# Masterpraktikum Scientific Computing

**High Performance Computing**
**Project Optimization strategy**

**Session 5: Project**

**Author**
Mantosh Kumar (Matriculation number : 03662915)

**Course of Study:** Master of Science Informatics

## Optimization strategy

To detect bottleneck of simulation, we can divide simulation into two phases, initial phase and running phase. Initial phase is a phase initializing simulation, creating agent objects and locating agents into grid or space. Running phase is a main part of the simulation, iteration of searching neighbor agents and migration to other location.

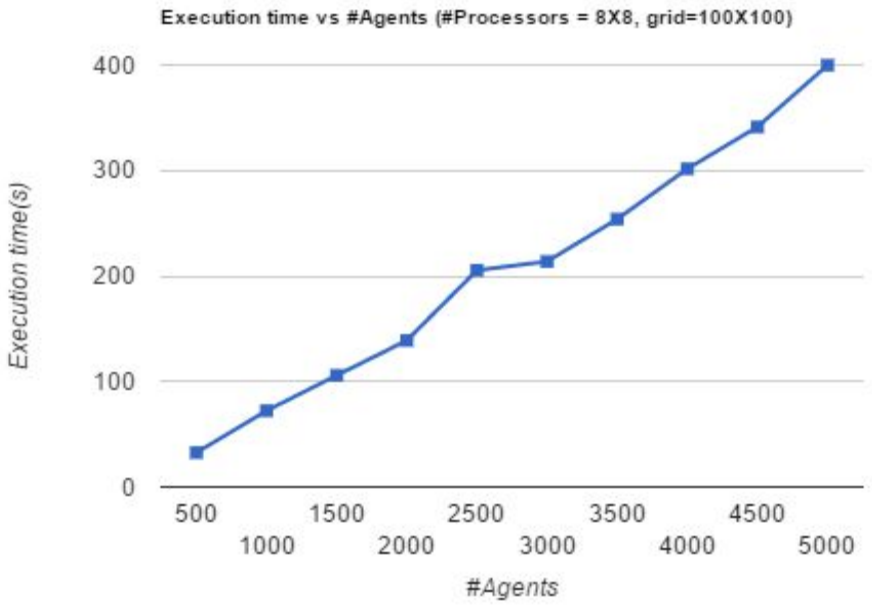



**Figure 1**

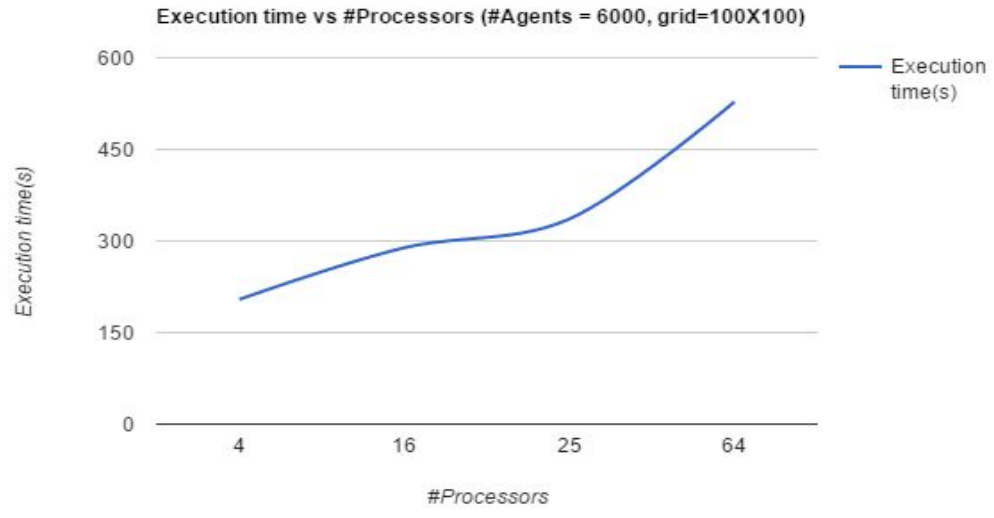



**Figure 2**

Figure 1 and 2 show transition of execution time. Figure 1 shows that execution time increases as number of agents increases. While figure 2 shows that the execution does not decrease even when we increase the number of cores. Observing this behaviour it can be safely deduced that execution time scales almost linearly with the number of cores for mainly independent agents, whereas speedup drops significantly in case of highly interdependent agents.

With the help of scalasca, we found out that the parts for giving agents coordinates and locating them on grid or space takes most execution time. For this part, Repast HPC API is used for these parts in original source code. It is expected that execution would be more efficient by using C++ standard library instead of Repast API.

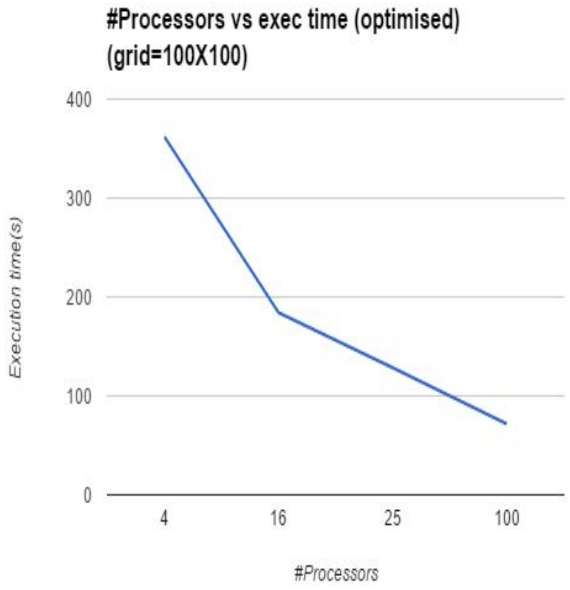
We took the input from HDF5 files which provided the application with a rich interface for I/O access to organize the data and store it efficiently. With its collective I/O feature, it leverages the data movement around between processes. With C++ standard library it is possible to locate the agent on the plane.

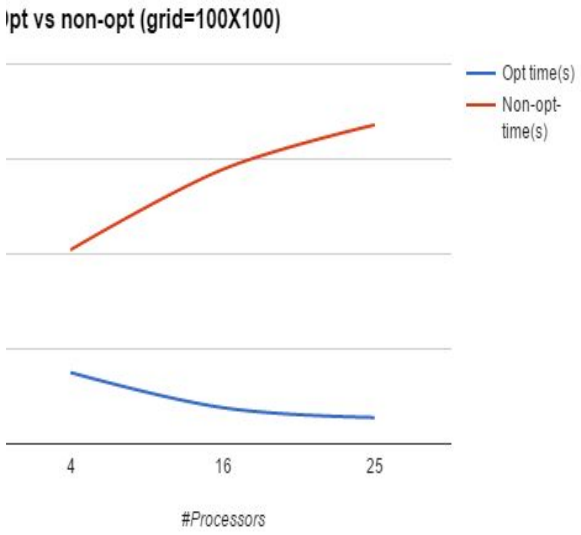
Most Expensive jobs:

- Giving agents coordinates
- Locating agents on grid or space

Optimisation Strategy

- Use HDF5 data as input
- use C++ standard library instead of Repast API as much as possible
- Try to design less interdependent agents




## measurements for performance (FLOPS)

With scalasca, PAPI is used to record hardware counters.
$ export SCOREP_METRIC_PAPI=PAPI_FP_INS,PAPI_FP_OPS

But on linux cluster though PAPI is available but it does not support any of the possible 108 events.
$ papi_avail
……...Of 108 possible events, 0 are available, of which 0 are derived.
This is why it is not possible to calculate floating point operations for the application.