**Parallel Systems - Assignment 1**

**Exercise 1:**

submission script that starts an arbitrary executable: */assignment1/job.script*

5 most important parameters:

-pe parallel-environment number-of-slots

🡪 essential parameter for parallel processsing

🡪 use *openmpi-Xperhost* to specify how many instances will be used per node

🡪 use *openmpi-fillup* to use the least number of nodes possible

-l

🡪 h\_rt=[hours:minutes:]seconds -- stop job if it takes too long

* h\_vmem=size[M|G] – limit per slot memory size

-hold\_jid job-id

🡪 wait for job with job-id to finish

-w v

🡪 check for syntax error in the job. Don’t use this parameter

-t 1-n

🡪 only for trivial parallelization (e.g. with no runtime communication between processes)

You need to specify the *-pe* parameter to run a program in parallel and tell the system the number of total slot as well as how many slots should be handled by each node/socket (*openmpi-Xperhost/openmpi-fillup)*

**Exercise 2:**

Tables & Figures of the OSU Benchmark: */assignment1/results/benchmark\_results.xlsx*

You could verify the rank placement without looking at the performance by executing the */bin/hostname* (or any other executable that gives info about the node/cpu) at the start of each benchmark.

As expected, out **latency** performance results show that two processes communicating between two different nodes takes longer than it would on processes on the same node and communication on the same socket is the fastest. The latency imbalance of the different ranks decreases with an increase in size. One interesting observation was that the latency on data sizes >2GB was best on two different nodes (maybe due to memory limitations for processes on the same node)!

**Bandwidth** performance showed similar results. Node to node communications was the slowest, and both one node communications where on par until a data size of 2MB. For larger data sizes the bandwidth performance dropped significantly (especially for the same socket benchmark) and dropped a second time after 2GB (possibly due to cache and memory limitations)!

After multiple benchmark runs, the results seem fairly stable except for occasional latency/bandwidth spikes for smaller data sizes within the same benchmark. Only during a single run did we get significantly higher latencies for the *different cores/same socket* benchmark. But that appears to be an outlier.