COMP 464 - High Performance Computing
High Frequency Trading Data Processing

Loyola University Chicago
Jose Luis Rodriguez

December 13, 2017

1 Overview

As computational capabilities continue to grow exponentially and financial markets around the world are increasingly dependent on automated systems, it is important to spend time analyzing market raw transactional data. Two approaches were developed - an openMP C++ application and a Spark/Hadoop application. Given the nature of the data (millions of transactions) it was necessary to implement the search, map, and reduce operations in parallel in order to efficiently process the market data. This report found that the openMP approach is easily scaleable and compatible with a multitude of hardware, despite its need for a more rigorous evaluation of code parallelization regions.

The Spark/Hadoop approach can be seen as the go-to tool for large datasets (big data analysis) and was indeed easier to manipulate in so far as uploading data and conducting a preliminary analysis. With that said, the main drawback of the Spark/Hadoop approach is its lack of easy transferability. As soon as the Hadoop cluster is turned off, this approach is no longer functional. After the development, testing, and benchmark of the applications, the primary goal is to analyze the events leading up to the May 6, 2010 market flash crash. Preliminary findings on this project could be compared with the official report to congress from the CFTC and the SEC for additional work in the future. The data used in this project is from the CME Group and corresponds to market depth transactions in milliseconds of the E-Mini S&P 500 futures and options contracts.

2 Methodology

According to the CME Group Data Mine website, their market depth data provides all market data messages required to recreate the order book (list of orders that a trading firm uses to record the interest of buyers and sellers in a particular financial instrument). Each message contains between five to ten orders deep in futures markets and three orders deep in options markets. This data is timestamped to the millisecond, allowing for an in-depth analysis of the price and volume movement.

To analyze the volume and price variation three main functions in openMP were utilized to first read the data into memory, search for a given tag that represents a value in the data, and finally (using the volume function) aggregate the results from the search method. These functions utilize openMP parallel pragmas in order to efficiently process the data. For benchmark purposes, different numbers of threads were used in order to identify the superior performance. For the Spark/Hadoop setup, the data reading process was different given the nature of the Hadoop. The data needed to be copied to the Hadoop cluster, then Spark was utilized for the analysis. While in Spark, a function to search for a given tag was created, and the results of the function were counted utilizing Spark parallelism features.

3 Performance

To measure performance, scalability metrics can be used in each of the map, search, reduce, and filtering operations. The performance measures are different depending upon the approach used. For openMP, it is possible to measure a number of performance metrics within the program, such as the time it takes to read a file into memory (RAM), the time it takes for a given number of threads to search a tag, and the time it takes to process and aggregate the results. In each of these functions, the time was recorded and a log was created. For the Spark/Hadoop approach, timing the operations was more direct as it is simply a matter of timing how long it takes to run a function. With that said, the Spark/Hadoop approach lacks the flexibility possessed by the openMP approach. A table and plots of the results of each approach can be found in the appendix. Further scaleability testing was possible utilizing the openMP approach as the number of threads can be increased to allow for quicker data processing.

4 Conclusion

The aim of this report was to highlight methods to approach high frequency trading data (raw market data) that can be utilized to conduct system and strategies backtesting and forensics of market transactions data. Addressing large transactional data sizes presents challenges in any setting, from transferring to opening, and even regarding storage resources. In many cases, raw financial transactional data is not stored in a common data type. Special search functions must be implemented in order to parse the data and make the data usable. Finally, having access to raw transaction data is proven to be a very challenging task, as these datasets are very costly and the amount of time and resources that these datasets demand can be prohibitive for companies and/or researchers. Nonetheless, it is in the best interest of all that applications such as the one built in this project are created and are open source. There is an opportunity to create a robust community that could lead the charge on increasing transparency, education, and training in the high frequency trading industry.

5 Reference

TACC Wrangler User Guide – The Wrangler Data Analysis and Storage System
`https://portal.tacc.utexas.edu/user-guides/wrangler`

Stampede2 User Guide – Supercomputer at the Texas Advanced Computing Center
`https://portal.tacc.utexas.edu/user-guides/stampede2`

Introduction to High Performance Scientific Computing – Victor Eijkhout
`http://pages.tacc.utexas.edu/~eijkhout/istc/istc.html`

Kirilenko, Andrei; Kyle, Albert S.; Samadi, Mehrdad; Tuzun, Tugkan (May 5, 2014), The Flash Crash: The Impact of High Frequency Trading on an Electronic Market (PDF), retrieved 8 November 2017
`http://www.cftc.gov/idc/groups/public/@economicanalysis/documents/file/oce_flashcrash0314.pdf`

Findings Regarding the Market Events of May 6, 2010 (Rep.). (2010, September 20). Retrieved December 12, 2017, from U.S. Securities & Exchange Commission website: https://www.sec.gov/news/studies/2010/marketevents-report.pdf

6 Appendix

Figure 1: Read [50,846,717] - 05/02/10 - 05/08/10 - Speed-Up (T1/Tp)

