# Assignment 1

Gabriel Mitterrutzner (11832162)
gabriel.mitterrutzner@student.uibk.ac.at

Philipp Gritsch (11702197)
p.gritsch@student.uibk.ac.at

14th October 2022

## Exercise 1

1. `--ntasks`
2. `--ntasks-per-node`
3. `--ntasks-per-socket`
4. `--contiguous`
5. `--exclusive`

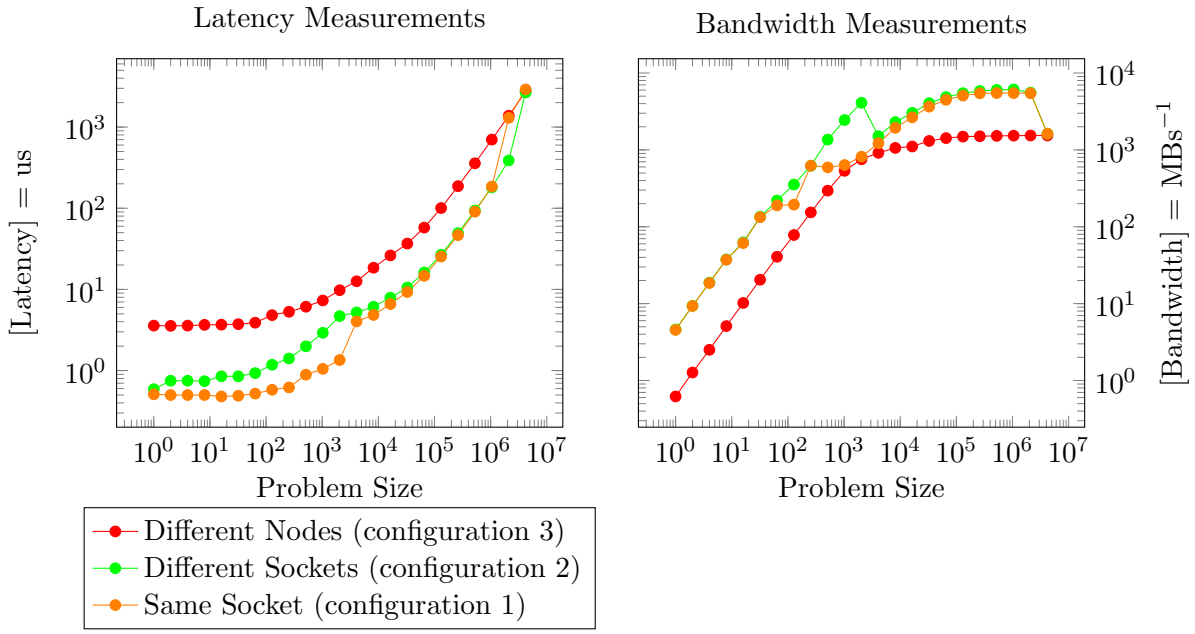
With parameters 1, 2 and 3, one is able to configure the task distribution across the system, without having to specify details, like the exact worker where to run one specific task. There are still more options, which allow more sophisticated control on which computing resources to utilize, like `--nodelist` or `--partition`, but as LCC2 is a homogeneous system, there is no special reason to specify which nodes or partitions to use.
Parameter 4 gives us the option to use nodes, which are locally coherent, which might be important to create reliable and reproducible measurements, as we expect larger latency and smaller bandwidth between nodes which are spatially farther apart. Additionally, using nodes which are not directly close to each other, might require additional network infrastructure, which might also be used by other tasks, possibly leading to I/O congestion.
To further enforce reproducible measurements, one might want to ensure that no other tasks run on the same node, which would definitely lead to different measurements, based on the workers utilization during runtime. Using parameter 5, one ensures, that a submitted task runs exclusively on its nodes.

## Exercise 2

Both benchmarks `osu_latency` and `osu_bw` were executed 10x with 3 different configurations:

1. `--ntasks-per-node=2 --ntasks-per-socket=2` (all tasks on 1 single socket)
2. `--ntasks-per-node=2 --ntasks-per-socket=1` (all tasks on 1 single node, but on 2 sockets)
3. `--ntasks-per-node=1` (all tasks on different nodes)

Depicted below are our measurements, averaged across 10 executions. Furthermore all results showed a maximum difference of 2 % between minimum and maximum values. Thus our results are very stable.




Our results for running the latency benchmark (`osu_latency`) resemble our expectations. Latency should be larger for longer communication distances. This is clearly the case when comparing configurations 1, 2 and 3.

For the bandwidth benchmark (`osu_bw`) we found, at least for us, some odd results. One would expect, that the bandwidth decreases, as the networking distance increases. Looking at our measurements and comparing the measurements between configuration 2 and 3 we can confirm this theory.
However comparing between the configurations 1 and 2 we see approx. the same results (with slightly smaller bandwidths for configuration 1), which, in our opinion is unexpected, as intra-socket communication should definitely offer higher bandwidth than socket-to-socket communication. This oddity was confirmed by running the benchmark multiple times, with no differences in the observations.
The documentation of `osu_bw` [1] doesn't add much more clarity:

> `The bandwidth tests are carried out by having the sender sending out a fixed number (equal to the window size) of back-to-back messages to the receiver and then waiting for a reply from the receiver. The receiver sends the reply only after receiving all these messages. This process is repeated for several iterations and the bandwidth is calculated based on the elapsed time (from the time sender sends the first message until the time it receives the reply back from the receiver) and the number of bytes sent by the sender. The objective of this bandwidth test is to determine the maximum sustained date rate that can be achieved at the network level. Thus, non-blocking version of MPI functions (MPI_Isend and MPI_Irecv) are used in the test.`

As the benchmark, by design, should show the maximum sustained data rate, which should result in our expectations, we can only guess, why this isn't the case. One such guess could be, that the `OpenMPI` implementation on LCC2 isn't properly configured or optimized.

## References

[1] Dhabaleswar Panda. MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot. `https://mvapich.cse.ohio-state.edu/static/media/mvapich/README-OMB.txt`, 2022. Accessed: 2022-10-14.