Comparative Analysis of Multi-Threaded and Non-Threaded

Implementations of the Merge Sort Algorithm

(Final Report)

By

Bernice Templeman

Florida Institute of Technology

CIS 5898 – Projects in Computer Information Systems

For

Dr. Stephen Johnson

November 7, 2016

# Table of Contents

Introduction 3

Sorting Decisions 3

Java Threads 3

BTMS Proposition 3

Algorithms / Project Solution 4

Merge-Sort ALgorithm 4

Threads 4

Data Set Size 4

Timing 4

Implementation 5

Platform 5

Development Tools 5

Programming Language and Code Libraries 5

Implementation Procedure 5

Schedule 6

Results 9

Future Enhancements 11

References 12

Appendices 13

# Introduction

Bernice’s Threaded Merge Sort (BTMS) will sort large datasets with Java using the merge sort algorithm.

### Sorting Decisions

There are several sorting algorithms to choose from. With an increase in the amount of data being created, implementing the best sort for the data will affect performance. George Heineman says in *Algorithms in a Nutshell* that “Because today’s computers are so much more powerful than the ones of 50 years ago, the size of the data sets being processed is now on the order of terabytes of information. Although you may not be called on to sort such huge data sets, you will likely need to sort large numbers of items.” Merge Sort is considered as a stable sort. It is also considered as well-suited for sorting data in secondary storage. (Heineman).

### Java Threads

To further improve the merge sort performance, multi-threaded sorts are another option. In *An Implementation of Sorting Algorithm Based on Java Multithread Technology*, Wang says

“With the spread of muticore computer, ordinary desktop computers have strongparallel processing ability. But using traditional serial sort algorithm cannot take full advantage of powerful parallel computing power of the computer. Therefore, how to reduce the recursion level and how to improve the traditional algorithms, made him able to adapt to the development of computer parallel technology, to increase the efficiency of traditional algorithms to a new level, is a worthy subject of study.”

## BTMS Proposition

BTMS will consist test the performance of a non-threaded merge sort and a multi-threaded merge sort on Windows 10 and Linux.

# Algorithms / Project Solution

This section describes the unique problem and the proposed solution that will be known as BTMS.

**The Merge-Sort Algorithm**

Merge-Sort is an algorithm that divides its input list into two lists, sorts them, and merges the two sub-lists into a sorted version of the input list.

MERGESORT(A)

ifLENGTH(A) =< 1

then returnA

B <- MERGESORT(first half of A)

C <- MERGESORT(second half of A)

A <- MERGE(B, C)

returnA

**Threads**

The multi-threaded program will allow the following number of threads to be used for the merge-sort:

* 1
* 2
* 4
* 8
* 16

**Data Set Size**

Different data set sizes will be created and tested. An array of integers will be created using the dataset. This size may hardware dependent and will be tested up to 10 billion.

**Timing**

The time will be taken before and after the sort is performed using: System.currentTimeMillis();

# Implementation

This section describes the resources that will be used to implement BTMS.

## Platform

BTMS will be a Java 8 application. It will be installed on an Intel platform running:

* Windows 10 Operating System (OS)
* Ubuntu Linux 14.04 Bootable Flash

Information and downloads for Java can be found at <https://docs.oracle.com/javase/8/>.

## Development Tools

Development will be done on an Intel machine running Windows 10 and a Linux OS Bootable drive. I will be using NetBeans IDE for Java EE Developers. Information and downloads can be found at <http://www.oracle.com/technetwork/articles/javase/jdk-netbeans-jsp-142931.html>.

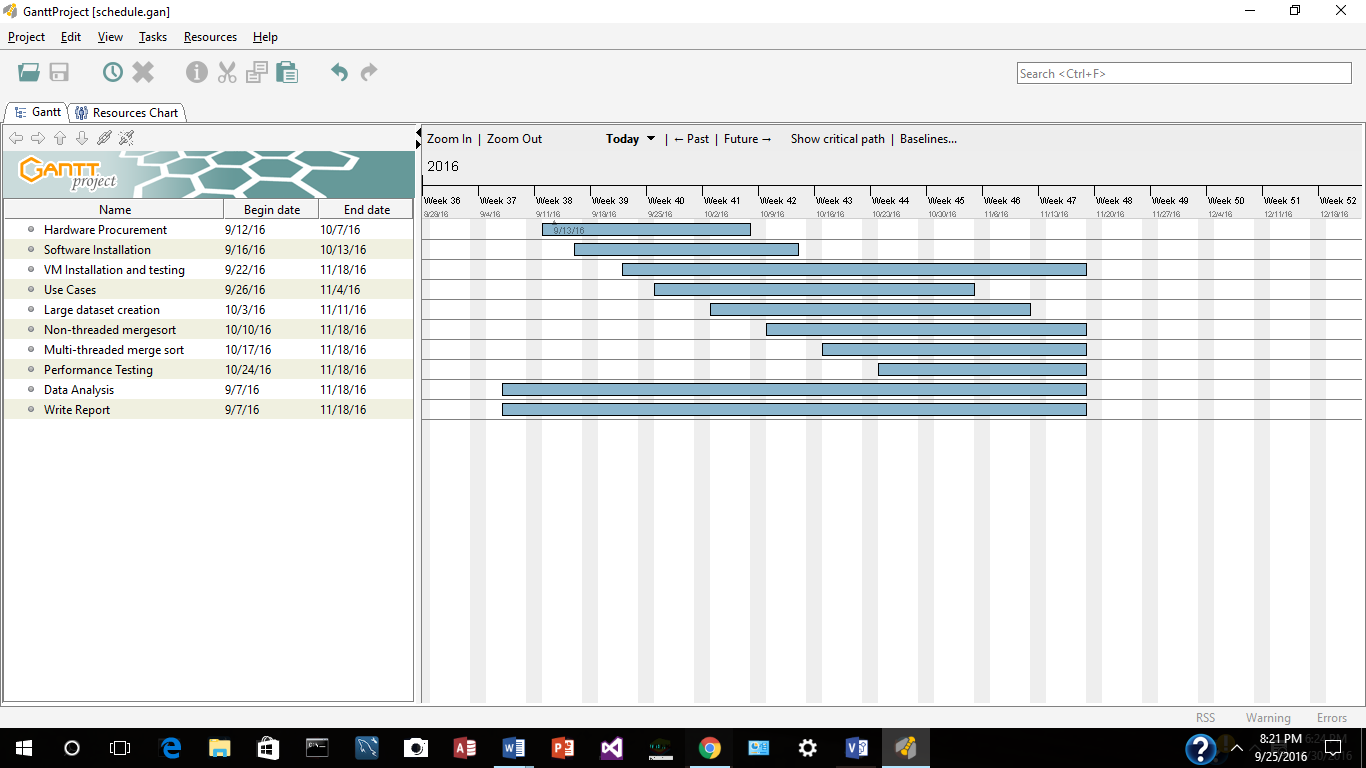
## Programming Language and Code Libraries

BTMS will be programmed in the Java programming language using the NetBeans IDE.

## Implementation Procedure

I plan to start by procuring a Windows 10 laptop, installing OS updates and installing software. I will also install Oracle VirtualBox, download a Windows 10 VM and test it during this phase. Next I will develop high-level use case diagrams for the threaded application. Also during this phase I’ll design the interface. Next, I will install Java and the NetBeans IDE on both Windows 10 hard and an Ubuntu Linux 14.04. I will research existing datasets, code to produce a dataset or writing a Java program to create a large dataset of integers to be sorted by the merge-sort programs. I will then write a non-threaded merge-sort. After the non-threaded merge-sort is complete I will write the multi-threaded merge-sort. Then I will run performance tests to compare the non-threaded merge-sort with the threaded merge-sort with different number of threads.

# Schedule



# Results

**Comparing Merge Sort Times on different Operating Systems:**

I am a long time Unix person. I first wrote C and C++ programs on Unix Systems and then also did Unix System Administration . Later I also did RedHat Linux System Administration. There were always questions on which OS developers should use. When I first ran the merge sort on the different Operating Systems, I was surprised by the results.

