Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

High-Performance Computing Lab for CSE **2024**

Student: Benedict Armstrong Discussed with: FULL NAME

Solution for Project 1a

Due date: 11 March 2024, 23:59

---

1. Euler warm-up [10 points]

1.1. Module System

The module system allows Euler users to quickly and easily configure their environment to use centrally installed software package. A detailed description can be found in the Module System documentation.

There are two systems currently in use. The older system is called `Environment Modules` and the newer system is called `LMOD Modules`. All new software installations are done with LMOD Modules.

```
# List all available modules
module avail

# Load a module
module load <module_name>

# list all loaded modules
module list
```

Listing 1: Module System

1.2. SLURM

The Euler cluster uses SLURM to manage and schedule jobs. To run a job on the cluster, you need to submit a job script to the SLURM scheduler. A detailed description can be found in the SLURM documentation.

1.3. Hello Euler!

We start by compiling and running a simple C program on the Euler cluster. The program is called `hello_euler.cpp` and should print "`Host name: <hostname>`" to standard out.

To run the compiled program on the cluster, we need to submit a job script to the SLURM scheduler. The job script is called `hello_euler.slurm` and should look like this:

The job can then be submitted to the SLURM scheduler with the following command:

The code and output can be found in the `hello_euler` directory.

```
#!/bin/bash
#SBATCH --job-name=hello_euler          # Job name    (default: sbatch)
#SBATCH --output=hello_euler.out        # Output file (default: slurm-%j.out)
#SBATCH --error=hello_euler.err         # Error file  (default: slurm-%j.out)
#SBATCH --time=00:01:00                 # Wall clock time limit
#SBATCH --nodes=1                       # Number of tasks
#SBATCH --ntasks=1                      # Number of tasks
#SBATCH --cpus-per-task=1               # Number of CPUs per task
#SBATCH --mem-per-cpu=1024              # Memory per CPU
#SBATCH --constraint=EPYC_9654

srun hello_euler
```

Listing 2: Job script for running hello_euler.cpp

```
sbatch hello_euler.sh
```

Listing 3: Submitting a job to the SLURM scheduler

```
#!/bin/bash
#SBATCH --job-name=hello_euler_2        # Job name    (default: sbatch)
#SBATCH --output=hello_euler_2.out      # Output file (default: slurm-%j.out)
#SBATCH --error=hello_euler_2.err       # Error file  (default: slurm-%j.out)
#SBATCH --time=00:01:00                 # Wall clock time limit
#SBATCH --nodes=2                       # Number of tasks
#SBATCH --ntasks=2                      # Number of tasks
#SBATCH --cpus-per-task=1               # Number of CPUs per task
#SBATCH --mem-per-cpu=1024              # Memory per CPU

srun hello_euler hello_euler
```

Listing 4: Job script for running hello_euler.cpp on multiple nodes

1.4. Multiple Nodes

We can run the same code on multiple nodes using the following job script:

Where we set the number of nodes to 2 and the number of tasks to 2. The output can be found in the `hello_euler_2.out` file.

2. Performance characteristics [50 points]

2.1. Peak performance

The peak performance of a cluster can be calculated using the following formula:

$$
\begin{aligned}
p_{core} &= n_{super} \times n_{FMA} \times n_{SIMD} \times f_{core} \\
p_{CPU} &= n_{core} \times p_{core} \\
p_{node} &= n_{sockets} \times p_{CPU} \\
p_{cluster} &= n_{nodes} \times p_{node}
\end{aligned}
$$

The the **Euler VII — phase 1** and **Euler VII — phase 2** nodes use the *EPYC_7H12* and *EPYC_7763* cpus, respectively.

Parameter	Euler VII — phase 1	Euler VII — phase 2	source
CPU	EPYC_7H12	EPYC_7763	Euler docs
n_{super}	2	2	UOPS Website ($= TP^{-1}$)
n_{FMA}	2	2	UOPS Website
n_{SIMD}	4	4	en.wikichip.org
f_{core}	2.6 GHz	2.45 GHz	Euler docs
n_{core}	64	64	Euler docs
$n_{sockets}$	2	2	Euler docs
n_{nodes}	292	248	Euler docs

Table 1: Parameters of the Euler VII — phase 1 and Euler VII — phase 2 nodes

Using the values from the table 1 we can calculate the peak performance of the Euler VII — phase 1 and Euler VII — phase 2 nodes.

Euler VII — phase 1:

$$\begin{aligned} p_{core} &= 2 \times 2 \times 4 \times 2.6\,\text{GHz} = 41.6\,\text{GFLOPS} \\ p_{CPU} &= 64 \times 41.6\,\text{GFLOPS} = 2662.4\,\text{GFLOPS} \\ p_{node} &= 2 \times 2662.4\,\text{GFLOPS} = 5324.8\,\text{GFLOPS} \\ p_{cluster} &= 292 \times 5324.8\,\text{GFLOPS} = \underline{1554.8\,\text{TFLOPS}} \end{aligned}$$

Euler VII — phase 2:

$$\begin{aligned} p_{core} &= 2 \times 2 \times 4 \times 2.45\,\text{GHz} = 39.2\,\text{GFLOPS} \\ p_{CPU} &= 64 \times 39.2\,\text{GFLOPS} = 2508.8\,\text{GFLOPS} \\ p_{node} &= 2 \times 2508.8\,\text{GFLOPS} = 5017.6\,\text{GFLOPS} \\ p_{cluster} &= 248 \times 5017.6\,\text{GFLOPS} = \underline{1244.4\,\text{TFLOPS}} \end{aligned}$$

2.2. Memory Hierarchies

The output of running `lscpu` and `hwloc-ls` can be found in the `memory_hierarchies` directory. As in the example in the assignment there are also two PDFs detailing the memory hierarchy of the EPYC_7H12 and EPYC_7763 nodes. Both nodes have 8 NUMA nodes, with 8 cores per NUMA node. More information on NUMA can easily be found in the Wikipedia page. Basically it means that the nodes have faster access to their specific part of the shared memory. The rest of the numbers can easily be read out of the two PDFs detailing the memory hierarchy. The main difference between the two nodes is the size of the L3 cache, which is 16MB for the EPYC_7H12 and 32MB for the EPYC_7763. The main memory size is the same for both nodes at 248GB.

2.2.1. Cache and main memory size

Cache	EPYC_7H12	EPYC_7763
L1d	32KB	32KB
L1i	32KB	32KB
L2	512KB	512KB
L3 (shared)	16MB (4 cores)	32MB (8 cores)
NUMA	31GB	31GB
Total Machine	248GB	248GB

Table 2: Cache and main memory size for both nodes

2.3. Bandwidth: STREAM benchmark

As per the STREAM benchmark documentation we need to set DSTREAM_ARRAY_SIZE to be a four times the L3 cache size.

For the EPYC_7H12 node the cache size (2) is 16MB.

$$16\text{MB} = 1.6e7\text{B}$$

So we set DSTREAM_ARRAY_SIZE to be $1.6e7 \times 4/8 + 2e6 = 10e6$ (we add a little extra as recommended).

2.4. Performance model: A simple roofline model