

MASS JAVA Benchmarking

Winter 2023 Term Report

Anirudh Potturi

In Partial Fulfillment of the Requirements

For the Degree of


Masters of Science

Under Guidance of Dr. Munehiro Fukuda

University of Washington

©March, 2023

Anirudh Potturi

 <https://orcid.org/0000-0002-9270-9628>

Contents

List of Figures	4
1 Introduction	5
2 Benchmark Applications	6
2.1 Closest Pair of Points	6
2.2 KDTree based Range Search	6
2.3 Convex Hull	8
2.4 Largest Empty Circle	9
2.5 Euclidean Shortest Path	9
3 International Conference on Agents and Artificial Intelligence 2023	11
4 Tool Development and Application Fixes	13
4.1 Nodes XML File Generator	13
4.2 Benchmark Runner Application for MASS Java	13
4.3 Convex Hull	14
4.4 Largest Empty Circle	14
5 Results and Analysis	15
5.1 KDTree Based Range Search	15
5.2 Closest Pair of Points	16
References	17
Appendices	18

LIST OF FIGURES

2.1	Neighborhood Patterns [1]	7
2.2	Closest Pair of Points	7
2.3	Range Search Data Points	8
2.4	KDTree based Range Search	8
2.5	Convex Hull [2]	9
2.6	Largest Empty Circle [3]	10
2.7	Euclidean Shortest Path [4]	10

Chapter 1

Introduction

The purpose of this paper is to provide a summary of the work performed in the Winter Quarter of 2023. This quarter's goal was to bring together applications in Computational Geometry together and perform additional benchmarkings. A total of Five applications are lined up. They are Closest Pair of Points, Largest Empty Circles, Point Location, Range Search and Euclidean Shortest Space. For each of these applications, we gathered the work of former and current students, some of which already existed within the MASS Java Application Repository at the beginning of this Quarter. The role of this quarter was to focus on setting up the benchmark executions of these applications. This includes rigorous testing of each application with 1 node and up to 24 nodes. Performance of the MASS library is tested using different sizes of data sets over multiple node configurations. Each test is conducted 3 times and finally used to compute average times. Lastly, upon completion of previous quarter's work, the paper submitted to the International Conference on Agents and Artificial Intelligence (ICAART) 2023 conference was accepted into the publishing and the paper was to be presented in Lisbon, Portugal in the month of February.

Chapter 2

Benchmark Applications

The following applications have been lined up towards a submission to The Journal of Supercomputing. All applications have been developed by former or current students, each using MapReduce, Apache Spark and MASS Java libraries. This section introduces the applications and explains the working of the MASS versions of each of the application.

2.1 Closest Pair of Points

The closest pair of points application was developed using one of the agent auto migration feature built into MASS Java Library. In this application, the data is distributed over a continuous geometric space and agents start migrating from the data points. This application tests the MigratePropagateRipple functionality in MASS where agents migrate to the Von Neumann and Moore neighboring places (as seen in Fig. 2.1) by spawning new child agents. The migration stops when agents originating from two different points collide with one another (as seen in Fig. 2.2).

2.2 KDTree based Range Search

This application is based on a multi dimensional space over which data is distributed (from Fig. 2.3). The points are then split into two subsets each of roughly the same size. While one subset contains all the points smaller than the splitting value, the other contains the points larger than the splitting value. This is a recursive process in which data is split into two halves at the median. Through this process of division,

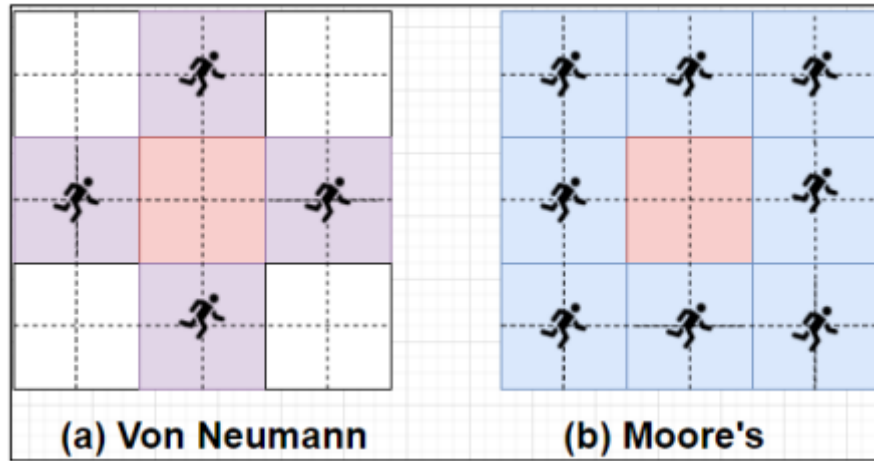


Figure 2.1: Neighborhood Patterns [1]