The following are the results of the first runs on a Windows 10 and Ubuntu Virtual Machine with 1 Processor (Random Data Set Averages):

|  |  |
| --- | --- |
| Windows 10 VM (1 Processor) | Ubuntu VM (1 Processor) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Array size  izeize | NT | 1T | 2T | 4T | 8T | 16T | NT | 1T | 2T | 4T | 8T | 16T |
| **1,000** | 0 | 17.2 | 92.5 | 192.8 | 107.63 | 282.8 | .02 | 8.6 | 10.1 | 19.3 | 39.8 | 71.6 |
| **2,000** | 0 | 17.3 | 42 | 111.2 | 291 | 278.4 | 1.9 | 9.3 | 10.2 | 17 | 41.1 | 71.3 |
| **4,000** | 0 | 15,5 | 35.8 | 134.2 | 174.8 | 426.6 | 0.1 | 8.5 | 8.6 | 17.7 | 76.8 | 89.9 |
| **8,000** | 0 | 17.2 | 39.2 | 114 | 314.2 | 355.8 | 0.2 | 8 | 12.9 | 17.9 | 42.6 | 87.4 |
| **16,000** | 0 | 39 | 37.5 | 98.5 | 171 | 325.2 | 1.8 | 7.3 | 12.2 | 18.3 | 38.8 | 69.9 |
| **32,000** | 6.4 | 15.5 | 35.8 | 114 | 152.2 | 343.9 | 1.1 | 9.7 | 13 | 20.3 | 53.1 | 114 |
| **64,000** | 6.1 | 16.2 | 56.2 | 215.6 | 147.7 | 346.7 | 2.8 | 11.1 | 14.3 | 20.6 | 59.5 | 146,8 |
| **128,000** | 1.5 | 55.8 | 51.6 | 83.1 | 624.1 | 384.2 | 7.4 | 19.4 | 17.9 | 23 | 49.4 | 169.1 |
| **25,6000** | 14.4 | 26.7 | 58.1 | 146.7 | 497.2 | 327.2 | 17 | 19.8 | 22.3 | 29.3 | 64.2 | 178.7 |
| **512,000** | 22.7 | 132.8 | 82.5 | 193.8 | 214 | 385.5 | 46.9 | 34.9 | 36.6 | 73.9 | 114 | 202.7 |
| **1,024,000** | 41.4 | 81.1 | 193.9 | 173.3 | 231 | 409.4 | 88.5 | 74.1 | 73.3 | 72.7 | 195.2 | 290.9 |
| **2,048,000** | 98.1 | 173.7 | 312,2 | 245.5 | 351.6 | 503.3 | 169.9 | 153.6 | 136.7 | 239.6 | 285.2 | 389.3 |
| **4096000** | 217.1 | 474.2 | 403.1 | 320.2 | 370.5 | 598.4 | 339.9 | 418.9 | 397.6 | 457.5 | 498.1 | 561.3 |
| **8192000** | 468.6 | 609.1 | 593.9 | 687.6 | 589.3 | 798.5 | 817.1 | 793.9 | 847.3 | 851.8 | 894.2 | 561.3 |
| **16384000** | 834.5 | 1334.3 | 1013.2 | 1124.5 | 1016.2 | 1206.3 | 1517.2 | 1509.6 | 1544.7 | 1623.4 | 1655.3 | 1708.2 |
| **32768000** | 1648.5 | 1864.1 | 1778.1 | 2162.7 | 1866.9 | 2101.7 | 2434 | 2945 | 2973.2 | 3107.7 | 2985.4 | 3300.6 |
| **65,536,000** | 3440.5 | 3917.5 | 3561.1 | 3468.8 | 3492.3 | 3623.4 | 4850.8 | 5356.6 | 5282.8 | 5323.3 | 5225.3 | 5360.9 |

The results show that on the Virtual Machines with 1 processor, Windows 10 Non-Threaded was the fastest.

I then changed the Virtual Machine Configurations to 2 Processors and these are the results from the Random Data Sets (Averages):

|  |  |
| --- | --- |
| Windows 10 VM(2 Processors) | Ubuntu VM (2 Processors) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Array size  izeize | NT | 1T | 2T | 4T | 8T | 16T | NT | 1T | 2T | 4T | 8T | 16T |
| **1,000** | 0 | 40.3 | 37.6 | 64.1 | 121.9 | 248.2 | .1 | 23 | 26.8 | 58.9 | 106.9 | 220.2 |
| **2,000** | 0 | 32.6 | 40.4 | 67.1 | 124.7 | 250 | 1.3 | 17.8 | 22.8 | 52.8 | 107.9 | 225.6 |
| **4,000** | 1.5 | 28.2 | 40.4 | 64.2 | 122.2 | 256.5 | 0.1 | 17.9 | 25.8 | 49.5 | 119.8 | 232 |
| **8,000** | 0 | 31.3 | 37.6 | 67.2 | 132.5 | 248.1 | 0.1 | 23.3 | 25.9 | 72.1 | 124.1 | 225.1 |
| **16,000** | 3 | 27.8 | 39.1 | 63.9 | 120.3 | 251.6 | 1.5 | 17.8 | 25.5 | 60 | 118.3 | 206.2 |
| **32,000** | 1.5 | 26.8 | 35.8 | 61.3 | 125.2 | 254.6 | 1.3 | 17.5 | 29.9 | 57.3 | 121 | 228.7 |
| **64,000** | 0 | 26.6 | 37.5 | 60.8 | 117.1 | 262.6 | 2.9 | 19.3 | 33.3 | 72.5 | 127.2 | 236.9 |
| **128,000** | 3.1 | 35.7 | 39 | 65.4 | 126.6 | 252.8 | 5.1 | 29.9 | 37.2 | 76.3 | 121.1 | 287.1 |
| **25,6000** | 7.9 | 36 | 46.8 | 68.8 | 129.6 | 245.5 | 10.9 | 35.5 | 52.5 | 85.1 | 125.5 | 245.2 |
| **512,000** | 22 | 48.3 | 56.2 | 73.3 | 131.2 | 245.1 | 25.1 | 55.7 | 59.6 | 120 | 154.6 | 267.1 |
| **1,024,000** | 42.3 | 77.9 | 66.9 | 86 | 140.4 | 262.4 | 50.9 | 84.5 | 108 | 146.7 | 245.6 | 327.5 |
| **2,048,000** | 85.6 | 120.4 | 98.4 | 107.5 | 159.2 | 276.1 | 112.1 | 159.2 | 185.5 | 212.5 | 280.7 | 391.6 |
| **4096000** | 168.8 | 215.6 | 163.6 | 162.5 | 210.9 | 318.8 | 235.1 | 264.4 | 278.2 | 325.4 | 403.7 | 531.7 |
| **8192000** | 336.1 | 395.8 | 295.2 | 265.6 | 314.1 | 415.5 | 390.8 | 478.6 | 447.1 | 476.8 | 571 | 764.1 |
| **16384000** | 688.9 | 767.4 | 535.7 | 493.8 | 512.4 | 628.1 | 747.5 | 915.8 | 727.8 | 736.9 | 875.9 | 1071.1 |
| **32768000** | 1361 | 1429.6 | 1006.2 | 867.3 | 909.3 | 1017.3 | 1479.6 | 1674.4 | 1324.4 | 1325 | 1403.4 | 1544.6 |
| **65,536,000** | 2739 | 2815.1 | 1954.7 | 1657.6 | 1715.6 | 1800 | 2952.8 | 3255.5 | 2500.2 | 2538.2 | 2575.2 | 2725.4 |

**Observations Of adding a processor to Virtual Machine:**

**Comparing Non-Threaded and Threaded Merge Sorts:**

**Comparing Different Data Sets:**

**Comparing Virtual Machine to Host:**

TABLE I. RESULTS OF IN Windows 10 Virtual Machine (1 Processor)

|  |  |  |  |
| --- | --- | --- | --- |
| **threads** | Results(data size:10000000 integers) | | |
| ***Traditional quick sort cost time(ms)*** | ***Our multi-thread sort cost time(ms)*** | ***Efficiency ratio*** |
| 1 | 2107 | 2088 | 1% |
| 2 | 2167 | 1306 | 40% |
| 4 | 2162 | 1377 | 36% |
| 8 | 2222 | 1495 | 33% |
| 500 | 2232 | 1823 | 18% |
| 1000 | 2214 | 2092 | 6% |

