Exercises for High Performance Computing
(MA-INF 1108)
WS 2023/2024

E. Suarez, M. Wolter and B. Kostrzewa
Tutors: O. Vrapcani and N. Pillath

---

8 Network Communication

ATTENTION!: eCampus can become unavailable without previous announcement due to an urgent maintenance. Students are responsible for submitting their checklists enough ahead of time of the deadline. Submissions via Email will not be accepted unless tutors explicitly authorized it beforehand.

This exercise will be performed on the **accelerated compute nodes of the JURECA-DC** system.

WARNING: Be very conscious about compute time. **Run your jobs only in batch mode**, unless the exercise explicitly asks to run an interactive session. When you use interactive mode, open the interactive session only to run your job (not while coding or debugging). Select your time windows as short as possible, and close the interactive session immediately after finishing the run (using `scancel <jobid>`).
If you use Jupyter, run your Jupyter notebook only on the login node. **Do NOT start Jupyter on the compute node** because these long sessions consume too much compute time. Instead, to start an interactive session from Jupyter running on the login node, just open a terminal from the Jupyter launcher.

In this exercise you will be running the Ohio State University (OSU) benchmarks `with OpenMPI`. The OSU microbenchmarks is a set of tests used in HPC to evaluate the communication capabilities (latency and bandwidth) of system-wide interconnects. The OSU microbenchmarks measure the performance of various MPI operations, and allows studying:

- How good is your network
- How good is your MPI library

There are three primary types of operations:

- Point-to-point
- Collective
- One-sided

The OSU microbenchmarks are available in the website:

`http://mvapich.cse.ohio-state.edu/benchmarks/`

Read this webpage, in particular the information about the C and MPI based *host-based benchmarks*. Download the `Tarball` onto your `src` folder and untar it there.

Follow the instructions in the `README` file to configure and install the OSU benchmark. You will need to load the modules `GCC` and `OpenMPI` first. Note: to find out the path to, e.g., `mpicc`, you can use the command `which mpicc`.

Note: The installation of the OSU benchmarks will generate many files in your `src` folder. Therefore, **in this exercise you should NOT commit the `src` folder to GitHub.**

1: JURECA-DC network configuration

Check this paper `https://doi.org/10.17815/jlsrf-7-182` and/or the configuration documentation `https://apps.fz-juelich.de/jsc/hps/jureca/configuration.html` to reply these questions about the **JURECA-DC** system network on the eCampus checklist.

a) **[0.5pt]** What is the interconnect technology?

b) **[0.5pt]** What is its topology?

c) **[1pt]** What is its latency per network link?

d) **[1pt]** What is its bandwidth per network link in the standard nodes?

e) **[1pt]** How many network cards (aka Host Channel Adaptors or HCAs) are included in a standard node?

f) **[1pt]** Check with `lstopo` on the JURECA-DC standard node and find out to which NUMA domain(s) the HCA(s) is/are directly connected.

g) **[1pt]** What is its bandwidth per network link in the JURECA-DC accelerated nodes?

h) **[1pt]** How many HCAs are included in a GPU node?

i) **[1pt]** Check with `lstopo` on the JURECA-DC accelerated node and find out to which NUMA domain(s) the HCA(s) is/are directly connected.

2: OSU point-to-point benchmark

Read the OSU documentation (README) to understand what a point-to-point operation consists of. You will be using the executables contained in the folder `c/MPI/pt2pt/standard`, created when you built the OSU benchmarks.

Do the following exercises on `JURECA-DC`, using the `dc-cpu-devel` queue unless otherwise explicitly stated. Save your output `.txt` and `.png` files on your solutions folder.

a) **[1pt]** Run `osu_latency` selecting 2 nodes and 2 tasks (1 task per node). Pin the process to specific core numbers between which the point-to-point communication should take place. To do this for core number 0, use the `srun` option `--cpu-bind=map_cpu:0,0`. The result of the run should be a text file containing the network latency (in μsec) obtained for increasing packet sizes (in bits). Save this output file as `osu_p2p_lat_c0.txt`.

b) **[1pt]** Run in the same manner the `osu_bw`. This should produce a text file containing the network bandwidth (in MB/sec) obtained for increasing packet sizes (in bits). Save the output file as `osu_p2p_bw_c0.txt`.

c) **[2pt]** Plot into a *double-plot* (two plots side-by-side). Left: network latency [μsec] vs. packet size [bits], including an horizontal line for the theoretical latency of the hardware (as obtained from documentation on exercise 8.1). Right: network bandwidth [MB/s] vs. packet size [bits], including an horizontal line for the theoretical bandwidth as of documentation. It is recommended to plot these graphs in a log-log scale, for better visibility. Save the plot as `osu_p2p_c0.png`.

d) **[3pt]** Redo the previous 3 exercises, but choosing now core number 112. Create the *double-plot* displaying both the previous and the current results, so that you can easily compare, following the style recommendations from before. Save the new plot as `osu_p2p_c112.png`.

e) **[1pt]** Reply in the eCampus checklist following question on Latency: which of the two cases (core-0 vs. core-112) has a lower latency and by how much? (Hint: for latency analysis the important region on which you should focus is for small messages (low packet sizes).

f) **[1pt]** Reply in the eCampus checklist following question on Bandwidth: which of the two cases (core-0 vs. core-112) has a higher bandwidth and by how much? (Hint: for bandwidth analysis the important region is for large messages (large packet sizes).

g) **[2pt]** JURECA-DC GPU nodes: Redo the plot from 7.2.d on an accelerated node of JURECA-DC (queue `dc-gpu-devel`), to compare the network latency and bandwidth from core-0 and core-112. Save the new plot as `osu_p2p_gpu_c112.png`.

h) **[1pt]** JURECA-DC GPU nodes: Reply in the eCampus checklist following question on Latency: which of the two cases (core-0 vs. core-112) has a lower latency and by how much?

i) **[1pt]** JURECA-DC GPU nodes: Reply in the eCampus checklist following question on Bandwidth: which of the two cases (core-0 vs. core-112) has a higher bandwidth and by how much?

Commit your solutions to the GitHub Classroom.
If you have used Jupyter, close your Jupyter session and stop JupyterLabs.