Assignment 1

Exercise 1

Tasks

a) Study how to submit jobs in SGE, how to check their state and how to cancel them.

Submission: qsub name_of_script

Deletion: qdel job_id

Status: qstat [options] (-u name or/and -g c: Display cluster queue summary)

b) Prepare a submission script that starts an arbitrary executable, e.g.

See appendix.

- In your opinion, what are the 5 most important parameters available when submitting a job and why? What are possible settings of these parameters, and what effect do they have?

-pe → to choose the parallel environment

-**l h_rt=** ***[hours:minutes:]seconds*** → requested real time (wallclock time from start to termination; the default (=maximum) depends on the system and, if applicable, the specified queue.

Team: Eduard Frankford, Ralf Pernecker, Linus Wald

-w v check whether the syntax of the job is okay (do not submit the job)

-l h_stack =*size*[m|g] request a **per slot** stack size limit of *size* bytes / megabytes / gigabytes. This parameter is typically needed if your programs allocate large amounts of memory on the stack (e.g. large dynamically sized local variables in Fortran programs).

-M *email address*

c) How do you run your program in parallel? What environment setup is required?

#$ -pe openmpi-Xperhost Y

This command sets up an openMPI environment with X processes per Host and Y processes in total. The number of Nodes results from Y/X.

Additionally, for openMPI we need to modify the environment by loading the module:

module load openmpi/4.0.3

For execution use the command:

mpiexec -n X Y

Where X is the number of processes and Y the command or binary to execute.

Exercise 2

Tasks

a) Modify your experiment such that the 2 MPI ranks are placed o different cores of the same socket,

#$ -pe openmpi-2perhost 2
mpiexec -n 2 binary

b) different sockets of the same node

#$ -pe openmpi-2perhost 2
mpiexec -map-by socket -n 2 binary

c) different nodes

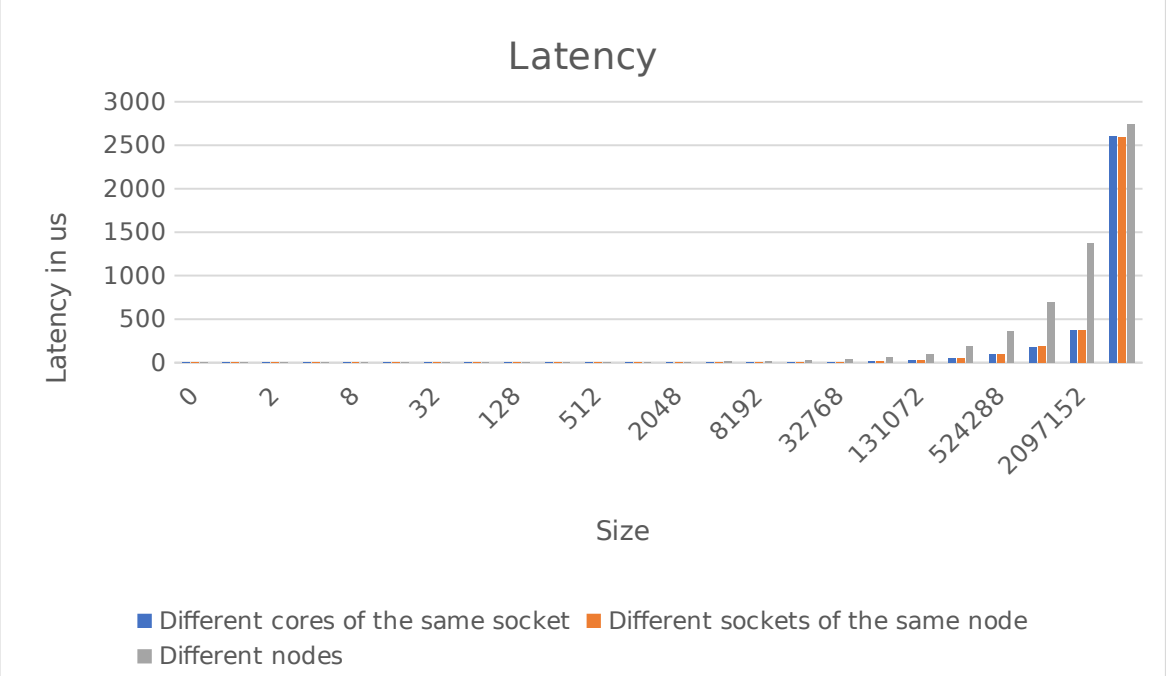
#$ -pe openmpi-1perhost 2

d) Amend your table and figures to include these additional measurements. What effects can you observe? How can you verify rank placement without looking at performance?

