HPCE 2016 Coursework 6

Yubo Zhi (`yz4116`), Jiayang Sun (`js11815`)

Analysis and parallelism

The main iteration loop

4 things are done in each iteration. Every edges (communication channels) and nodes (computation cores) have their statistics collected, then simulated by 1 step each iteration. They are separated to 4 independent loops firstly.

Both of the simulation step loop iterations are modified to be completely independent. Therefore it can be easily parallelised.

The statistics collections are all accumulations. Therefore it is possible to be parallelised by using map-reduce technique. For better multi-core utilisation, the collection is done in series, if there are less than 64 items. This value was determined by experimenting.

Only the node statistic collection is made parallel, and combined with node simulations, to save cost on task creation.

The output iteration

The above parallelism works well for rectangle topology, but neither hex lattices nor meshes benefits much from the parallelisation.

By analysing through interrupting program multiple times during run-time, then look at call stacks (the same approach as a sampling profiler), we found the output rendering loop is taking most of the time for these 2 topologies. In particular, the `find_closest_device` function.

A quick analysis reveals the function `renderSlice` that is calling `find_closest_device` , has a 2D iteration space that is output independent, therefore can be made parallel.

Results

Topology	Reference	Parallelised	Speed up
Rect 128	10.393s	4.942s	2.10x
Hex	4m 9.435s	47.249s	5.28x
Mesh	45.289s	9.346s	4.85x

Future works

OpenCL GPU acceleration can be used to parallelise the output independent iterations, and realise map-reduce technique, to potentially gain more speed up for large scale problems. Using local work-groups and local memories inside the GPU may also be beneficial, as each computation core only need to communicate with a few nearby cores, without global memory accesses.

However, due to time limit of this coursework, we do not have enough time for OpenCL implementations, which involves complicated kernel setup, complicated host and device memory structure optimisation and transfers.