Exercise 1
Numerical Integration and Multithreading

High Performance Computing for Science and Engineering I

September 26, 2014

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Administrative Notes

- Please contact one of the TAs if you need anything
- Hand-in to your TA
  - You can find your assigned TA at the course webpage
  - If you have a special request, please ask by the end of the day
- Exam
  - Friday, 19.12.2014, 09:00 - 12:00
  - Computer rooms

The Brutus and Euler Clusters

High Performance Computing for Science and Engineering I

September 26, 2014

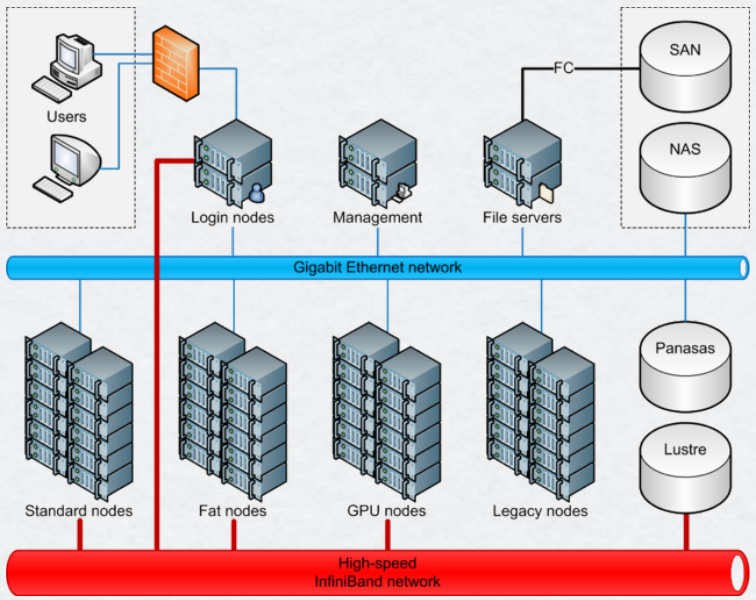
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