TABLE II. RESULTS OF IN UBUNTU 11.04

|  |  |  |  |
| --- | --- | --- | --- |
| **threads** | Results(data size:10000000 integers) | | |
| ***Traditional quick sort cost time(ms)*** | ***Our multi-thread sort cost time(ms)*** | ***Efficiency ratio*** |
| 1 | 1859 | 1824 | 2% |
| 2 | 1897 | 1165 | 40% |
| 4 | 2022 | 1286 | 36% |
| 8 | 1862 | 1271 | 32% |
| 500 | 1944 | 1637 | 16% |
| 1000 | 1813 | 1745 | 4% |

TABLE I. RESULTS OF IN WIN7

|  |  |  |  |
| --- | --- | --- | --- |
| **threads** | Results(data size:10000000 integers) | | |
| ***Traditional quick sort cost time(ms)*** | ***Our multi-thread sort cost time(ms)*** | ***Efficiency ratio*** |
| 1 | 2107 | 2088 | 1% |
| 2 | 2167 | 1306 | 40% |
| 4 | 2162 | 1377 | 36% |
| 8 | 2222 | 1495 | 33% |
| 500 | 2232 | 1823 | 18% |
| 1000 | 2214 | 2092 | 6% |

TABLE II. RESULTS OF IN UBUNTU 11.04

|  |  |  |  |
| --- | --- | --- | --- |
| **threads** | Results(data size:10000000 integers) | | |
| ***Traditional quick sort cost time(ms)*** | ***Our multi-thread sort cost time(ms)*** | ***Efficiency ratio*** |
| 1 | 1859 | 1824 | 2% |
| 2 | 1897 | 1165 | 40% |
| 4 | 2022 | 1286 | 36% |
| 8 | 1862 | 1271 | 32% |
| 500 | 1944 | 1637 | 16% |
| 1000 | 1813 | 1745 | 4% |

1. Non threaded vs Threaded merge sort performance

Chart of windows random set

Nonthreaded and each thread

Chart of linux random set

Nonthreaded and each thread

1. Windows vs Linux
2. Any difference in different data sets?
3. Data for windows laptop (different number of processors)?

Windows non threaded?

Multi threaded very large data sets?

Strange windows time cost at the beginning of threaded runs (running in loop)

Test windows threaded 1 run at a time to see time cost each run?

Latest Windows 10 and Ubuntu

# Future Enhancements

Performance optimizations

External data set

Report generation

Multiple processors

# References

Heineman, George T., et al. *Algorithms in a Nutshell*, O’Reilly Media, Sebastopol, CA, 2016.

Wang, Deming, et al*. An Implementation of Sorting Algorithm Based on Java Multithread*

*Technology*, vol. 1, IEEE, 2012.doi:10.1109/ICCSEE.2012.152.

Concurrency Tutorial by Oracle,

<http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html>, Accessed in September 2016.

Java Documentation by Oracle,

<https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html>, Accessed in September 2016.

Java Documentation by Oracle,

<https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html>, Accessed in September 2016.

# Appendices

**APPENDIX A**

**Status Reports**

**A.1** Capstone Project Week 5 Status Report for “Comparative Analysis of Multi-Threaded and Non-Threaded Implementations of the Merge Sort Algorithm”

**Date:** 25-September-2016

**Accomplishments**

Activity 1: Hardware and software Installation and testing (22-August-2009 - 25-September-2016)

1. Hardware Procurement – completed for existing hardware. On-going for Virtual Machine requirement.
2. Software Installation – completed for existing hardware. On-going for Virtual Machine requirement.
3. VM installation and testing – completed on existing hardware.

**Current Activities (26-Sept-2016 – 03-Oct-2016)**

Activity 2: Virtual Machine hardware and software procurement and testing and Use Cases

1. Hardware requirements research for Virtual Machines, possible additional hardware procurement and testing.
2. Use Cases

**Challenges**

The challenges are:

1. Additional hardware requirements for Virtual Machines
2. How to obtain or produce a valid data set

The challenges listed above can be resolved and do not place the project at risk.

**Work to be Completed by Oct. 9, 2016**

1. Hardware for Virtual Machine because it is a Capstone Requirement
2. Use Cases
3. Data Set research

**A.2** Capstone Project Week 7 Status Report for “Comparative Analysis of Multi-Threaded and Non-Threaded Implementations of the Merge Sort Algorithm”

**Date:** 03-October-2016

**Accomplishments**

Activity 1: Hardware and software Installation and testing (25-September-2009 - 03-October-2016)

1. Hardware Procurement – completed for existing hardware and for Virtual Machine requirement.
2. Software Installation – completed for existing hardware and for Virtual Machine requirement.
3. VM installation and testing – completed on existing hardware.

 Activity 2: Data Set Research

1. Data Set Research – on-going. Downloaded code from *”Algorithms in a Nutshell”* and researching datasets through the FIT Library.
2. One research paper used several different datasets.

* Random
* Function to return Integers in a range
* Bernoulli function to return 0 or 1
* Geometric function
* Pascal function
* Binomial function
* Zero function
* Unbalanced function

**Current Activities (03-Oct-2016 – 09-Oct-2016)**

Activity 2: Data Set Research and Use Cases

1. Data Set Research and testing code for MergeSort from “*Algorithms in a Nutshell”*
2. Use Cases

**Challenges**

The challenges are:

1. How to obtain or produce a valid data set

The challenges listed above can be resolved and do not place the project at risk.

**Work to be Completed by Oct. 9, 2016**

1. Data Set research
2. Use Cases

**A.3 Capstone Project Week 9 Status Report for** **“Comparative Analysis of Multi-Threaded and Non-Threaded Implementations** **of the Merge Sort Algorithm”**

**Date:** 23-October-2016

**Accomplishments**

Activity 1: Hardware Procurement – complete

Activity 2: Software Installation – complete

Activity 3: VM installation and testing – complete

Activity 4: Uses Cases – in progress

Activity 5: Large Data Set creation – in progress

Activity 6: Non-threaded Merge Sort – in progress

Activity 7: Multi-threaded Merge sort – in progress

Activity 8: Performance Testing – in progress

Activity 9: Data Analysis – in progress

Activity 10: Write report – in progress

**Current Activities (23-Oct-2016 – 06-Nov-2016)**

Activity 4: Uses Cases – in progress

* I have a model for a Use case on paper. My laptop with Visio is at another location. Plan to complete on software this week.

Activity 5: Large Data Set creation – in progress

* Testing existing code for large data set creation
* Coding to create additional data sets based on research results

Activity 6: Non-threaded Merge Sort – in progress

* Testing some existing code
* Researching other merge sort code

Activity 7: Multi-threaded Merge sort – in progress

* Testing some existing code
* Researching other merge sort code

Activity 8: Performance Testing – in progress

Activity 9: Data Analysis – in progress

Activity 10: Write report – in progress

**Challenges**

The challenges are:

1. I am currently testing on my Windows OS on my laptop. I need to Finalizing the code for the data set creation and merge sort algorithm then move to virtual machines and test on the Windows VM and Linux VM.

The challenges listed above can be resolved and do not place the project at risk.

