Assignment 7: MPI - Master Worker

The purpose of this assignment is for you to learn more about

- implementing a static workload partitioning scheme in MPI,
- implementing a dynamic workload partitioning scheme in MPI (here Master-Worker).

As usual all time measurements are to be performed on the cluster.

1 Preliminary

To be able to compile and run an MPI program on mamba, you need to add the line `module load openmpi` at the end of the file `.bashrc` located in the home directory of your account on mamba.

To compile an MPI application, use the `mpicc` compiler in C and the `mpicxx` compiler in C++. They also serve as linker. To run an MPI application using 19 processes, just run `mpirun -n 19 ./myprogram`. But you will need to have a proper node allocation first.

To queue an MPI job on mamba, you will need to specify how many nodes and how many cores per node you plan on using. You could use `qsub -l nodes=2:ppn=16` to request two nodes with 16 cores each, or `qsub -l procs=32` if you only care about having 32 cores independently of where they are. You can also use `qsub -l nodes=4:ppn=4`, to have 4 cores from 4 different nodes. There is currently a cap on mamba that prevents you from requiring more than 32 processes in total.

Question: Write a simple MPI program that start 32 MPI processes (split on two and then on four nodes), and where each program report its MPI rank ID and the name of the machine it is running on (use `gethostname`).

Question: Run it on mamba and verify it runs on two different machines.

2 Numerical Integration : static

Question: Adapt the numerical integration application to make it use MPI in a simple way. The first MPI process should take the first N/P iterations of the loop, the second should take the next N/P iterations of the loop, etc.. The partial integration should be accumulated on rank 0 so that it can print the correct answer.

Question: Run and time that program on mamba using 1, 2, 4, 8, 16, 32 cores. Use the parameters (intensity, number of points, and granularity) from the OpenMP-loop assignment. And plot speedup charts.

3 Numerical Integration : Master-Worker

A master-worker system enables dynamic scheduling of applications. The idea is that one of the MPI processes (usually rank 0) will be responsible for giving work to the other MPI processes and collect results.

Usually the master process starts by sending one chunk of work to all the workers and when one of the worker provides the result of that chunk of work, the master process will send a new chunk of work to the worker that just completed the work.

The workers will wait for a chunk of work to perform, perform that chunk of work, and return the result to the master node.

Pay attention that the master node needs to notify the worker nodes when there is no more work to perform so the workers can quit gracefully.

Question: Adapt the numerical integration to make it schedule the calculation using a master-worker system.

Question: Run and time that program on mamba using 1, 2, 4, 8, 16, 32 cores. Use the parameters (intensity, number of points, and granularity) from the OpenMP-loop assignment. And plot speedup charts.

4 Numerical Integration : Advanced Master-Worker

One of the issue with such a master-worker implementation is that the worker process will be idle until the master provides the next chunk. To avoid this, it is common that the master will give more than a single chunk of work (say, 3 chunks of work) to each worker. That way, when a worker sends a result to the master node, the worker still has some chunks of work (here, 2 chunks) to perform.

To implement this you will need to use the non-blocking MPI functions `MPI_Isend`, `MPI_Irecv`. And also, using functions that wait on completion of some non-blocking operation : `MPI_Wait` and variants.

Question: Adapt the numerical integration code to use advanced scheduling.

Question: Run and time that program on mamba using 1, 2, 4, 8, 16, 32 cores. Use the parameters (intensity, number of points, and granularity) from the OpenMP-loop assignment. And plot speedup charts.