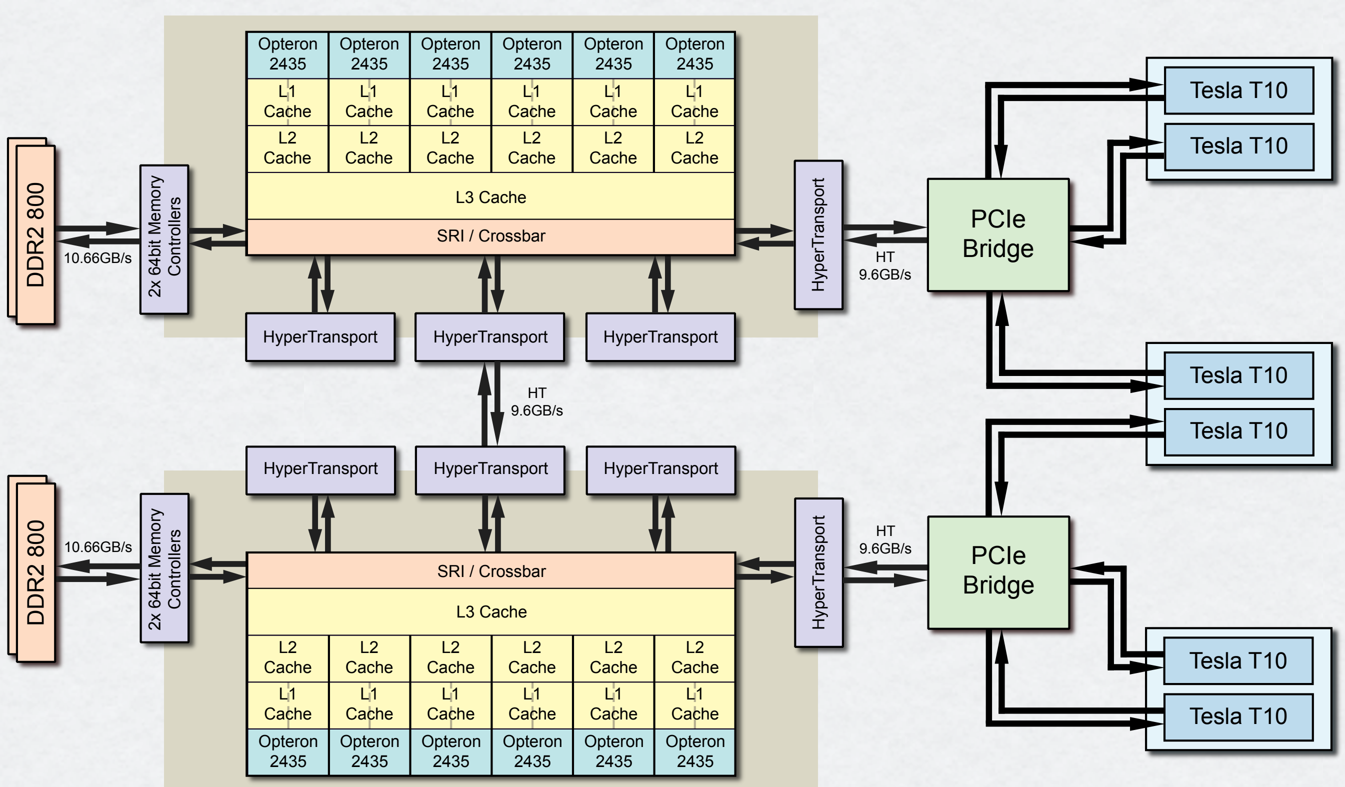
Brutus and Euler

- High performance clusters of ETH Zurich
- Brutus: composed of different kinds of compute nodes
  - 120 nodes with 48 cores each
  - 36x Nvidia Tesla C2050 (Fermi Architecture)
  - Many others (check brutuswiki.ethz.ch)
- Euler: improved performance compared to Brutus
  - 416 compute nodes
  - Two 12-core Intel Xeon processors
  - http://brutuswiki.ethz.ch/brutus/Introducing_EULER