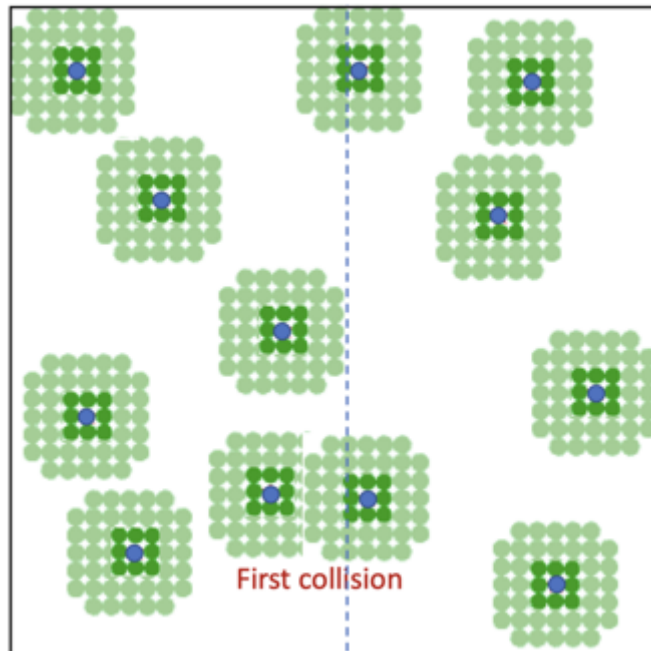


Figure 2.2: Closest Pair of Points

From Fig. 2.4 a tree is formed with a left and a right branch where nodes of the tree represent the data. The `MigratePropagateTree` method from MASS Library is used to automate the agent migration over this tree. The agents work towards finding all the points that exists within a user specified range.