**Work to be Completed by Nov. 6, 2016**

Activity 4: Use Cases - in progress

Activity 5: Large Data Set creation – in progress

Activity 6: Non-threaded Merge Sort – in progress

Activity 7: Multi-threaded Merge sort – in progress

Activity 8: Performance Testing – in progress

Activity 9: Data Analysis – in progress

Activity 10: Write report – in progress

A4. Capstone Project Week 11 Status Report for “Comparative Analysis of Multi-Threaded and Non-Threaded Implementations of the Merge Sort Algorithm”

**Date:** 05-November-2016

**Accomplishments**

Activity 5: Large Data Set creation – in progress

Data Sets I am currently using for testing the performance of my project are:

* Random ( from [https://docs.oracle.com/javase/8/docs/api/java/util/Random.html (Links to an external site.)](https://docs.oracle.com/javase/8/docs/api/java/util/Random.html) )
* StdRandom.java ( from [http://introcs.cs.princeton.edu/java/stdlib (Links to an external site.)](http://introcs.cs.princeton.edu/java/stdlib) )
  + Bernoulli,
  + uniform,
  + Gaussian,
  + Discrete
* Not Tested yet:
  + Zero function (all zeros)

Data Set Size

* Testing sizes of data sets to run on my hardware.
* Have not tested external storage since this will need to run on Virtual Machines

Activity 6: Non-threaded Merge Sort – in progress

* Testing some existing code
* Researching other merge sort code

Activity 7: Multi-threaded Merge sort – in progress

* Testing some existing code
* Researching other merge sort code

Activity 8: Performance Testing – in progress

Activity 9: Data Analysis – in progress

Activity 10: Write report – in progress

Activity 11: Configuration Management

* I have configured GitHub on my laptop to sync with an external site

**Current Activities (05-Nov-2016 – 20-Nov-2016)**

Activity 4: Uses Cases – in progress

Activity 5: Large Data Set creation – in progress

Activity 6: Non-threaded Merge Sort – in progress

Activity 7: Multi-threaded Merge sort – in progress

* Need to test different number of threads

Activity 8: Performance Testing – in progress

Activity 9: Data Analysis – in progress

Activity 10: Write report – in progress

Activity 12: Video  - not tested or completed.

**Challenges**

The challenges are:

1. Added configuration management to the plan for security purposes.

The challenges listed above can be resolved and do not place the project at risk.

**Work to be Completed by Nov. 20, 2016**

Activity 5: Large Data Set creation – in progress

Activity 6: Non-threaded Merge Sort – in progress

Activity 7: Multi-threaded Merge sort – in progress

Activity 8: Performance Testing – in progress

Activity 9: Data Analysis – in progress

Activity 10: Write report – in progress

Activity 12: Video

**APPENDIX B**

**Software Configuration**

**B.1 Steps to install install DAWS Server**

\*\* Note: DAWS Server is currently running at <http://50.56.72.178/DAWS/services/DAWSService/> for this project demonstration. This is the service endpoint reference for DAWS. For manual installation, please follow the instructions below:

Dependency Files:

JDK 6 or higher (<http://www.java.com>)

Apache Tomcat 6 or higher (<http://tomcat.apache.org>)

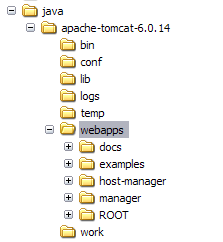
Project File:

DAWS.war (<http://50.56.72.178/Downloads>)

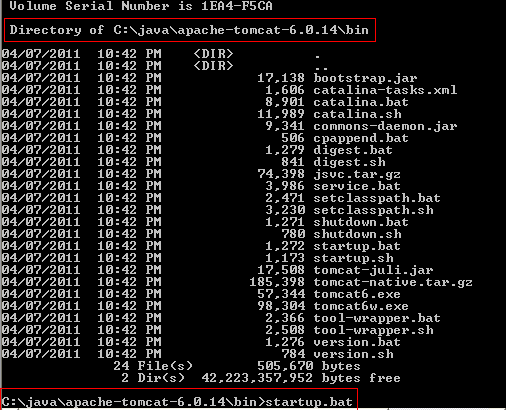
DAWS Server is dependent on the server running JDK 6 or higher. The JRE is not enough, it must be JDK. The DAWS.war file is a java web archive compliant with J2EE standards of deploying web applications. It can be installed on any Java Web Server capable of deploying WAR files. For this demo, Apache Tomcat 6 or higher should be used. You can download Apache Tomcat at <http://tomcat.apache.org/>. Once downloaded, follow the documentation on the Apache Tomcat website on how to install. Be sure to you have Java SDK (not JRE) 6 or higher installed first. If you are unable to receive the DAWS.war file as an attachment to email due to size limitations, go to <http://50.56.72.178/Downloads/> and click on the DAWS.war file to download the file.

Once all resource files are downloaded and JDK 6 and Apache Tomcat are installed, follow these steps to install DAWS:

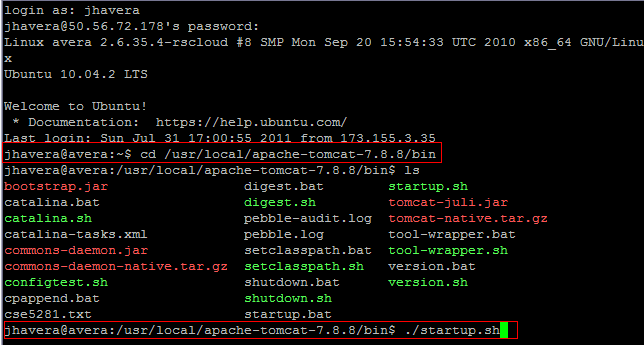
1. Copy DAWS.war to the webapps directory where Tomcat is installed (see image below)



1. Open a command prompt and go to the root directory of where Tomcat is installed and type the following command:
   1. Windows: {$tomcat-home}\bin\startup.bat



* 1. Unix: {$tomcat-home}/bin.startup.sh



1. This will startup Tomcat which will automatically deploy DAWS web application / web service.

**B.2 Steps to install DAWS Database**

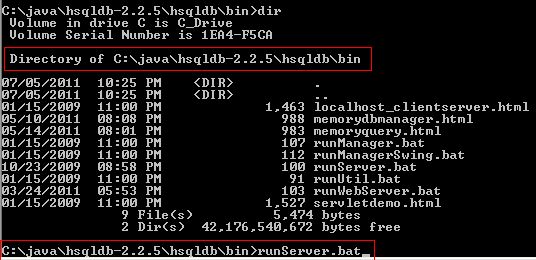
\*\* Note: DAWS Database Server is currently running at jdbc:hsqldb:hsql://50.56.72.178/ for this project demonstration. This is the JDBC connection URL for DAWS Database. The DAWS Server is already configured to connect to the DAWS Database for the demonstration. For manual installation, please follow the instructions below:

Dependency Files:

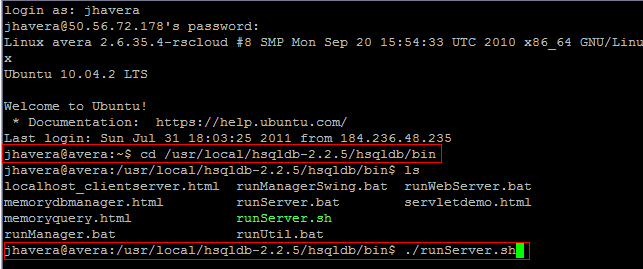
JDK 6 or higher ([www.java.com](http://www.java.com))

HyperSQL 2.2.5 ([www.hsqldb.org](http://www.hsqldb.org)) or higher (<http://sourceforge.net/projects/hsqldb/files/>)

1. Follow the documentation on the HyperSQL site on how to install and run in server mode.
2. Open a command prompt and go to the root directory of where Tomcat is installed and type the following command:
   1. Windows: {$hsqldb-home}\bin\runServer.bat



* 1. Unix: {$hsqldb-home}/bin.startup.sh



1. Execute the following script to build the TESTDATA table:

CREATE TABLE TESTDATA (

ID INTEGER,

FIRSTNAME VARCHAR(50),

LASTNAME VARCHAR(50),

STREET VARCHAR(50),

CITY VARCHAR(50)

);

1. Data can be loaded into the table using the DAWS RCP client application.

**B.3 Steps to install DAWS RCP Client**

Dependency Files:

JDK 6 or higher (<http://www.java.com>)

Project File:

DAWS-RCP.zip (<http://50.56.72.178/Downloads>)