Anatomy of a Cluster

Brutus



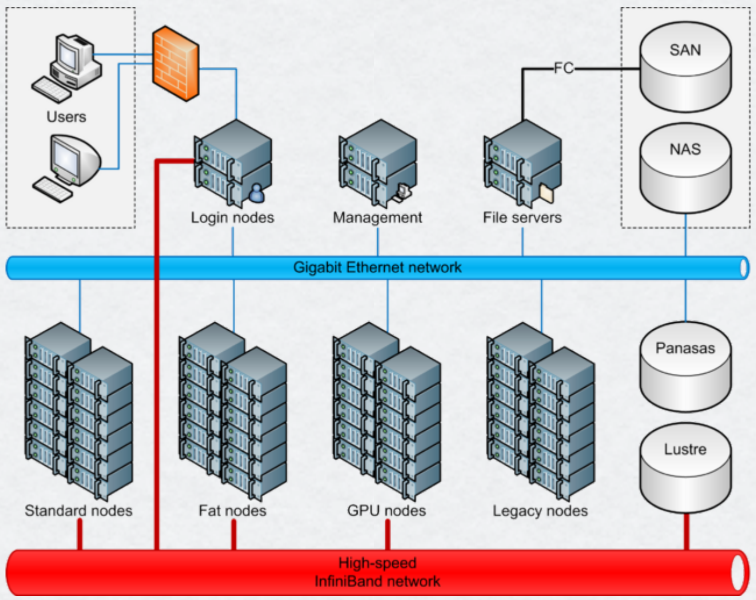


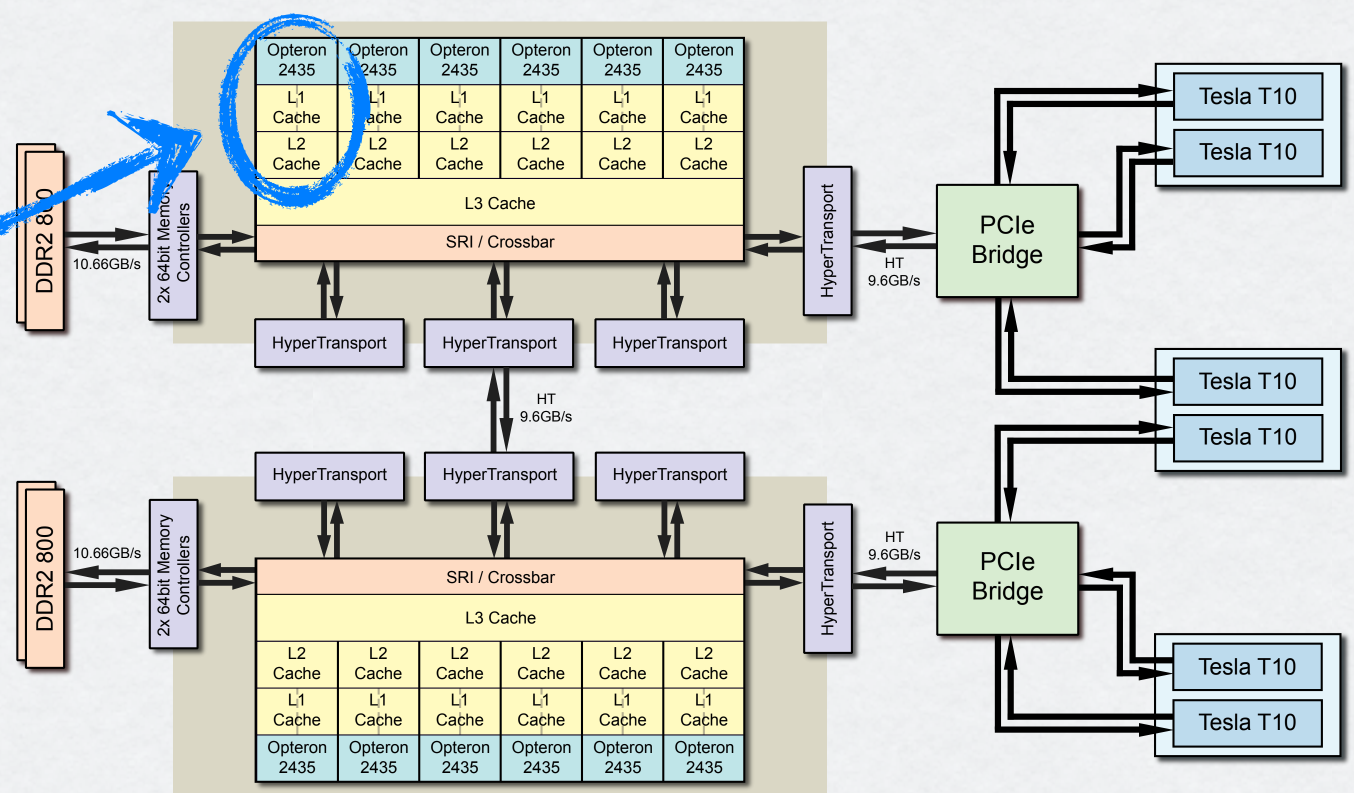
Details

- Effective bandwidth with 12 cores: 20GB/s (STREAM Benchmark)