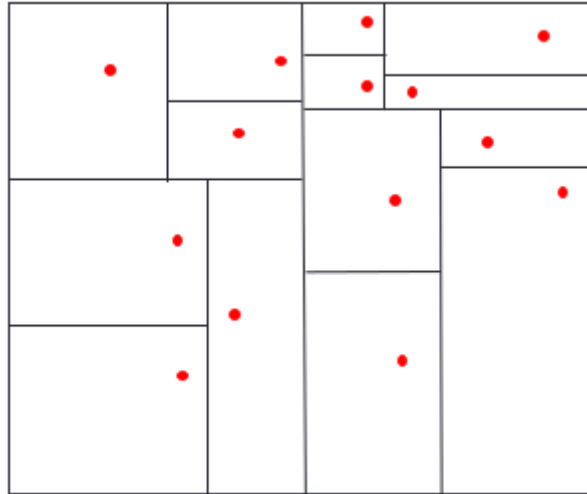


Figure 2.3: Range Search Data Points

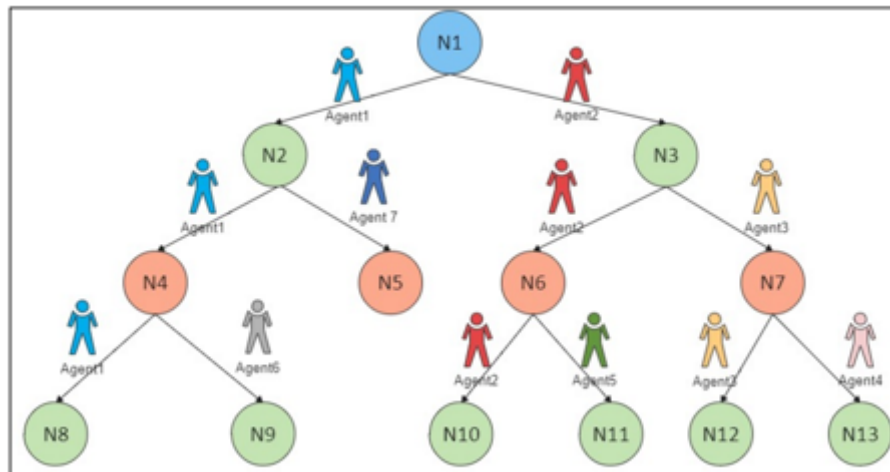


Figure 2.4: KDTree based Range Search

2.3 Convex Hull

This application is a 2-Dimensional space based problem wherein a set of points are distributed. The goal is to identify the smallest convex polygon that encloses all the data points. The spatial features in MASS are leveraged to apply agents to squeeze inwards, starting from the boundary of the grid. From Fig. 2.5 This way, as agents move inwards, they identify the outermost points, thus identifying candidate points. Currently, no features exist in the MASS library that support this style of agent migration.

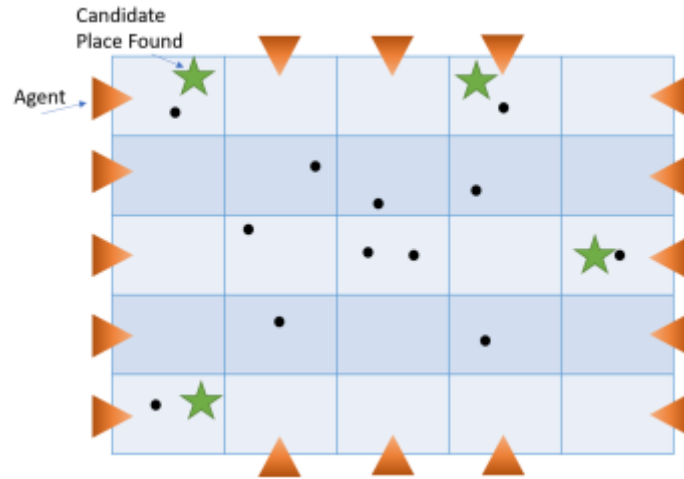


Figure 2.5: Convex Hull [2]

2.4 Largest Empty Circle

Similar to a Convex Hull or Voronoi Diagram problem, the Largest Empty Circle problem is defined over a set of points. The goal of this problem is to identify the largest circle that contains the maximum number of points from the data set (see Fig. 2.6). This problem is based on Voronoi Diagram, a problem that has been explored by the DSLab. Given the Voronoi vertices, one can simply identify the largest circle around the vertices. With the use of features in MASS Java like SpaceAgents and SpacePlace components, the Voronoi site and vertex points are mapped to places. Agents explore and identify the farthest pair of the data point pairs by migrating to Moore and Von Neumann neighbor places.

2.5 Euclidean Shortest Path

Given a 2-Dimensional area, a distribution of objects are scattered that act as obstacles. The motivation behind the application is to identify the shortest path from a source to destination point. What makes this application intuitive using MASS is through the use of Point Location, another Computational Geometry Application is used to find the area of the said obstacles. Next, agents propagate from the source in the Von Neumann and Moore neighbor patterns. From Fig. 2.7 On encountering an obstacle

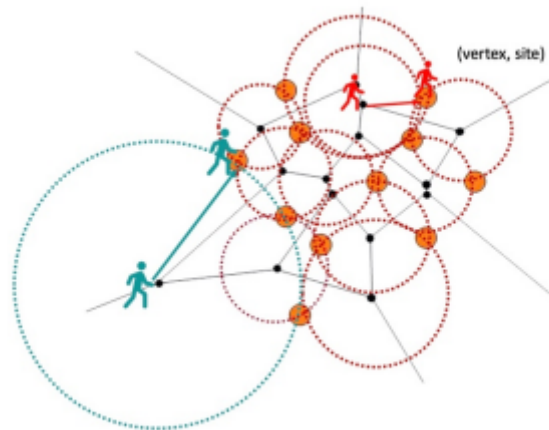


Figure 2.6: Largest Empty Circle [3]

like the red box, agents move along the edge of the obstacle, thus making their way finally reaching the destination vertex.

