Training @ CINES:
How to make the most of supercomputing technologies
MPI and OpenMP for beginners

Johanne Charpentier & Gabriel Hautreux
charpentier@cines.fr  hautreux@cines.fr
Clusters Architecture

OpenMP

MPI

Hybrid MPI+OpenMP
OCCIGEN

Intel Haswell Xeon technology
Bi sockets nodes
12 cores per socket
Infiniband FDR 14 interconnect
LUSTRE storage system

OCCIGEN REACHED RANK 26 IN THE LAST TOP 500

Th. peak performance: 2,1 PFlop/s
Achieved performance: 1,63 PFlop/s
ONLY
The CPU (for central processing unit) is the smallest unit in a cluster

Processing Unit

L1 Cache  L2 Cache

RAM (Random Access Memory)

What does a CPU do?

- Interpret instructions
- Retrieves data from the RAM
- Compute using those data
- Send back the information to the RAM
- Now you have your output and can process your results
The socket is a group of CPU and represents a processor (as the ones you can buy).

What can we do with a socket?

- Use multiple cores at the same time that can access the same memory (this memory is called the L3 Cache).
- Increase the speed of the code by using parallelism.
- Shared memory => OpenMP.
The node is multiple sockets that share the same physical memory.

What can we say about a node?

- Multiple sockets
- Share the same memory space but with a loss of performance if going through the QPI Link
- More or less shared memory
- => MPI and/or OpenMP
The nodes are stored in racks and linked together using Infiniband technology.

How are the nodes connected?

- Nodes are stored in racks.
- Each node has an Infiniband connection to the switch.
- Communications are handled through the switch.
- No shared memory => MPI Only.
To use the cluster, you have to log at the front end (login node) and run your code using a job scheduler which will dispatch the MPI processes among the nodes.

How to use the cluster?

- Now that you know everything, building a code for High Performance Computing technologies becomes straightforward.
- Aim of this course: make you use both MPI + OpenMP to make the most of HPC architectures.
Computing the CPU’s peak performance implies no bandwidth bottleneck

**Peak performance** = frequency x #operations per CPU cycle

Haswell: 2 FMA AVX (‘+’ AND ‘x’ on 4 doubles) per cycle => 16ops

Peak CPU: 2.6Ghz x 16 = 41.6 Gflop/s

Peak Socket (12 CPUs) = 499.2 Gflop/s

Peak Node (2 sockets = 998.4 Gflop/s

Peak Occigen (2106 nodes) = 2,102,630.2 Gflop/s
Bandwidth peak performance computation is easy but not well-known

\[
\text{Peak performance} = \text{freq} \times \text{#data transfers/clock} \times \text{Bus width} \times \text{#interfaces}
\]

Frequency: DDR4 => 2133MHz

DDR: Double Data Rate => 2 data transfers/cycle

Bus width = 8B

2 ports per socket = 2 interfaces

Peak per socket = 2133*2*8*2 = 68,256 GB/s
CPUs on one socket access to the same memory => Bandwidth limitations

How to use the bandwidth?

• Feed your CPUs efficiently
• Give them as less data as possible
• Make those data as reusable as possible
• Everyone has to be fed using the same pipe
• People prefer macaroon
CPUs on one socket access to the same memory => Bandwidth limitations

How to use the bandwidth?

- Feed your CPUs efficiently
- Give them as less data as possible
- Make those data as reusable as possible
- Everyone has to be fed using the same pipe
- People prefer macaroon
Let’s visit Occigen!