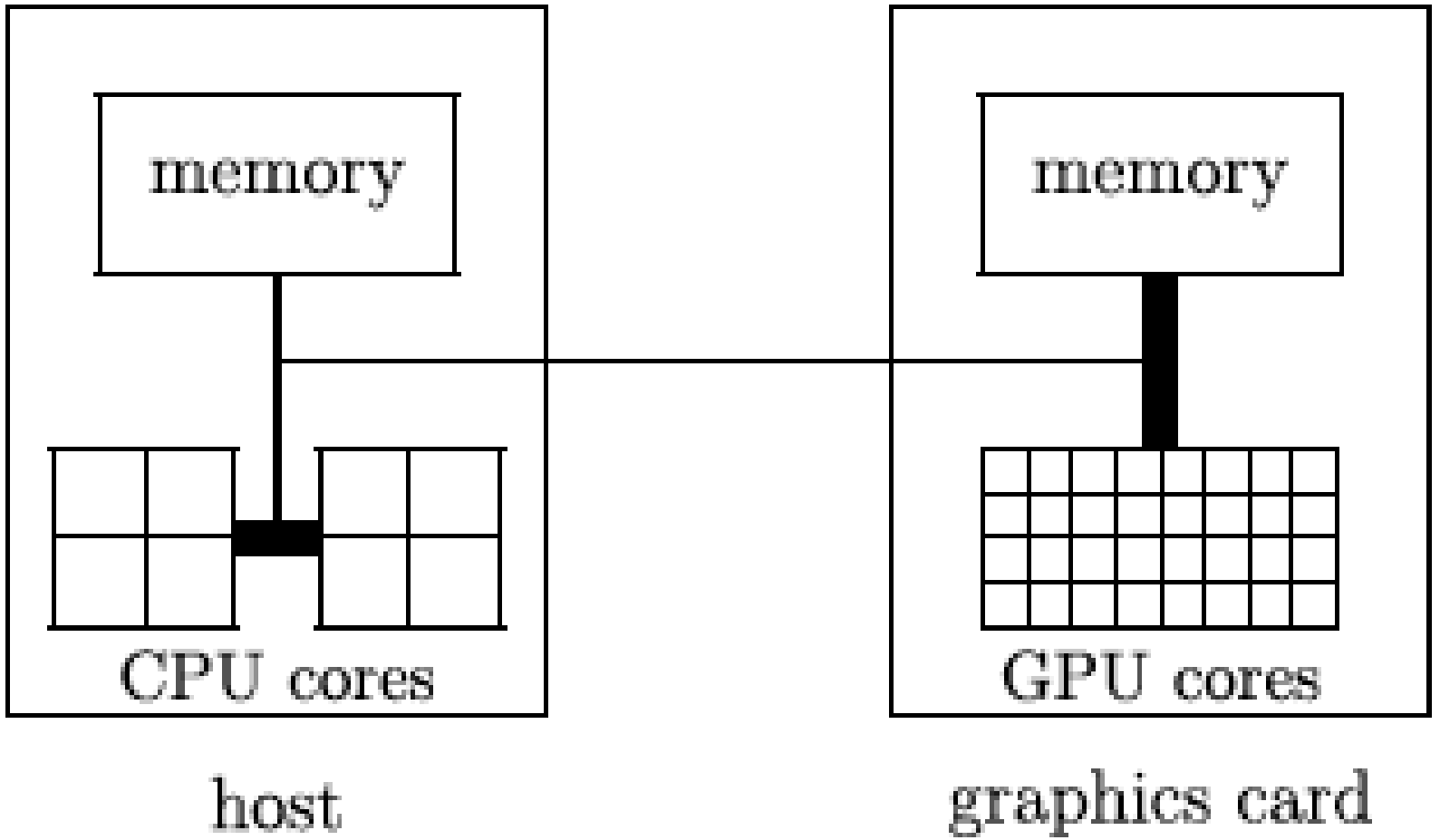


HOW THE CPU AND GPU WORK TOGETHER

A GPU can't run a whole computer on its own because it doesn't have access to all the computer's hardware.

In a GPU-capable computer, the CPU is the main processor, and the GPU is an optional hardware add-on.

The CPU uses the GPU like a human would use a hand-held calculator: the CPU does all the main thinking and the GPU does the most cumbersome bits and pieces of number-crunching.


memory
memory
CPU cores
GPU cores
host
graphics card

GPUs AND PARALLEL COMPUTING

parallelism: The practice of running multiple calculations simultaneously.

GPUs parallelize repetitive arithmetic calculations much better than CPUs.

Note: there are several kinds of parallelism, all implemented differently:

1. CPU parallelism
2. GPU parallelism
3. parallel cloud computing
4. parallelism for openMP

I will only focus on GPU parallelism, which does not completely generalize to other kinds of parallelism.

Theoretical maximum speedup, Amdahl's quantity:

$$\frac{1}{1 - P + \frac{P}{N}}$$

P: fraction of the program that can be parallelized

N: number of parallel processors

As $N \to \infty$:

$$\frac{1}{1 - P}$$

So if 99% of the program can be parallelized, theoretically we could have a 100-fold speedup.

