Università
della
Svizzera
italiana

**Institute of
Computing
CI**

High-Performance Computing Lab

Institute of Computing

Student: Paolo Deidda

Discussed with: FULL NAME

Solution for Project 1

In this project you will practice memory access optimization, performance-oriented programming, and OpenMP parallelizaton on the Rosa Cluster .

Contents

1. Rosa Warm-Up   *(5 Points)*

1.1. exercise 1

The **module system** is a utility that allows the user to dynamically manage their software environment on the Rosa HPC cluster and to load different compilers, libraries and applications in order to modify environment variagles (like PATH, LD LIBRARY PATH, MANPATH, etc.) without creating conflicts between different software versions or dependencies.

As riported in the USI resource page the module system provides several commands to manage the environment.

- `module avail` – lists all available modules (on the current system)

- `module list` – lists all currently loaded modules

- `module show` – display information about

- `module load` – loads module

- `module switch` – unloads, loads

- `module rm` – unloads module

- `module purge` – unloads all loaded modules

1.2. exercise 2

The **Slurm** (Simple Linux Utility for Resource Management) is a job scheduler for Linux clusters. Main features of Slurm are:

- Job scheduling and resource management

- Framework for starting, executing, and monitoring work (jobs) on a set of allocated nodes

- Queuing management to handle multiple users and jobs

The two main components are:

- *sulurmd*: the deamnon that runs on each compute node responsible for launching, monitoring, and terminating jobs

- *slurmctld*: the central management daemon that manages job queues and allocates resources

Main commands:

- `srun`: submit a job for execution

- `sbatch`: submit a batch job

- `squeue`: view the status of jobs in the queue

- `scancel`: cancel a job

- `salloc`: allocate resources for an interactive job

1.3. exercise 3

Here below is the code of the C program that prints "Hello World" and the information about the system where it is executed.

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
    char hostname[256];
    gethostname(hostname, sizeof(hostname));
    printf("Hello World from host: %s\n", hostname);
}
```

Listing 1: Hello World C Program - *src/1-Rosa-warm-up/hello_world.c*

We can process the script with the following command:

```
srun -N1 --time=00:01:00 ./hello_worldc > hello_worldc.out 2> hello_worldc.err
```

and we can see from the output that the program has been correctly compiled and executed on the cluster on node `icsnode22` from the *slim* partion.

```
Hello World from host: icsnode22
```

1.4. exercise 4

We can see the output of the command `sinfo` here below:

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
slim*	up	2-00:00:00	7	mix	icsnode[22,27-28,32-35]
slim*	up	2-00:00:00	12	alloc	icsnode[17-21,23-26,29-31]
slim*	up	2-00:00:00	2	idle	icsnode[36-37]
lr-slim	up	30-00:00:0	1	idle	icsnode38
gpu	up	2-00:00:00	8	mix	icsnode[05-06,08-13]
gpu	up	2-00:00:00	2	idle	icsnode[14-15]
fat	up	2-00:00:00	4	idle	icsnode[01-04]
bigMem	up	2-00:00:00	2	idle	icsnode[07,15]
debug-slim	up	4:00:00	1	idle	icsnode39
debug-gpu	up	4:00:00	1	idle	icsnode16
multi_gpu	up	2-00:00:00	2	idle	icsnode[41-42]

Listing 2: Output of line command `sinfo`

As we can see nodes are divided in partitions with names that already give us some information about their characteristics. As explained in the sbatch guide on slurm website, we can use different flags on the sbatch command to specify the partition to use with different commands.

The flag that applies to our case is:

```
sbatch --partition=fat job_script.sh
```
Submit with bigMem partition to run on nodes with very large memory:
```
sbatch --partition=bigMem job_script.sh
```
Similarly, for GPU partitions:
```
sbatch --partition=gpu job_script.sh
```
For example, we can modify the script file `slurm_job_one.sh` to specify the partition with the gpu partition with the following line:

```
#SBATCH --partition=gpu
```

After inserting the line into the file and reprocessing the script, we can see the following result:

```
Loading gcc/13.2.0−gcc−8.5.0−5hqhkwo
  Loading requirement: gcc−runtime/8.5.0−gcc−8.5.0−7fyorqa
    gmp/6.2.1−gcc−8.5.0−lrpcvy5 mpfr/4.2.1−gcc−8.5.0−ybeybcx
    mpc/1.3.1−gcc−8.5.0−cv2gjfw zlib−ng/2.1.6−gcc−8.5.0−ztbc5xt
    zstd/1.5.5−gcc−8.5.0−azepnn7
Currently Loaded Modulefiles:
 1) gcc−runtime/8.5.0−gcc−8.5.0−7fyorqa    5) zlib−ng/2.1.6−gcc−8.5.0−ztbc5xt
 2) gmp/6.2.1−gcc−8.5.0−lrpcvy5            6) zstd/1.5.5−gcc−8.5.0−azepnn7
 3) mpfr/4.2.1−gcc−8.5.0−ybeybcx           7) gcc/13.2.0−gcc−8.5.0−5hqhkwo
 4) mpc/1.3.1−gcc−8.5.0−cv2gjfw
Model name:              Intel(R) Xeon(R) CPU E5−2650 v3 @ 2.30GHz
Hello World from host: icsnode08
```

Listing 3: Output of the job script `slurm_job_one.sh` after specifying the partition

From the very last line we can assert that the job has been submitted to node `icsnode08`, and from the `sinfo` output before (2) we can see that this node belongs to the *gpu* partition instead of the *slim* partition as before (1).

1.5. exercise 5

2. Performance Characteristics   *(30 Points)*

3. Optimize Square Matrix-Matrix Multiplication   *(50 Points)*

4. Quality of the Report   *(15 Points)*