Anatomy of a Cluster

Brutus





CPU core

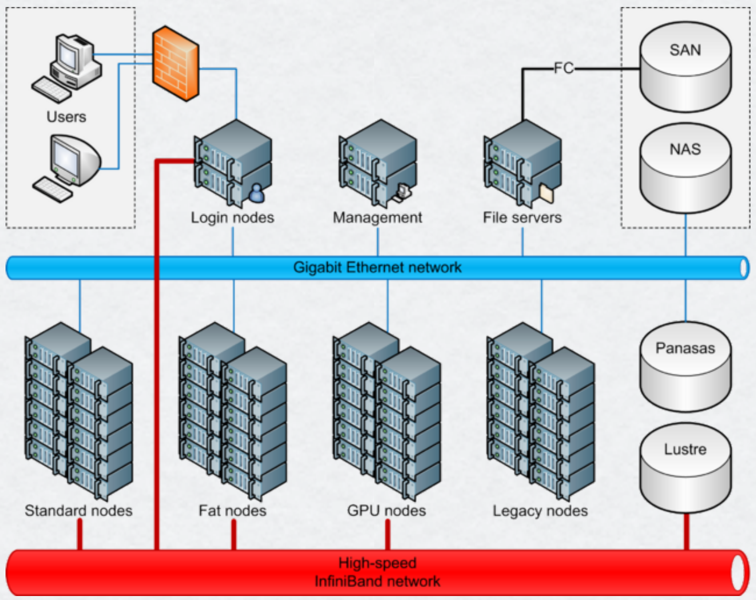
C++

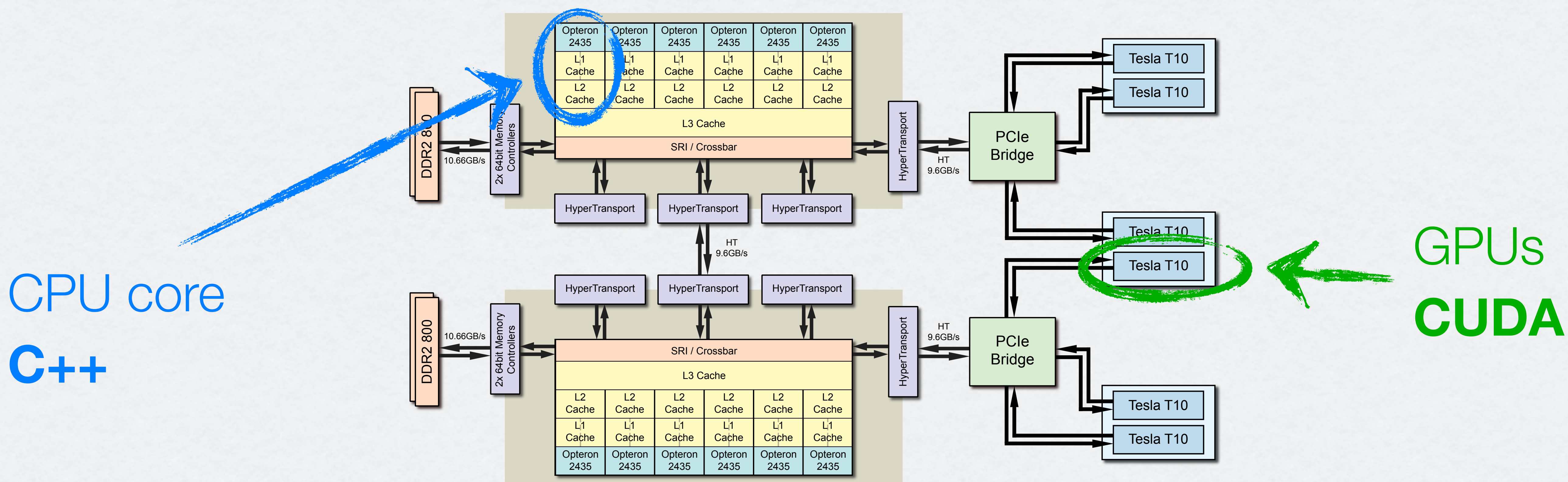
Details

- Effective bandwidth with 12 cores: 20GB/s (STREAM Benchmark)

Anatomy of a Cluster

Brutus



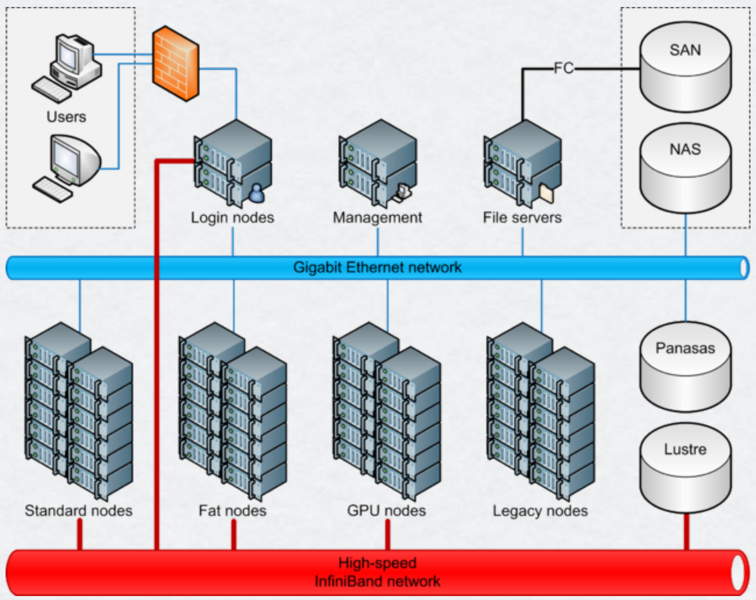


Details

- Effective bandwidth with 12 cores: 20GB/s (STREAM Benchmark)