DAWS-RCP is a client application for Windows designed to demonstrate a practical purpose and capability of DAWS. Although it is a Java application, a Unix version was not built for this demonstration. To install, please follow these steps:

1. Unzip DAWS-RPC.zip file to your hard drive on a Windows machine.
2. Go to the install directory and find eclipse.exe in the eclipse subdirectory.
3. Double-click the eclipse.exe, and you should see a brief splash screen followed by the main client application:

**APPENDIX C**

**User’s Manual**

DAWS-RCP is a client application for Windows designed to demonstrate a practical purpose and capability of DAWS. Once you have it up and running, follow these steps to demonstrate the capabilities:

1. Select TESTDATA in the Data Sets Available combo box:
2. Click Get Data button:
3. See results of how long the data file was produced and transferred to the client using DAWS. The screenshot below are the results of running DAWS over a cellular network driving down the road using TESTDATA with 10,000 rows. The results will vary based on bandwidth. For most broadband connection, the performance will be less than 2 seconds.
4. Optional step is to click the Display Data to see the data once it is parsed on the client side. A time is displayed to perform this function. This would be similar if the client wanted to parse the data file and persist the data to its own database:

**Experiment 1: Random Data Set: No Threads**

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | **Windows 10**  **Virtual Machine** | **Ubuntu 16.04 LTS**  **Virtual Machine** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_RUN\_\_**  **Elements** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1,000** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2,000** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4,000** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8,000** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **16,000** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| **32,000** | 0 | 0 | 0 | 16 | 0 | 16 | 17 | 0 | 15 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 |
| **64,000** | 15 | 15 | 0 | 0 | 0 | 0 | 0 | 15 | 0 | 16 | 2 | 3 | 3 | 2 | 3 | 6 | 2 | 3 | 2 | 2 |
| **128,000** | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 | 5 | 8 | 5 | 5 | 10 | 5 | 5 | 4 | 4 |
| **25,6000** | 16 | 17 | 0 | 15 | 16 | 16 | 16 | 16 | 16 | 16 | 41 | 10 | 16 | 24 | 9 | 17 | 10 | 9 | 24 | 10 |
| **512,000** | 15 | 16 | 31 | 16 | 16 | 14 | 15 | 32 | 16 | 16 | 172 | 27 | 41 | 48 | 22 | 50 | 19 | 17 | 36 | 37 |
| **1,024,000** | 31 | 31 | 31 | 31 | 47 | 48 | 40 | 31 | 46 | 78 | 277 | 84 | 95 | 73 | 66 | 96 | 39 | 38 | 71 | 46 |
| **2,048,000** | 109 | 93 | 109 | 94 | 109 | 94 | 93 | 93 | 78 | 109 | 548 | 152 | 178 | 154 | 110 | 159 | 82 | 79 | 142 | 95 |
| **4096000** | 251 | 204 | 218 | 203 | 203 | 202 | 187 | 202 | 204 | 297 | 908 | 248 | 366 | 381 | 192 | 360 | 218 | 223 | 339 | 164 |
| **8192000** | 952 | 422 | 422 | 453 | 375 | 422 | 375 | 422 | 421 | 422 | 1947 | 772 | 738 | 645 | 740 | 669 | 624 | 778 | 690 | 568 |
| **16384000** | 1438 | 797 | 750 | 735 | 766 | 766 | 719 | 765 | 781 | 828 | 1770 | 1485 | 1484 | 1298 | 1537 | 2355 | 1378 | 1521 | 1193 | 1151 |
| **32768000** | 1968 | 1546 | 1657 | 1579 | 1594 | 1704 | 1656 | 1593 | 1610 | 1578 | 3398 | 2786 | 2937 | 2640 | 3042 | 1297 | 2889 | 3118 | 2541 | 2478 |
| **65,536,000** | 5671 | 3437 | 3141 | 3188 | 3172 | 3203 | 3109 | 3140 | 3203 | 3141 | 7946 | 5134 | 5869 | 4461 | 5295 | 4771 | 5177 | 5322 | 4823 | 4844 |

**Non Threaded Merge Sort Averages: Random Data Set**

|  |  |  |  |
| --- | --- | --- | --- |
| **1,000** | 0 | .02 | 0 |
| **2,000** | 0 | 1.9 | 0 |
| **4,000** | 0 | 0.1 | 0 |
| **8,000** | 0 | 0.2 | 0 |
| **16,000** | 0 | 1.8 | 0 |
| **32,000** | 6.4 | 1.1 | 1 |
| **64,000** | 6.1 | 2.8 | 3 |
| **128,000** | 1.5 | 7.4 | 5 |
| **25,6000** | 14.4 | 17 | 10 |
| **512,000** | 22.7 | 46.9 | 27 |
| **1,024,000** | 41.4 | 88.5 | 84 |
| **2,048,000** | 98.1 | 169.9 | 152 |
| **4096000** | 217.1 | 339.9 | 248 |
| **8192000** | 468.6 | 817.1 | 772 |
| **16384000** | 834.5 | 1517.2 | 1485 |
| **32768000** | 1648.5 | 2434 | 2786 |
| **65,536,000** | 3440.5 | 4850.8 | 5134 |

**Experiment 2: Random Data Set: 1 Thread**

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | **Windows 10**  **Virtual Machine** | **Ubuntu 16.04 LTS**  **Virtual Machine** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_RUN\_\_**  **Elements** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1,000** | 31 | 16 | 16 | 16 | 16 | 15 | 16 | 15 | 15 | 16 | 17 | 5 | 11 | 7 | 6 | 13 | 5 | 6 | 10 | 6 |
| **2,000** | 31 | 15 | 16 | 16 | 16 | 16 | 16 | 16 | 0 | 31 | 27 | 5 | 11 | 5 | 5 | 12 | 5 | 5 | 11 | 7 |
| **4,000** | 32 | 16 | 15 | 15 | 15 | 15 | 16 | 15 | 16 | 0 | 14 | 5 | 15 | 6 | 5 | 16 | 6 | 7 | 6 | 5 |
| **8,000** | 31 | 16 | 16 | 15 | 16 | 16 | 15 | 16 | 15 | 16 | 11 | 6 | 11 | 6 | 7 | 14 | 8 | 5 | 6 | 6 |
| **16,000** | 251 | 15 | 16 | 16 | 15 | 15 | 15 | 15 | 16 | 16 | 9 | 6 | 12 | 6 | 6 | 9 | 7 | 5 | 7 | 6 |
| **32,000** | 14 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 15 | 14 | 10 | 7 | 13 | 17 | 7 | 16 | 7 | 6 | 7 | 7 |
| **64,000** | 17 | 16 | 15 | 15 | 16 | 16 | 16 | 19 | 16 | 16 | 19 | 9 | 16 | 13 | 15 | 7 | 8 | 8 | 8 | 8 |
| **128,000** | 359 | 31 | 16 | 17 | 16 | 15 | 15 | 27 | 31 | 31 | 12 | 10 | 81 | 28 | 10 | 11 | 10 | 11 | 10 | 11 |
| **25,6000** | 31 | 15 | 31 | 30 | 16 | 32 | 32 | 33 | 15 | 32 | 17 | 15 | 39 | 32 | 15 | 21 | 15 | 14 | 14 | 16 |
| **512,000** | 906 | 47 | 46 | 32 | 47 | 47 | 109 | 31 | 32 | 31 | 29 | 26 | 55 | 70 | 25 | 39 | 25 | 25 | 26 | 29 |
| **1,024,000** | 156 | 78 | 63 | 63 | 93 | 63 | 62 | 78 | 77 | 78 | 116 | 47 | 105 | 128 | 76 | 70 | 45 | 47 | 59 | 48 |
| **2,048,000** | 688 | 110 | 124 | 125 | 125 | 110 | 110 | 110 | 126 | 109 | 183 | 153 | 205 | 175 | 182 | 106 | 116 | 129 | 132 | 155 |
| **4096000** | 2578 | 219 | 251 | 251 | 251 | 242 | 219 | 281 | 232 | 218 | 1187 | 278 | 407 | 411 | 359 | 396 | 207 | 341 | 311 | 292 |
| **8192000** | 1954 | 406 | 641 | 436 | 437 | 437 | 500 | 437 | 437 | 406 | 957 | 745 | 797 | 614 | 735 | 790 | 774 | 875 | 821 | 831 |
| **16384000** | 5657 | 891 | 844 | 827 | 859 | 874 | 796 | 875 | 829 | 891 | 1796 | 1226 | 1604 | 1289 | 1395 | 1565 | 1628 | 1700 | 1345 | 1548 |
| **32768000** | 3532 | 1875 | 1656 | 1641 | 1672 | 1625 | 1687 | 1672 | 1594 | 1687 | 3305 | 2417 | 3085 | 2486 | 2932 | 3148 | 2998 | 3187 | 2848 | 3044 |
| **65536000** | 9156 | 3471 | 3172 | 3219 | 3188 | 3140 | 4297 | 3094 | 3220 | 3218 | 5848 | 5002 | 6051 | 4913 | 5326 | 5394 | 5226 | 5307 | 5253 | 5246 |