PARALLELIZING A WORKLOAD ON A GPU

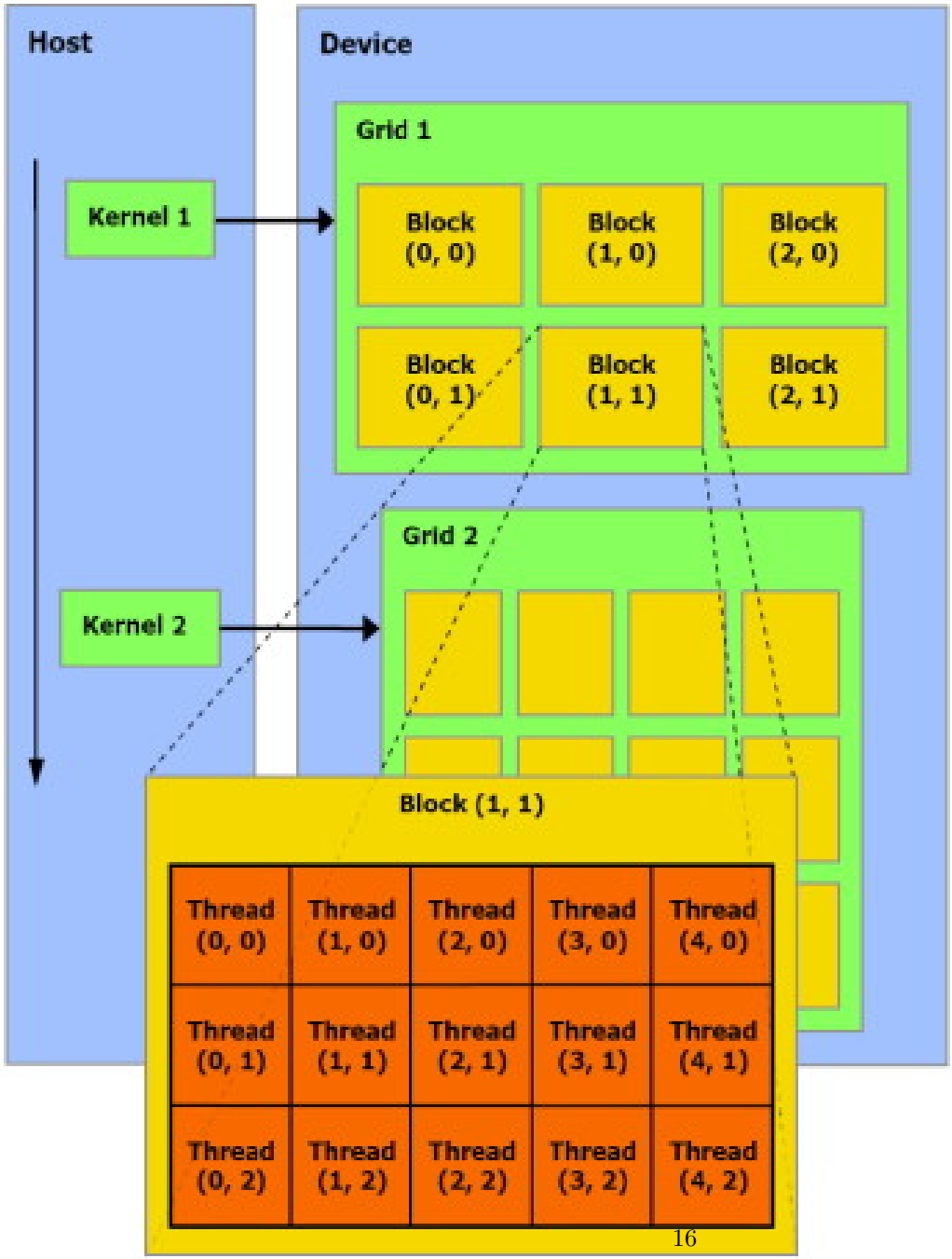
1. The CPU sends a CPU-to-GPU command called a **kernel** to a single GPU core.

2. The GPU core multitasks to execute the command:

   a. The GPU makes $B \cdot T$ copies of the kernel's code, and then runs all those copies simultaneously. Those parallel copies are called **threads**.

   b. The $B \cdot T$ threads are partitioned into B groups, called **blocks**, of T threads each.

   c. The sum total of all the threads from a kernel call is a **grid**.


Host
Device
Grid 1
Kernel 1
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Grid 2
Kernel 2
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)

GPUS WITH CUDA: COMPUTE UNIFIED DEVICE ARCHITECTURE

- First released by NVIDIA in 2007
- Supports CUDA C, an extension of C for programs that can run on GPUs and CPUs simultaneously.

CUDA systems have the data crunching power of the GPU and the versatility of the CPU.

WE HAVE CUDA SYSTEMS!