Anatomy of a Cluster

Brutus



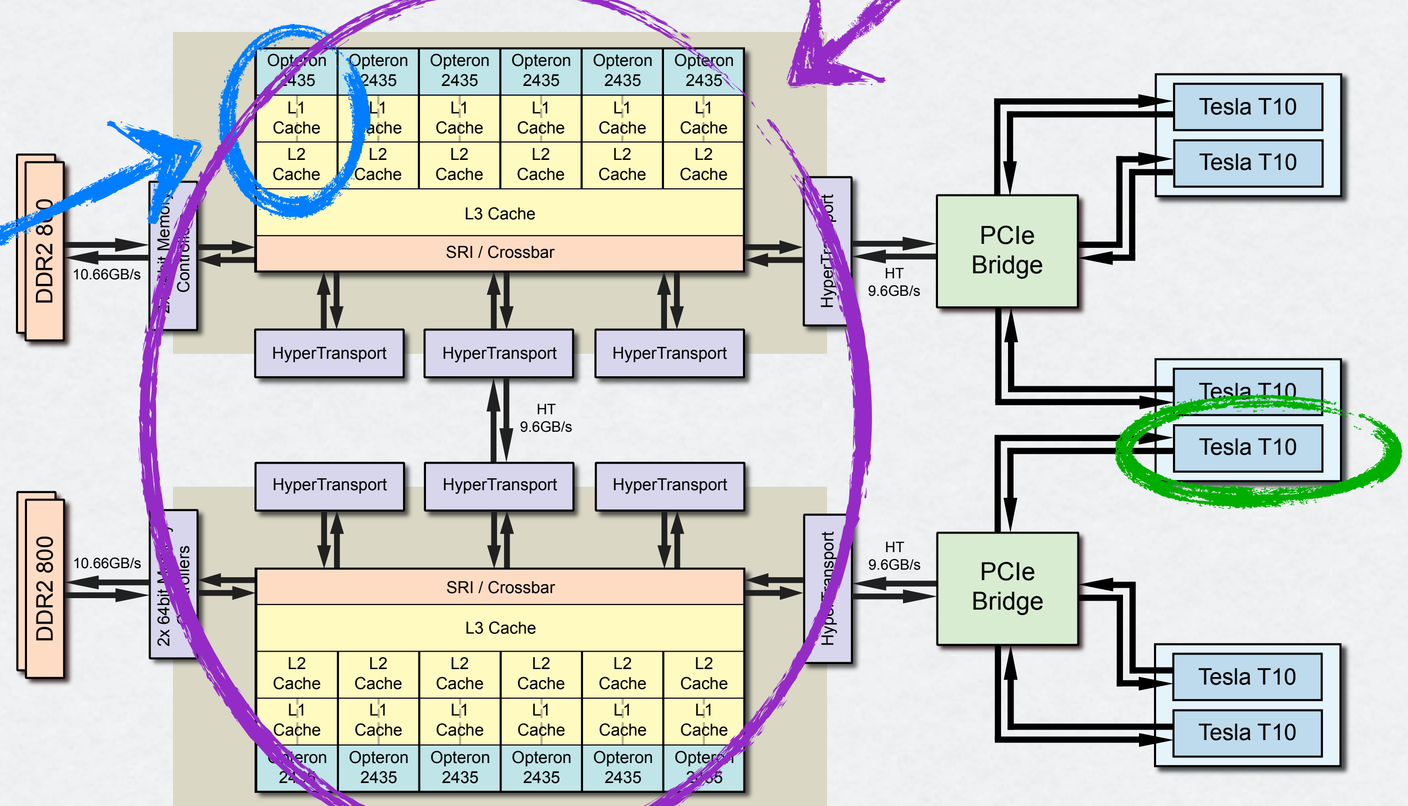
Node: multiple processors

Shared Memory

OpenMP

CPU core

C++



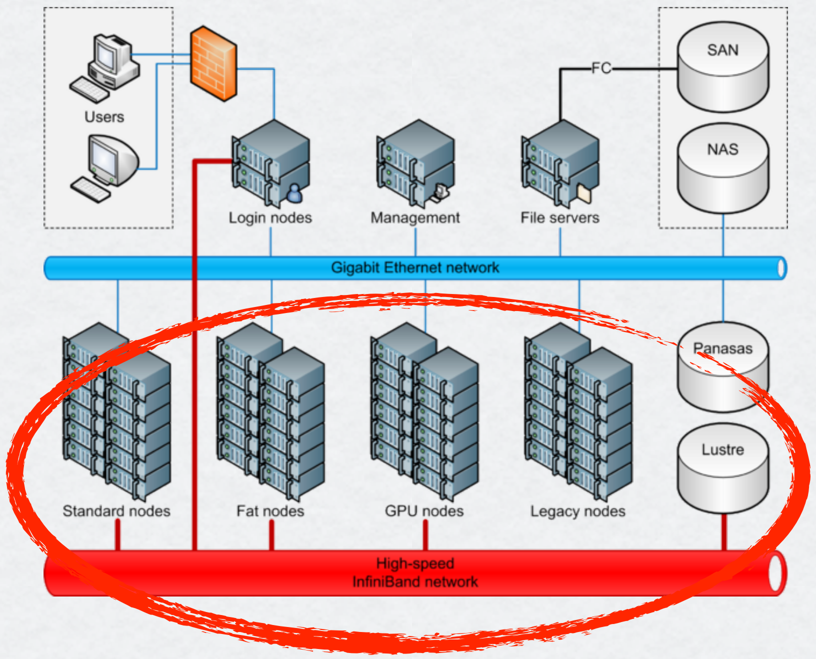
GPUs

CUDA

Details
- Effective bandwidth with 12 cores: 20GB/s (STREAM Benchmark)