Team: Eduard Frankford, Ralf Pernecker, Linus Wald

Latency (us):

Size	Different cores of the same socket	Different sockets of the same node	Different nodes
0	0,45	0,45	3,51
1	0,49	0,70	3,57
2	0,49	0,74	3,58
4	0,72	0,73	3,57
8	0,72	0,73	3,68
16	0,84	0,84	3,71
32	0,84	0,84	3,76
64	0,93	0,94	3,89
128	1,21	1,22	4,81
256	1,43	1,45	5,21
512	2,11	2,18	5,90
1024	2,99	3,05	7,47
2048	4,65	4,71	9,74
4096	4,80	4,82	12,56
8192	5,68	5,67	18,50
16384	7,49	7,48	26,25
32768	10,19	10,15	36,71
65536	15,78	15,60	57,69
131072	26,78	26,35	100,64
262144	48,51	47,76	187,09
524288	91,95	90,66	357,78
1048576	179,00	183,81	698,07
2097152	367,40	376,41	1377,89
4194304	2606,74	2590,96	2737,45

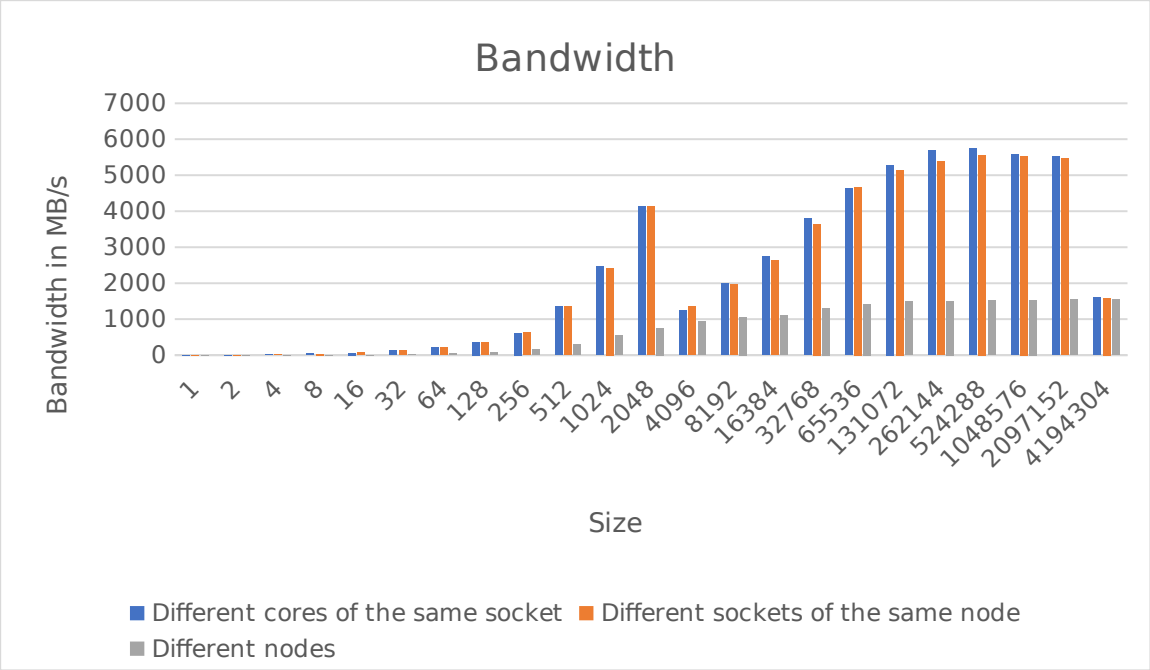


Team: Eduard Frankford, Ralf Pernecker, Linus Wald

Bandwidth (MB/s):

Size	Different cores of the same socket	Different sockets of the same node	Different nodes
1	4.66	4,53	0,63
2	9,47	9,05	1,27
4	18,85	18,26	2,53
8	38,25	33,43	5,09
16	64,61	66,64	10,24
32	134,53	127,63	20,46
64	220,53	217,81	40,71
128	365,43	360,55	77,93
256	614,53	625,24	153,28
512	1355,00	1350,85	294,88
1024	2469,18	2426,56	536,41
2048	4132,54	4120,68	753,16
4096	1252,10	1349,09	927,11
8192	1996,76	1970,51	1065,59
16384	2737,98	2641,10	1107,89
32768	3792,61	3648,49	1307,63
65536	4623,24	4669,65	1424,60
131072	5285,86	5147,05	1485,24
262144	5689,29	5380,36	1504,24
524288	5733,05	5562,37	1521,90
1048576	5580,83	5538,08	1531,88
2097152	5526,06	5459,85	1538,26
4194304	1609,66	1589,22	1540,01

Team: Eduard Frankford, Ralf Pernecker, Linus Wald



-display-devel-map, --display-devel-map

Displays a more detailed table showing the mapped location of each process prior to launch.

- How stable are the measurements when running the experiments multiple times?

Very stable only differ very little

Conclusion:

Team: Eduard Frankford, Ralf Pernecker, Linus Wald

Latency benchmark: As expected, the configuration using two cores from the same socket is the fastest solution overall. Interestingly at the size of 2097152 the “different socket same node” configuration is noticeably better than the other ones. This is probably due to the low number of executions. Overall, the “different node” configuration is the slowest, simply because the data must be transferred across the network, which results in weaker performance. However, when the size increases the other variants have about the same latency as the different node configuration. This is probably because the other variants exceed a memory limit. In general, the measurements were quite stable throughout the benchmark and didn't alternate very much.

Bandwidth benchmark: In this benchmark the "different cores same socket" and "different sockets same node" configurations are completely the same. They just slightly differ, but not very noticeable. In terms of stability the benchmark delivers the same results as the latency benchmark. The measurements were again quite stable throughout the benchmark. At the size 1096 however, the benchmark shows a decreasing spike, which decreases as problem size rises again. This performance drop happens again at the size of 4194304. Again, this might be a memory limit (probably exceeding cache size) leading to the performance loss.

Team: Eduard Frankford, Ralf Pernecker, Linus Wald