##### Image result for UWA LOGO

## Distributed All Pairs Shortest Path Algorithm

Written by Clayton Herbst (22245091)

##### Abstract

The all pairs shortest path problem is a computationally intensive task that seeks to find the shortest path between every pair of vertices in a given graph (Wolfram). There exist many adaptations to the problem that vary in computational difficulty depending on certain graphical properties such as the existence of edge weights, negative cycles and directed or undirected vertex edges. In this report we consider the performance achieved when computing the all pairs shortest path on a weighted-directed graph that contains no negative-cycles. Distributed computing techniques have been implemented to further increase the efficiency of available computing power thus, achieve further speed-up in the computation of the all pairs shortest path graph.

##### Floyd-Warshall’s Algorithm

The Floyd-Warshall problem is an algorithm for finding the shortest paths with positive or negative edge-weights (Wikipedia 2019).

##### Sequential Algorithm Performance Analysis

##### Distributed All Pairs Shortest Path Approach

##### Distributed Algorithm Performance Analysis

##### Conclusion

##### Appendix A

**Scalar Multiplication Table of Results:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Size** | **File Reading (s)** | **Integer - sync (s)** | **Integer - async (s)** | **Float - sync (s)** | **Float - async (s)** |
| **32** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **64** | 0.001 | 0.000 | 0.001 | 0.000 | 0.000 |
| **256** | 0.018 | 0.000 | 0.003 | 0.000 | 0.003 |
| **1024** | 0.238 | 0.006 | 0.002 | 0.003 | 0.001 |
| **2048** | 0.934 | 0.032 | 0.010 | 0.026 | 0.005 |
| **8192** | 14.740 | 2.249 | 0.142 | 1.762 | 0.167 |
| **16384** | 59.782 | 9.226 | 5.162 | 14.817 | 5.437 |

*Note:*

* *Input files used during testing were dense in order to ensure consistent computational difficulty across all result sets.*
* *The ‘size’ integer represents the row and column sizes of the tested square matrices.*
* *Sync represents the single threaded program execution environment.*
* *Async represents the multithreading execution environment.*

##### Appendix B

**Trace Calculation Table of Results:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Size** | **File Reading (s)** | | **Integer - sync** | **Integer - async** | **Float - sync** | **Float - async** |
| **32** | | 0.001 | 0.000 | 0.000 | 0.000 | 0.000 |
| **64** | | 0.001 | 0.000 | 0.000 | 0.000 | 0.000 |
| **256** | | 0.018 | 0.000 | 0.000 | 0.000 | 0.000 |
| **1024** | | 0.238 | 0.003 | 0.002 | 0.003 | 0.001 |
| **2048** | | 0.934 | 0.027 | 0.008 | 0.024 | 0.005 |
| **8192** | | 14.740 | 0.317 | 0.080 | 1.523 | 0.086 |
| **16384** | | 59.782 | 8.060 | 4.260 | 15.925 | 7.238 |

*Note:*

* *Input files used during testing were dense in order to ensure consistent computational difficulty across all result sets.*
* *The ‘size’ integer represents the row and column sizes of the tested square matrices.*
* *Sync represents the single threaded program execution environment.*
* *Async represents the multithreading execution environment.*

##### Appendix C

**Matrix Addition Table of Results:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Size** | **File Reading (s)** | **Integer - sync** | **Integer - async** | **Float - sync** | **Float - async** |
| **32** | 0.001 | 0.000 | 0.002 | 0.000 | 0.002 |
| **64** | 0.004 | 0.000 | 0.004 | 0.001 | 0.007 |
| **256** | 0.113 | 0.004 | 0.114 | 0.004 | 0.119 |
| **1024** | 5.059 | 0.066 | 1.876 | 0.064 | 1.900 |
| **2048** | 40.080 | 0.260 | 7.479 | 0.308 | 7.543 |

*Note:*

* *Input files used during testing were dense in order to ensure consistent computational difficulty across all result sets.*
* *The ‘size’ integer represents the row and column sizes of the tested square matrices.*
* *Sync represents the single threaded program execution environment.*
* *Async represents the multithreading execution environment.*

##### Appendix D

**Matrix Transposition Table of Results:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Size** | **File Reading (s)** | **Integer - sync** | **Integer - async** | **Float - sync** | **Float - async** |
| **32** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **64** | 0.002 | 0.000 | 0.000 | 0.000 | 0.000 |
| **256** | 0.064 | 0.000 | 0.003 | 0.000 | 0.001 |
| **1024** | 2.574 | 0.003 | 0.004 | 0.004 | 0.005 |
| **2048** | 18.995 | 0.012 | 0.022 | 0.018 | 0.017 |

*Note:*

* *Input files used during testing were dense in order to ensure consistent computational difficulty across all result sets.*
* *The ‘size’ integer represents the row and column sizes of the tested square matrices.*
* *Sync represents the single threaded program execution environment.*
* *Async represents the multithreading execution environment.*

##### Appendix E

**Matrix Multiplication Table of Results:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Size** | **File Reading (s)** | **Integer - sync** | **Integer - async** | **Float - sync** | **Float - async** |
| **32** | 0.000 | 0.000 | 0.002 | 0.000 | 0.002 |
| **64** | 0.004 | 0.002 | 0.011 | 0.002 | 0.011 |
| **256** | 0.131 | 0.114 | 0.292 | 0.111 | 0.260 |
| **1024** | 6.188 | 6.900 | 8.812 | 7.245 | 9.093 |
| **2048** | 46.149 | 54.324 | 64.459 | 56.689 | 65.791 |

*Note:*

* *Input files used during testing were dense in order to ensure consistent computational difficulty across all result sets.*
* *The ‘size’ integer represents the row and column sizes of the tested square matrices.*
* *Sync represents the single threaded program execution environment.*
* *Async represents the multithreading execution environment.*

##### References

Datta A, 2014. High Performance Computing. Available from: http://teaching.csse.uwa.edu.au/units/CITS3402/lectures/index.html.

[17 Sep 2019].

<https://en.wikipedia.org/wiki/Distributed_computing>

<https://en.wikipedia.org/wiki/Graph_theory>

<http://mathworld.wolfram.com/All-PairsShortestPath.html>