Anatomy of a Cluster

Brutus



Cluster: network of nodes

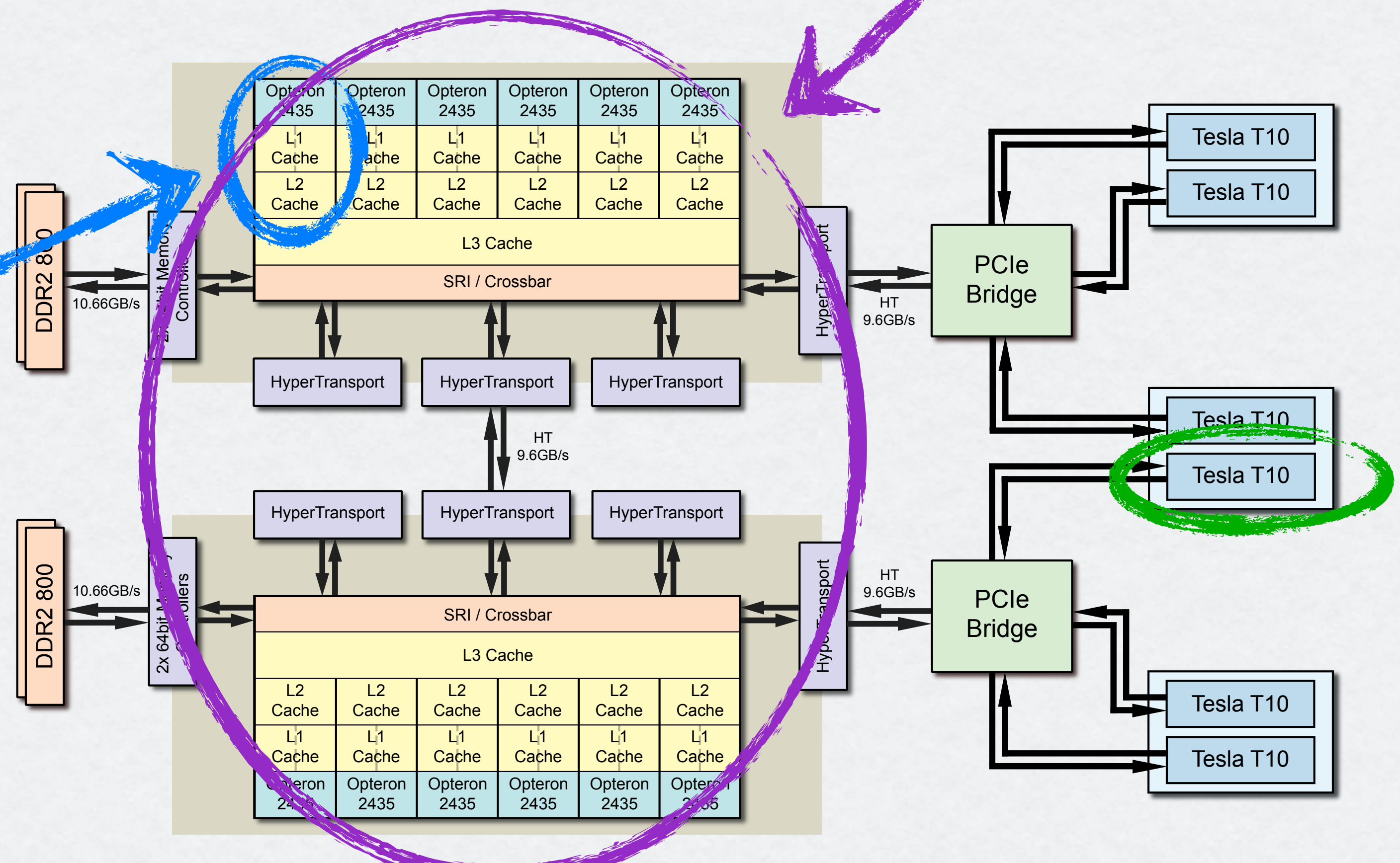
Distributed memory

MPI

Node: multiple processors

Shared Memory

OpenMP



CPU core

C++

GPUs

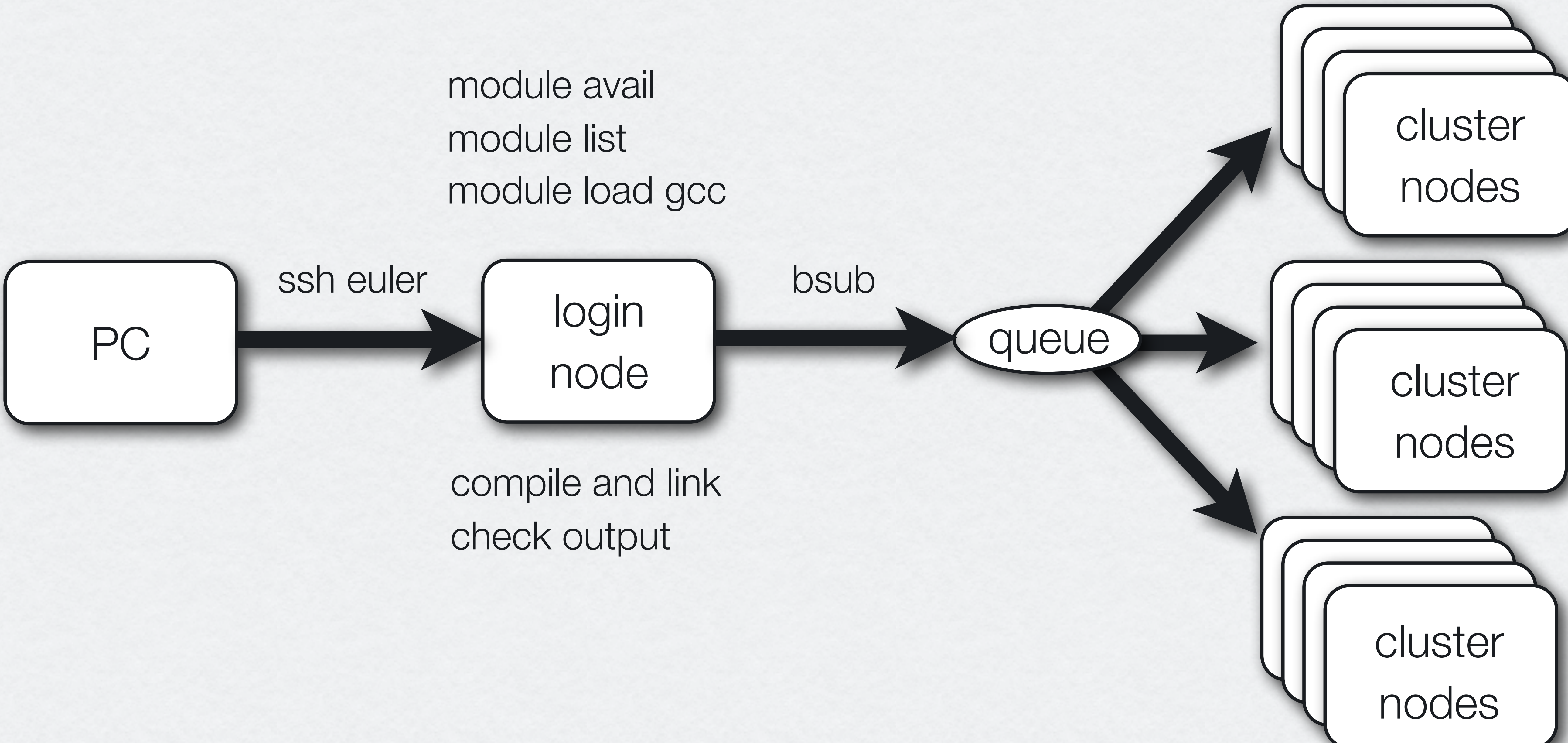
CUDA

Details

- Effective bandwidth with 12 cores: 20GB/s (STREAM Benchmark)

Accessing and using Euler

No request account for Euler :-)



Basic steps

1. Connect to a login node of Euler
2. Copy or edit your program files
3. Compile your program
4. Submit a job / run your program on compute nodes
5. Check your job (status and output)

1. Connect

- ssh -Y <usename>@euler.ethz.ch
  - -Y: Enables trusted X11 forwarding
  - Access to one of the Euler login nodes

2. Develop

- Copy your files to Euler, e.g.
  - scp code.tar.gz <username>@euler.ethz.ch:*code.tar.gz*
- Use a text editor to write/modify your code
  - vi, emacs, nano

3. Compile

- You will need the appropriate programming tools and libraries to compile your code
- Just load the environment module you need
- Examples
  - module load gcc (newer version of gcc)
  - module load open_mpi (MPI library)
  - module list (shows loaded modules)
  - module avail (what is available)
  - module unload gcc (unloads a module)

3. Compile

- Compile your code and produce the executable
- Example:
  - g++ cputest.cpp -o cputest

4. Submit your job

- The login nodes are used only for development
- The program must run on a compute node
- To do that, you must use the bsub command:
  bsub -W 00:10 -n 1 ./cputest
- You can submit script files too: bsub -n 1 < myscript