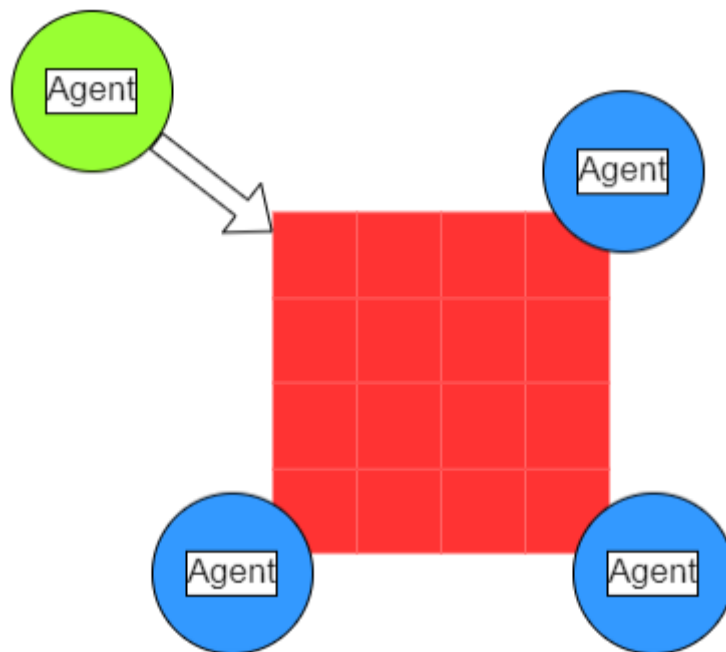


Figure 2.7: Euclidean Shortest Path [4]

Chapter 3

International Conference on Agents and Artificial Intelligence 2023

The ICAART 2023 conference took place in Lisbon, Portugal this year. I had the opportunity to travel to the conference and present our work. The conference took place from February 22nd to the 24th, 2023. With the motivation of our work driven towards relieving a programmer from describing fine-grained code to describe agents, the results presented were in favor. We were able to show that MASS in fact requires less programming when compared to its competitor Agent Based Modelling libraries like Repast Symphony. We were also able to compare and contrast MASS with more libraries like NetLogo, RepastHPC and FLAME for use in the field of Data Sciences and Big Data.

We had the opportunity to introduce the MASS Library to researchers from all over the world, its functionality, technical specifications and strengths. We explained how we used the library in developing application like Breadth First Search, Tringle Counting, KDTree Based Range Search and Closest Pair of Points to name a few.

We were able to introduce the Automated Agent Migration Techniques introduced into MASS Library while also introducing the use cases of these techniques. By explaining the use of one such migration technique in an application called TriangleCounting, we demonstrated the ease of developing an application that leverages the in built feature. A user need not describe the pattern of migration of an agent to count the number of triangles in a graph.

We were also able to show that with the improvements introduces in MASS like ad-

dition of features to improve Agent and Spatial Management, Support for tree construction, Reduction of thread control and inter-cluster communication overheads helped in supporting application while working with large data sets.

By demonstrating the use of all eight agent migration techniques in the applications we worked with for this research paper, we were also able to compare the programmability of these applications using a library like Repast Symphony. In essence, we contrasted the programmability of Legacy MASS vs New MASS vs Repast simphony. This further solidified our motivation since New MASS had major desirable metrics programmers would look for.

Towards the end, we were able to conclude that Automation of Agent Migration enhances the execution performance and also improves programmability [5]. We did this by comparing the execution performance and lines of code metrics of what we term as Legacy MASS and New MASS for each of the applications.

Chapter 4

Tool Development and Application Fixes

4.1 Nodes XML File Generator

An additional yet useful tool that will be helpful is the NodeFileGen Tool. The tool was initially not part of the plan but seemed to be useful for new users to MASS who may not have complete knowledge of how to setup their nodes.xml file. MASS application users and developers require this file to specify the nodes/machines that will run the application. It will be used to list the master and worker nodes. In essence, it is a file that lists all the machines in the cluster. The principle of this tool is simple. When called from within a MASS Application folder, the tool will generate a nodes.xml file for the application instantly. The user passes their Username which in our case is the Netid and their assigned port number. They will pass a third argument which is the total number of nodes they require in their cluster. This includes the master node as well. By default, the master node is decided as 'cssmpilh.uwb.edu' by the tool. However, users may modify the code in the tool as per their requirements. This tool utilizes the Javax XML Package to generate the XML document.