**1 Threaded Merge Sort Averages: Random Data Set**

|  |  |  |  |
| --- | --- | --- | --- |
| **1,000** | 17.2 | 8.6 | 0 |
| **2,000** | 17.3 | 9.3 | 0 |
| **4,000** | 15,5 | 8.5 | 0 |
| **8,000** | 17.2 | 8 | 0 |
| **16,000** | 39 | 7.3 | 0 |
| **32,000** | 15.5 | 9.7 | 1 |
| **64,000** | 16.2 | 11.1 | 3 |
| **128,000** | 55.8 | 19.4 | 5 |
| **25,6000** | 26.7 | 19.8 | 10 |
| **512,000** | 132.8 | 34.9 | 27 |
| **1,024,000** | 81.1 | 74.1 | 84 |
| **2,048,000** | 173.7 | 153.6 | 152 |
| **4096000** | 474.2 | 418.9 | 248 |
| **8192000** | 609.1 | 793.9 | 772 |
| **16384000** | 1334.3 | 1509.6 | 1485 |
| **32768000** | 1864.1 | 2945 | 2786 |
| **65,536,000** | 3917.5 | 5356.6 | 5134 |

**Experiment 3:Random Data Set: 2 Threads: All Runs, ms**

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | **Windows 10**  **Virtual Machine** | **Ubuntu 16.04 LTS**  **Virtual Machine** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_\_\_\_\_\_RUN\_\_\_\_\_**  **Elements** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1,000** | 578 | 31 | 31 | 32 | 32 | 32 | 110 | 16 | 31 | 32 | 12 | 9 | 18 | 10 | 9 | 8 | 9 | 8 | 9 | 9 |
| **2,000** | 94 | 31 | 31 | 30 | 31 | 31 | 78 | 32 | 31 | 31 | 11 | 9 | 17 | 9 | 9 | 9 | 9 | 10 | 9 | 10 |
| **4,000** | 31 | 31 | 32 | 16 | 30 | 31 | 94 | 31 | 31 | 31 | 11 | 9 | 17 | 9 | 9 | 9 | 9 | 9 | 9 | 13 |
| **8,000** | 94 | 32 | 31 | 32 | 32 | 31 | 62 | 31 | 31 | 16 | 9 | 21 | 30 | 9 | 10 | 9 | 9 | 10 | 9 | 13 |
| **16,000** | 80 | 31 | 31 | 31 | 31 | 31 | 62 | 31 | 32 | 15 | 16 | 21 | 18 | 9 | 9 | 10 | 9 | 9 | 9 | 12 |
| **32,000** | 61 | 31 | 32 | 31 | 32 | 31 | 47 | 31 | 31 | 31 | 16 | 11 | 30 | 10 | 10 | 10 | 10 | 10 | 10 | 13 |
| **64,000** | 125 | 62 | 46 | 31 | 32 | 47 | 109 | 47 | 31 | 32 | 17 | 12 | 29 | 13 | 11 | 12 | 11 | 12 | 11 | 15 |
| **128,000** | 47 | 31 | 48 | 78 | 31 | 31 | 110 | 47 | 47 | 46 | 18 | 15 | 47 | 14 | 15 | 13 | 15 | 15 | 13 | 14 |
| **25,6000** | 172 | 33 | 46 | 31 | 32 | 48 | 94 | 31 | 47 | 47 | 21 | 19 | 50 | 20 | 20 | 20 | 18 | 19 | 18 | 18 |
| **512,000** | 265 | 46 | 62 | 47 | 46 | 62 | 109 | 47 | 63 | 78 | 34 | 29 | 72 | 45 | 29 | 39 | 28 | 32 | 29 | 29 |
| **1,024,000** | 657 | 94 | 110 | 77 | 95 | 93 | 266 | 110 | 93 | 344 | 68 | 79 | 125 | 84 | 55 | 101 | 52 | 52 | 49 | 68 |
| **2,048,000** | 1859 | 125 | 140 | 141 | 110 | 124 | 233 | 139 | 126 | 125 | 153 | 111 | 265 | 90 | 131 | 118 | 109 | 116 | 168 | 106 |
| **4096000** | 1547 | 282 | 234 | 234 | 250 | 250 | 438 | 265 | 266 | 265 | 509 | 398 | 441 | 392 | 394 | 377 | 348 | 409 | 325 | 383 |
| **8192000** | 1422 | 438 | 468 | 453 | 454 | 437 | 859 | 454 | 453 | 501 | 1137 | 780 | 1015 | 824 | 786 | 647 | 784 | 914 | 849 | 737 |
| **16384000** | 1765 | 844 | 875 | 883 | 875 | 876 | 1484 | 827 | 875 | 828 | 1797 | 1498 | 1581 | 1595 | 1704 | 1306 | 1478 | 1693 | 1431 | 1364 |
| **32768000** | 2797 | 1578 | 1672 | 1672 | 1624 | 1719 | 1735 | 1672 | 1656 | 1656 | 3671 | 2773 | 2827 | 3067 | 3181 | 2704 | 2670 | 3170 | 2897 | 2772 |
| **65536000** | 6265 | 3453 | 3235 | 3234 | 3281 | 3235 | 3187 | 3250 | 3174 | 3297 | 5994 | 5210 | 5153 | 5418 | 5506 | 4863 | 5164 | 5240 | 5177 | 5103 |

**Averages 2 threads**

|  |  |  |  |
| --- | --- | --- | --- |
| **1,000** | 92.5 | 10.1 | 0 |
| **2,000** | 42 | 10.2 | 0 |
| **4,000** | 35.8 | 8.6 | 0 |
| **8,000** | 39.2 | 12.9 | 0 |
| **16,000** | 37.5 | 12.2 | 0 |
| **32,000** | 35.8 | 13 | 1 |
| **64,000** | 56.2 | 14.3 | 3 |
| **128,000** | 51.6 | 17.9 | 5 |
| **25,6000** | 58.1 | 22.3 | 10 |
| **512,000** | 82.5 | 36.6 | 27 |
| **1,024,000** | 193.9 | 73.3 | 84 |
| **2,048,000** | 312,2 | 136.7 | 152 |
| **4096000** | 403.1 | 397.6 | 248 |
| **8192000** | 593.9 | 847.3 | 772 |
| **16384000** | 1013.2 | 1544.7 | 1485 |
| **32768000** | 1778.1 | 2973.2 | 2786 |
| **65,536,000** | 3561.1 | 5282.8 | 5134 |