5. Check your job

- Some useful commands
  - bqueues: displays information about queues
  - bjobs: displays information about jobs (bjobs -l -u <username>)
  - bkill <jobID>: kills a job
- Output files
  - lsf.o<jobID>: created in your working directory when the job finishes
    - includes information about your job (statistics, etc.) and the messages the program prints (standard output)

Numerical Integration and Multithreading

High Performance Computing for Science and Engineering I

September 26, 2014

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Riemann sum

- The value of the integral $\int_a^b f(x)dx$
- Can be approximated with the Riemann sum:

$$S = \sum_{i=1}^{n} f(x_i^*)\Delta x$$

- where: $\Delta x = x_i - x_{i-1} = (b-a)/n$

  x_i^*: some point in the interval $[x_{i-1}, x_i]$

  $x_0 = a, x_n = b$

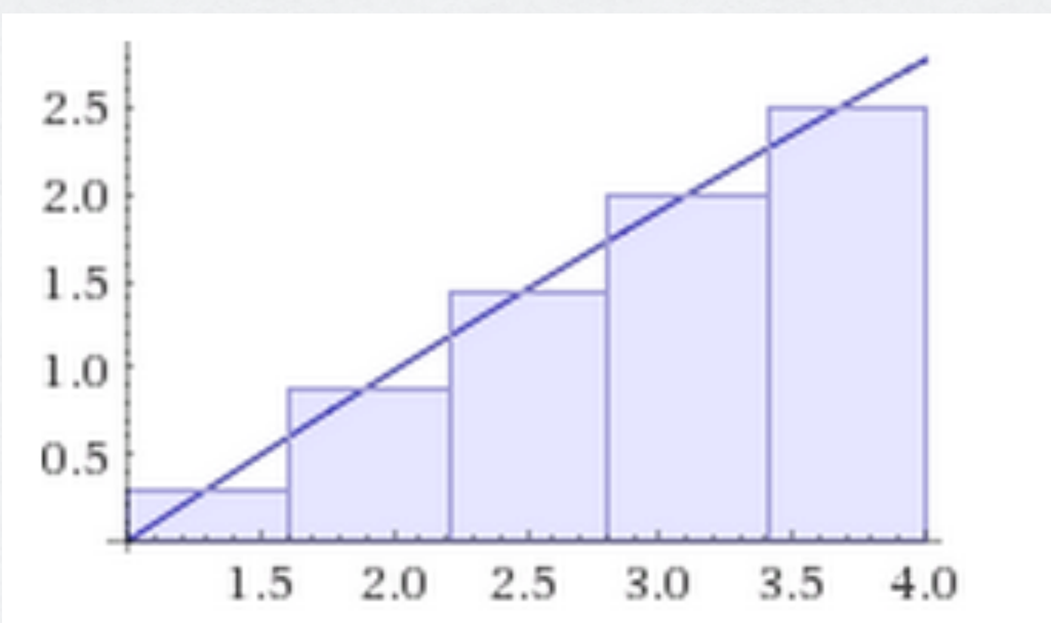
- Midpoint approximation: for each x_i^* we use:

$$\bar{x}_i = \frac{(x_{i-1} + x_i)}{2}$$

Serial C++ code

- Compute the value of $\int_1^4 f(x)dx$
- where $f(x) = \sqrt{x} \cdot ln(x)$
- http://www.wolframalpha.com
  - “integrate ln(x)*sqrt(x) using midpoint method from x=1 to 4”

Plot for 5 intervals



Exact result

$$\frac{4}{9}(4\log(64) - 7) \approx 4.282458814861639$$

Parallel code with C++11 threads

- Use multiple threads to reduce execution time
  - Distribute intervals among threads
  - Each thread should handle a different interval of the integral
- Avoid race conditions
  - Be careful with the computation of the final sum
- Verify that your implementation is correct
  - Against the output of the serial program

Time measurements

- Study how wall-clock decreases as the number of threads increases
    - Choose an appropriate number of intervals
- Plot time vs # threads and report your observations
- You can find a timer class in timer.hpp

```
timer t;
t.start();
<computations>
t.stop();
double elapsed = t.get_timing(); // time in seconds
```

Final details

- Not required to use Euler
  - Include some details about the hardware/software configuration of your system (#cores+memory, OS+compiler)
- Code from scratch
  - Become familiar with the systems at the computer rooms
- Hand in: PDF (plots, comments) + Code (not binary!)
  - To your assigned TA (check webpage)
  - Hard deadline: Next Friday 03/08/2014, 08:00
  - The solution will be uploaded then
- Ask for our help!