# Conclusion and Future Work

Agent-based simulations are becoming ever more demanding and increasing is size. As of this writing there is no framework that supports parallel agent-based simulations. MUSE is purposed as a scalable and efficient solution. Railsback explored several agent-based simulation frameworks and described criteria for what a great agent-based framework should have at the very least. Among these criteria, documentation and ease of use was noted as being most important. After the development of MUSE the public API section 4 showed just how easy developing with MUSE is, this included the use of the MUSE code generator. Empirical evaluations in section 5.1 showed that MUSE achieved super linear speedup and excellent efficiency. These facts combined demonstrated MUSE as a very scalable framework. MUSE was benchmarked against a non-agent-based parallel framework (WARPED) and against a serial agent-based framework (MASON). Benchmarking against WARPED showed how efficient MUSE was because it needed far fewer nodes to achieve similar runtimes as WARPED, the results can be viewed in section 5.2. The next benchmark against MASON showed some of the limitations a distributed framework and of the TimeWarp protocol. The first experiment was a direct comparison of the PHOLD simulation on one compute node. MASON serial implementation of PHOLD proved to outperform MUSE on a single compute node. However, the second experiment showed MUSE scalability and as the model gets larger MUSE will always come out on top because MUSE can effectively and efficiently make use of more compute nodes. Moreover the benchmark showed that for a PHOLD simulation with a size of 512 x 512 it took at least 7 compute nodes to outperform MASON best serial time. There are many things that can be considered as future work.

In the MUSE core, there was no real optimization done. Like MASON, if MUSE was to use custom created data structures for the storage of events and states MUSE should be able to increase the performance in the one compute node benchmark. So a future work can be to learn all the optimization done in MASON and port those tricks over to MUSE. Making the single node benchmark as close as possible to that runtime of MASON will make MUSE even more scalable and efficient. Another future that is important is to add an API to handle visualization. Ways to visualize 3D and 2D simulations would be a great benefit to the overall framework. Simulations that are very popular with agent-based frameworks are spatial based simulations. Hence, a library to optimize communication between agents by wisely moving frequently communicating agents to the same compute node would be a great library.