**Experiment 4: Random Data Set: 4 Threads ms**

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | **Windows 10**  **Virtual Machine** | **Ubuntu 16.04 LTS**  **Virtual Machine** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_\_\_\_\_\_RUN\_\_\_\_\_**  **Elements** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1,000** | 1344 | 62 | 62 | 55 | 94 | 62 | 62 | 63 | 62 | 62 | 25 | 30 | 17 | 16 | 16 | 17 | 17 | 17 | 16 | 22 |
| **2,000** | 624 | 63 | 48 | 63 | 47 | 48 | 63 | 62 | 47 | 47 | 17 | 18 | 18 | 16 | 16 | 16 | 17 | 19 | 16 | 17 |
| **4,000** | 781 | 63 | 93 | 47 | 62 | 62 | 47 | 62 | 62 | 63 | 18 | 18 | 20 | 17 | 17 | 17 | 17 | 18 | 16 | 19 |
| **8,000** | 391 | 78 | 63 | 78 | 78 | 46 | 109 | 125 | 79 | 93 | 23 | 17 | 18 | 16 | 18 | 18 | 17 | 19 | 16 | 17 |
| **16,000** | 438 | 62 | 62 | 62 | 63 | 79 | 47 | 63 | 46 | 63 | 19 | 20 | 18 | 17 | 20 | 18 | 18 | 18 | 17 | 18 |
| **32,000** | 562 | 63 | 62 | 63 | 63 | 62 | 62 | 78 | 63 | 62 | 19 | 18 | 20 | 38 | 17 | 18 | 18 | 18 | 18 | 19 |
| **64,000** | 1578 | 62 | 94 | 62 | 62 | 63 | 63 | 47 | 62 | 63 | 19 | 20 | 20 | 29 | 19 | 20 | 19 | 20 | 19 | 21 |
| **128,000** | 125 | 79 | 63 | 79 | 94 | 62 | 95 | 78 | 78 | 78 | 24 | 22 | 30 | 23 | 22 | 22 | 22 | 22 | 21 | 22 |
| **25,6000** | 782 | 77 | 62 | 109 | 78 | 94 | 62 | 62 | 63 | 78 | 40 | 27 | 27 | 32 | 27 | 27 | 31 | 27 | 27 | 28 |
| **512,000** | 1204 | 94 | 94 | 78 | 78 | 78 | 78 | 78 | 78 | 78 | 78 | 100 | 94 | 56 | 62 | 78 | 71 | 74 | 55 | 71 |
| **1,024,000** | 609 | 140 | 94 | 124 | 141 | 110 | 125 | 141 | 124 | 125 | 67 | 65 | 69 | 86 | 96 | 82 | 63 | 66 | 69 | 64 |
| **2,048,000** | 875 | 203 | 173 | 188 | 171 | 156 | 156 | 173 | 188 | 172 | 932 | 233 | 246 | 202 | 150 | 146 | 105 | 177 | 103 | 102 |
| **4096000** | 656 | 281 | 297 | 265 | 296 | 297 | 266 | 297 | 266 | 281 | 432 | 482 | 506 | 441 | 466 | 470 | 449 | 480 | 398 | 451 |
| **8192000** | 2468 | 437 | 500 | 485 | 500 | 516 | 470 | 516 | 500 | 484 | 960 | 765 | 817 | 830 | 876 | 849 | 821 | 906 | 854 | 840 |
| **16384000** | 2844 | 844 | 858 | 922 | 949 | 1250 | 875 | 937 | 875 | 891 | 2039 | 1544 | 1704 | 1634 | 1598 | 1565 | 1604 | 1648 | 1372 | 1526 |
| **32768000** | 5735 | 1766 | 1688 | 1703 | 1735 | 2313 | 1656 | 1704 | 1671 | 1656 | 3465 | 2757 | 3239 | 3265 | 3213 | 3168 | 3151 | 3195 | 2669 | 2955 |
| **65536000** | 4812 | 3359 | 3251 | 3297 | 3406 | 3454 | 3266 | 3234 | 3234 | 3375 | 5824 | 5338 | 5447 | 5343 | 5366 | 5419 | 5367 | 5382 | 4661 | 5086 |

**Averages 4 threads**

|  |  |  |  |
| --- | --- | --- | --- |
| **1,000** | 192.8 | 19.3 | 0 |
| **2,000** | 111.2 | 17 | 0 |
| **4,000** | 134.2 | 17.7 | 0 |
| **8,000** | 114 | 17.9 | 0 |
| **16,000** | 98.5 | 18.3 | 0 |
| **32,000** | 114 | 20.3 | 1 |
| **64,000** | 215.6 | 20.6 | 3 |
| **128,000** | 83.1 | 23 | 5 |
| **25,6000** | 146.7 | 29.3 | 10 |
| **512,000** | 193.8 | 73.9 | 27 |
| **1,024,000** | 173.3 | 72.7 | 84 |
| **2,048,000** | 245.5 | 239.6 | 152 |
| **4096000** | 320.2 | 457.5 | 248 |
| **8192000** | 687.6 | 851.8 | 772 |
| **16384000** | 1124.5 | 1623.4 | 1485 |
| **32768000** | 2162.7 | 3107.7 | 2786 |
| **65,536,000** | 3468.8 | 5323.3 | 5134 |

**Experiment 5:Random Data Set: 8 Threads ms**

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | **Windows 10**  **Virtual Machine** | **Ubuntu 16.04 LTS**  **Virtual Machine** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_\_\_\_\_\_RUN\_\_\_\_\_**  **Elements** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1,000** | 1703 | 108 | 124 | 109 | 125 | 110 | 125 | 125 | 108 | 125 | 40 | 44 | 33 | 34 | 35 | 34 | 42 | 30 | 73 | 33 |
| **2,000** | 1671 | 157 | 142 | 141 | 126 | 125 | 141 | 126 | 141 | 140 | 38 | 56 | 33 | 36 | 32 | 37 | 32 | 33 | 81 | 33 |
| **4,000** | 687 | 109 | 109 | 125 | 109 | 109 | 125 | 140 | 109 | 126 | 32 | 37 | 37 | 32 | 62 | 30 | 32 | 31 | 43 | 32 |
| **8,000** | 1939 | 141 | 140 | 140 | 140 | 126 | 125 | 125 | 126 | 140 | 41 | 33 | 37 | 38 | 32 | 31 | 35 | 33 | 87 | 59 |
| **16,000** | 453 | 250 | 125 | 126 | 126 | 109 | 140 | 125 | 124 | 132 | 35 | 45 | 34 | 34 | 33 | 32 | 33 | 33 | 75 | 34 |
| **32,000** | 359 | 109 | 125 | 125 | 139 | 140 | 125 | 141 | 126 | 133 | 45 | 51 | 52 | 51 | 69 | 49 | 80 | 34 | 34 | 66 |
| **64,000** | 313 | 125 | 132 | 140 | 126 | 125 | 126 | 139 | 125 | 126 | 59 | 39 | 49 | 80 | 64 | 78 | 34 | 82 | 62 | 48 |
| **128,000** | 5015 | 125 | 180 | 125 | 141 | 141 | 140 | 126 | 124 | 124 | 78 | 64 | 56 | 36 | 38 | 36 | 36 | 51 | 61 | 38 |
| **25,6000** | 3720 | 141 | 125 | 125 | 140 | 156 | 141 | 157 | 125 | 142 | 73 | 64 | 105 | 69 | 45 | 43 | 41 | 43 | 116 | 43 |
| **512,000** | 765 | 140 | 157 | 157 | 141 | 156 | 172 | 155 | 141 | 156 | 132 | 149 | 146 | 137 | 100 | 120 | 54 | 98 | 134 | 70 |
| **1,024,000** | 718 | 188 | 172 | 172 | 156 | 203 | 171 | 171 | 172 | 187 | 207 | 218 | 198 | 217 | 156 | 190 | 180 | 200 | 181 | 205 |
| **2,048,000** | 1000 | 547 | 265 | 282 | 235 | 234 | 235 | 249 | 218 | 251 | 249 | 237 | 262 | 316 | 282 | 254 | 324 | 316 | 277 | 335 |
| **4096000** | 563 | 359 | 359 | 344 | 344 | 344 | 376 | 344 | 328 | 344 | 462 | 431 | 507 | 546 | 465 | 506 | 514 | 564 | 442 | 544 |
| **8192000** | 876 | 594 | 563 | 578 | 515 | 561 | 532 | 595 | 547 | 532 | 841 | 860 | 1103 | 915 | 844 | 790 | 947 | 922 | 813 | 907 |
| **16384000** | 1360 | 968 | 969 | 922 | 1030 | 1039 | 953 | 1015 | 969 | 937 | 1861 | 1549 | 1664 | 1700 | 1478 | 1579 | 1842 | 1706 | 1404 | 1770 |
| **32768000** | 2780 | 1812 | 1750 | 1780 | 1735 | 1703 | 1781 | 1797 | 1750 | 1781 | 3202 | 2627 | 3275 | 3053 | 2398 | 3022 | 3286 | 3135 | 2678 | 3178 |
| **65536000** | 4625 | 3406 | 3360 | 3343 | 3312 | 3407 | 3345 | 3375 | 3328 | 3422 | 5409 | 5057 | 5229 | 5434 | 4937 | 5319 | 5335 | 5271 | 4924 | 5338 |