4.2 Benchmark Runner Application for MASS Java

The Benchmark Runner Application for MASS Java is a tool that will automate the execution of all applications that have been lined up. The application will work on the principle of multi-threading. Threads will be created based on the number of applications to be run at the same time. A thread will be responsible for running a shell script which in turn automates the execution of an application. The Benchmark

runner application is currently under development. The complete application will be ready when all the benchmark applications have been thoroughly tested and their run scripts are prepared. This quarter, work has progressed towards setting up the scripts for applications that were available at the start. With changes in applications used in this automated execution tool, the new applications are undergoing testing.

Although the plan is to keep this tool separate from the NodeFileGen Tool, similar logic from the tool can be integrated into this benchmark runner tool as well for automation of all XML file generations before beginning benchmark executions. With the structure of this tool ready, completion of the shell scripts will conclude our work on this tool.

4.3 Convex Hull

This application requires the creation of a very large number of places. Depending on the data, we must create an $N \times N$ grid where N is the maximum value from both X and Y coordinate lists. When working with a data set with 500,000 and 1,000,000 points, the maximum of these coordinates is large enough that we would need 100,000,000 places. This is not a very practical option. Instead we had to choose a data set with a limit on the maximum of the values. Another issue that arose was with agent migration in the application. The agents at all times would try to migrate to the same place. This meant that the migration was not taking place as intended. With these issues fixed, the application is ready for testing. However, the creation of a large number of places is still time consuming.

4.4 Largest Empty Circle

The current application of Largest empty circles (MASS) does not get instantiated with the data points as intended. The file passed as input is not being processed into an array of data points.

Chapter 5

Results and Analysis

With benchmarks being performed of each application in both MASS Java and MapReduce, while some also in Apache Spark, this section highlights the execution performance of applications developed using MASS Java.

5.1 KDTree Based Range Search

The KDTree Based Range Search performance results were unforeseen during the KDTree Construction times. Although our previous work suggested that the Distributed KDTree construction times were not reasonable, we were not expecting to see such high wait times for the tree construction. From Table 5.1, it is clear that at no point does the distributed tree construction improve performance. On the contrary, the automated agent migration performance has shown impressive results. Generally, we would expect the performance improvement to level down at some point as we increase the number of computing nodes. In the case of this application, we saw constant improvement of the execution performance of the Agent Migration across the KDTree.

Table 5.1: Range Search Benchmark Results - Distributed KDTree Construction (time in seconds)

Number of worker nodes	500,000 points	1,000,000 points
1	7.29	15.561
2	1603.236	-
4	2560.58	5259.769
8	3881.647	8111.904
12	3590.451	7586.234
16	3903.864	8105.958
24	4654.77	9876.906

It is important to mention that two computing nodes together always resulted in a

Table 5.2: Range Search Benchmark Results - Agent Migration (time in seconds)

Number of worker nodes	500,000 points	1,000,000 points
1	25.119	86.929
2	19.375	-
4	11.353	71.727
8	15.533	118.155
12	9.103	26.564
16	10.228	22.73
24	12.124	23.396

deadlocks situation upon investigating the thread dumps. With the use of logging and the thread dumps, it was revealed that the program would not make it past through the tree construction after constructing nearly 60 percent of the tree. The threads would enter into waiting state and never exit.

5.2 Closest Pair of Points

The benchmark results of the Closest Pair of Points application are complete in all three versions. With the work done on Continuous space in MASS Core Library, the handling of such large sets of data was efficient when compared to that of Range Search.

Table 5.3: Closest Pair of Points Benchmark Results (time in seconds)

Number of worker nodes	500,000 points	1,000,000 points
1	282.17	931.98
2	84.10	268.25
4	33.14	81.96
8	22.57	39.53
12	25.30	34.73
16	37.01	47.56
24	52.64	53.59

REFERENCES

- [1] V. Mohan, “Automated agent migration over structured data.” UW Master’s White Paper, 2022.
- [2] J. L. Jaron Wang, Shaun Stangler, “Convex hull css 534.” UW Master’s Level Assignment Report, 2022.
- [3] S. Paronyan, “Agent-based computational geometry.” UW Master’s White Paper, 2021.
- [4] R. Ng, “Evaluation of euclidean shortest path, voronoi digram and line segment intersection using mass, spark, and mapreduce.” UW Master’s Term Report, 2022.
- [5] V. Mohan, A. Potturi, and M. Fukuda, “Automated agent migration over distributed data structures,” in *In Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 1*.

APPENDICES