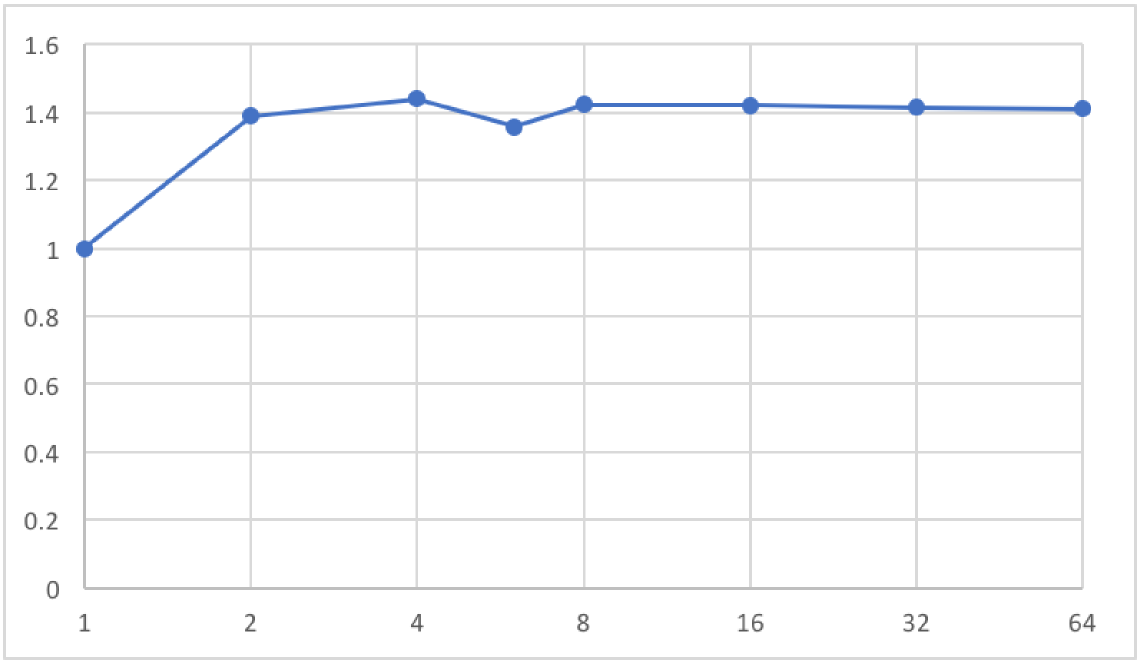


Figure 2: Read [50,846,717] - 05/02/10 - 05/08/10 - Efficiency (T1/Tp)

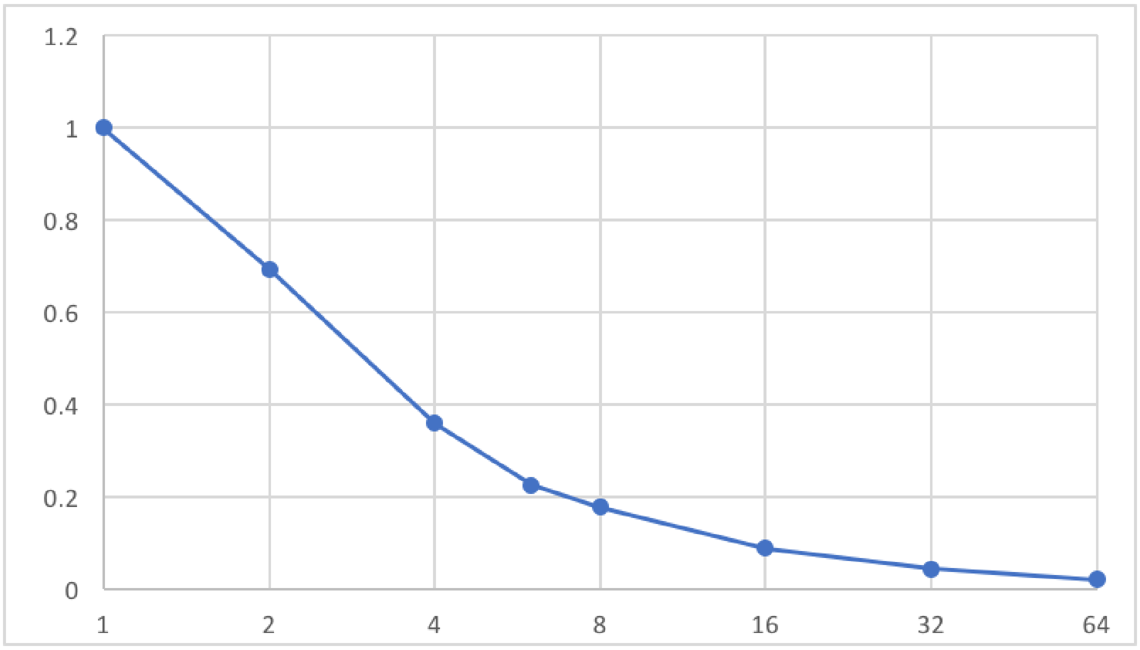


Figure 3: Search [50,846,717] - 05/02/10 - 05/08/10 - Speed-Up (T1/Tp)