**Averages 8 threads**

|  |  |  |  |
| --- | --- | --- | --- |
| **1,000** | 107.63 | 39.8 | 0 |
| **2,000** | 291 | 41.1 | 0 |
| **4,000** | 174.8 | 76.8 | 0 |
| **8,000** | 314.2 | 42.6 | 0 |
| **16,000** | 171 | 38.8 | 0 |
| **32,000** | 152.2 | 53.1 | 1 |
| **64,000** | 147.7 | 59.5 | 3 |
| **128,000** | 624.1 | 49.4 | 5 |
| **25,6000** | 497.2 | 64.2 | 10 |
| **512,000** | 214 | 114 | 27 |
| **1,024,000** | 231 | 195.2 | 84 |
| **2,048,000** | 351.6 | 285.2 | 152 |
| **4096000** | 370.5 | 498.1 | 248 |
| **8192000** | 589.3 | 894.2 | 772 |
| **16384000** | 1016.2 | 1655.3 | 1485 |
| **32768000** | 1866.9 | 2985.4 | 2786 |
| **65,536,000** | 3492.3 | 5225.3 | 5134 |

Experiment 6 Random Data Set: 16 Threads All Runs, in MilliSeconds.

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | **Windows 10**  **Virtual Machine** | **Ubuntu 16.04 LTS**  **Virtual Machine** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_\_\_\_\_\_RUN\_\_\_\_\_**  **Elements** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1,000** | 546 | 282 | 250 | 250 | 251 | 249 | 250 | 266 | 250 | 234 | 69 | 90 | 93 | 96 | 61 | 64 | 61 | 63 | 60 | 59 |
| **2,000** | 485 | 297 | 234 | 313 | 249 | 235 | 235 | 218 | 251 | 267 | 82 | 65 | 71 | 62 | 63 | 85 | 69 | 62 | 61 | 93 |
| **4,000** | 860 | 250 | 266 | 1156 | 282 | 250 | 234 | 251 | 265 | 374 | 83 | 88 | 121 | 128 | 63 | 87 | 65 | 65 | 59 | 140 |
| **8,000** | 718 | 312 | 249 | 827 | 250 | 218 | 250 | 234 | 250 | 250 | 62 | 93 | 75 | 71 | 124 | 71 | 98 | 96 | 119 | 65 |
| **16,000** | 532 | 266 | 266 | 969 | 250 | 250 | 250 | 234 | 250 | 235 | 90 | 61 | 94 | 63 | 66 | 64 | 61 | 63 | 69 | 68 |
| **32,000** | 843 | 250 | 234 | 625 | 251 | 251 | 250 | 235 | 266 | 234 | 171 | 140 | 142 | 159 | 66 | 120 | 64 | 88 | 65 | 125 |
| **64,000** | 687 | 265 | 282 | 687 | 265 | 250 | 251 | 250 | 265 | 265 | 135 | 112 | 168 | 123 | 152 | 174 | 124 | 166 | 135 | 179 |
| **128,000** | 876 | 360 | 296 | 751 | 249 | 249 | 265 | 265 | 281 | 250 | 112 | 176 | 196 | 185 | 174 | 123 | 179 | 150 | 167 | 229 |
| **25,6000** | 846 | 249 | 282 | 282 | 267 | 313 | 266 | 266 | 250 | 251 | 195 | 197 | 117 | 176 | 202 | 192 | 178 | 193 | 179 | 158 |
| **512,000** | 1309 | 314 | 266 | 281 | 296 | 266 | 296 | 281 | 281 | 265 | 164 | 226 | 201 | 169 | 232 | 224 | 167 | 226 | 183 | 235 |
| **1,024,000** | 1329 | 312 | 313 | 327 | 313 | 296 | 297 | 313 | 298 | 296 | 312 | 276 | 285 | 268 | 293 | 344 | 263 | 296 | 266 | 306 |
| **2,048,000** | 1640 | 408 | 359 | 375 | 406 | 375 | 375 | 391 | 344 | 360 | 466 | 338 | 438 | 417 | 403 | 365 | 335 | 407 | 360 | 364 |
| **4096000** | 1531 | 500 | 469 | 562 | 484 | 470 | 500 | 484 | 469 | 515 | 529 | 587 | 482 | 515 | 620 | 576 | 526 | 635 | 570 | 573 |
| **8192000** | 1749 | 703 | 688 | 703 | 718 | 719 | 672 | 657 | 688 | 688 | 805 | 1021 | 982 | 1045 | 1112 | 985 | 871 | 1044 | 961 | 969 |
| **16384000** | 2109 | 1172 | 1109 | 1079 | 1093 | 1141 | 1078 | 1079 | 1109 | 1094 | 1699 | 1739 | 1443 | 1828 | 1940 | 1856 | 1381 | 1901 | 1568 | 1727 |
| **32768000** | 3703 | 2016 | 1922 | 1875 | 1922 | 1954 | 1906 | 1890 | 1876 | 1953 | 4022 | 3055 | 3126 | 3410 | 3485 | 3473 | 2976 | 3490 | 2783 | 3186 |
| **65536000** | 4563 | 3578 | 3687 | 3469 | 3485 | 3485 | 3374 | 3422 | 3562 | 3609 | 5205 | 5292 | 5186 | 5546 | 5656 | 5590 | 5271 | 5412 | 5056 | 5395 |

**Averages 16 threads**

|  |  |  |  |
| --- | --- | --- | --- |
| **1,000** | 282.8 | 71.6 | 0 |
| **2,000** | 278.4 | 71.3 | 0 |
| **4,000** | 426.6 | 89.9 | 0 |
| **8,000** | 355.8 | 87.4 | 0 |
| **16,000** | 325.2 | 69.9 | 0 |
| **32,000** | 343.9 | 114 | 1 |
| **64,000** | 346.7 | 146,8 | 3 |
| **128,000** | 384.2 | 169.1 | 5 |
| **25,6000** | 327.2 | 178.7 | 10 |
| **512,000** | 385.5 | 202.7 | 27 |
| **1,024,000** | 409.4 | 290.9 | 84 |
| **2,048,000** | 503.3 | 389.3 | 152 |
| **4096000** | 598.4 | 561.3 | 248 |
| **8192000** | 798.5 | 561.3 | 772 |
| **16384000** | 1206.3 | 1708.2 | 1485 |
| **32768000** | 2101.7 | 3300.6 | 2786 |
| **65,536,000** | 3623.4 | 5360.9 | 5134 |

**Average Turnaround (execution) Time (seconds)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Windows 7 Enterprise | Windows 8.1 Pro | Ubuntu 14.04 LTS |
| Experiment 1 | 931.39 | 952.78 | 728.95 |
| Experiment 2 | 8127.39 | 8176.68 | 5913.24 |
| Experiment 3 | 381.96 | 375.95 | 285.36 |
| Experiment 4 | 4.11 | 4.318 | 2.86 |

**Standard deviation (seconds)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Windows 7 Enterprise | Windows 8.1 Pro | Ubuntu 14.04 LTS |
| Experiment 1 | 7.492 | 28.339 | 16.11 |
| Experiment 2 | 19.68 | 16.52 | 82.17 |
| Experiment 3 | 4.084 | 2.686 | 2.618 |
| Experiment 4 | 0.0624 | 0.07315 | 0.089 |

**APPENDIX E**

**Source Code**