- impact1.stat.iastate.edu (up and running)
- impact2.stat.iastate.edu (coming soon)
- impact3.stat.iastate.edu (coming soon)
- impact4.stat.iastate.edu (coming soon)

SPECS OF IMPACT1

- Linux: Red Hat Enterprise Linux Server release 6.2 (Santiago)
- no GUI or remote desktop capabilities yet (use the command line for now)
- Four CUDA-capable Tesla M2070 GPUs, each with:
  - 448 cores.
  - CUDA Driver and Runtime Version 4.1

Enter:

```
cd /usr/local/NVIDIA_GPU_Computing_SDK/C/bin/linux/release
```

and then:

```
./deviceQuery
```

in the command line while logged into impact1 for more details.

LOGGING INTO IMPACT1

1. Connect to the internet and open your favorite command line utility: Terminal in Mac OS X, Command Prompt in Windows, etc.
2. Type in:

```
ssh -p 323 your_ISU_ID@impact1.stat.iastate.edu
```

   and press enter.

For me, a login looks like this:

```
~> ssh -p 323 landau@impact1.stat.iastate.edu
Last login: Mon May 28 11:37:06 2012 from landau.student.iastate.edu
KRB5CCNAME: Undefined variable.
[landau@impact1 ~]$
```

NOTE: You may have to change "-p 323" to "-p <some other port number>"

Contact me (at landau@iastate.edu or in person) if you'd like help with:

- Command line tools for logging in.
- Easy ways to transfer files between impact1 and your local machine.
- SSH key setup: for logging in from your personal machine without having to type your password.
- Setting up a shortcut command for logging in so that you don't have to type in all of "ssh -p 323 your_ISU_ID@impact1.stat.iastate.edu" every time you log in.

For questions about using command line tools or the linux file system in general, contact me or see:

- http://www.makeuseof.com/tag/an-introduction-to-the-linux-command-line/
- http://www.freesoftwaremagazine.com/articles/command_line_intro
- http://tldp.org/LDP/intro-linux/html/
- http://tldp.org/LDP/intro-linux/html/sect_03_01.html
- http://dhavalv.wordpress.com/2007/10/17/quick-introduction-to-linux-filesystem-fhs/
- http://linux.die.net/Intro-Linux/chap_03.html
- http://linux.about.com/od/itl_guide/a/gdeitl28t02.htm

IMPORTANT DIRECTORIES ON IMPACT1

- `/home/your_ISU_ID`

  Your private home folder on the department's linux repository (also connects with linux10 and linux11). Code and data in here are stored remotely on the linux repository but used locally with the hardware in impact1.

- `/Cyfiles/your_ISU_ID`

  Your private Cyfiles folder. Code and data in here are stored remotely on the university's Cyfiles system but run locally on impact1.

- **/tmp**

  Everything in here is stored locally on impact1. To ensure fast computation, put your huge data set here. That way, your program doesn't have to stream lots of data through a network. WARNING: /tmp automatically empties periodically.

- **/usr/local/NVIDIA_GPU_Computing_SDK**

  Contains example code for those of you who want to learn CUDA C. Stored locally on impact1.

LECTURE SERIES MATERIALS

These lecture slides, a tentative syllabus for the whole lecture series, and code are available at:

https://github.com/wlandau/gpu.

After logging into you home directory on impact1, type:

```
git clone https://github.com/wlandau/gpu
```

into the command line to download all the course materials.

OUTLINE

- Why GPU computing?
- Computer processors
  - CPUs
  - GPUs
- Parallel computing
- Our GPU cluster

TENTATIVE SYLLABUS

ISU Statistics Department

GPU Computing Lecture Series

Tentative Syllabus

Lectures are on Thursdays at 10 AM in Snedecor 2113.

Lecture 1: Introduction to GPU computing

Lecture 2: Usage and performance of the R package, gputools.

Lecture 3: A codeless introduction to the paradigm of GPU parallelization.

Lecture 4: Introduction to programming on the GPU using CUDA C.

Lecture 5: CUDA C: shared memory and thread cooperation.

Lecture 6: CUDA C: performance measurement, race conditions, atomics, and warps.

Lecture 7: CUBLAS and CULA: linear algebra libraries for CUDA C

Lecture 8: Rejection sampling in CUDA C

Lecture 9: An introduction to programming in Python: preparation for PyCUDA

Lecture 10: PyCUDA: a CUDA-implemented Python library for GPU computing

Lecture 11: ABC-SysBio: A tool for parameter inference and model selection

REFERENCES

D. Kirk, W.H. Wen-mei, and W. Hwu. *Programming massively parallel processors: a hands-on approach.* Morgan Kaufmann, 2010.

A. Lee, C. Yau, M.B. Giles, A. Doucet, and C.C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced monte carlo methods. *Journal of Computational and Graphical Statistics*, 19(4): 769-789, 2010.

J. Niemi and M. Wheeler. Statistical computation on GPGPUs. Iowa State University. 28 September, 2011.

J. Sanders and E. Kandrot. *CUDA by Example.* Addison-Wesley, 2010.

M.A. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West. Understanding gpu programming for statistical computation: Studies in massively parallel mixtures. *Journal of Computational and Graphical Statistics.* 19(2): 419-438, 2010.