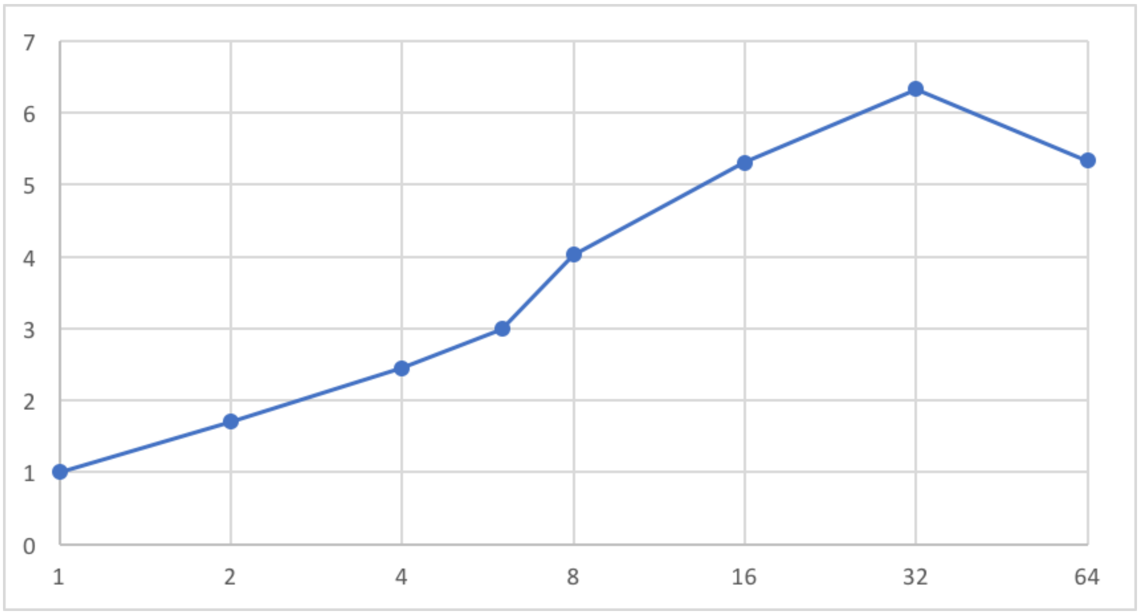


Figure 4: Search [50,846,717] - 05/02/10 - 05/08/10 - Efficiency (T1/Tp)

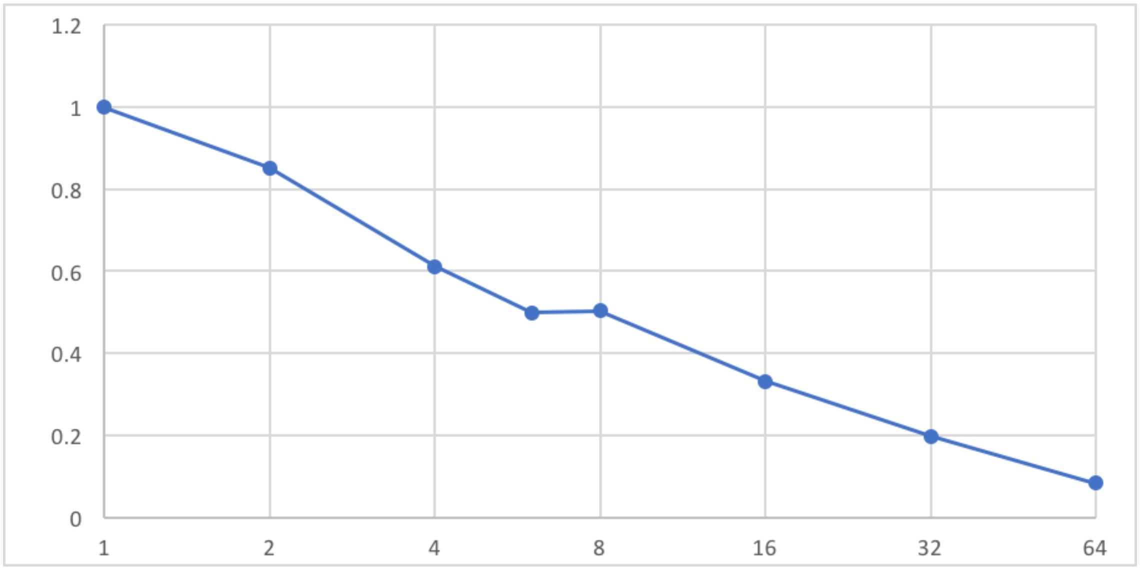


Figure 5: Volume [50,846,717] - 05/02/10 - 05/08/10 - Speed-Up (T1/Tp)

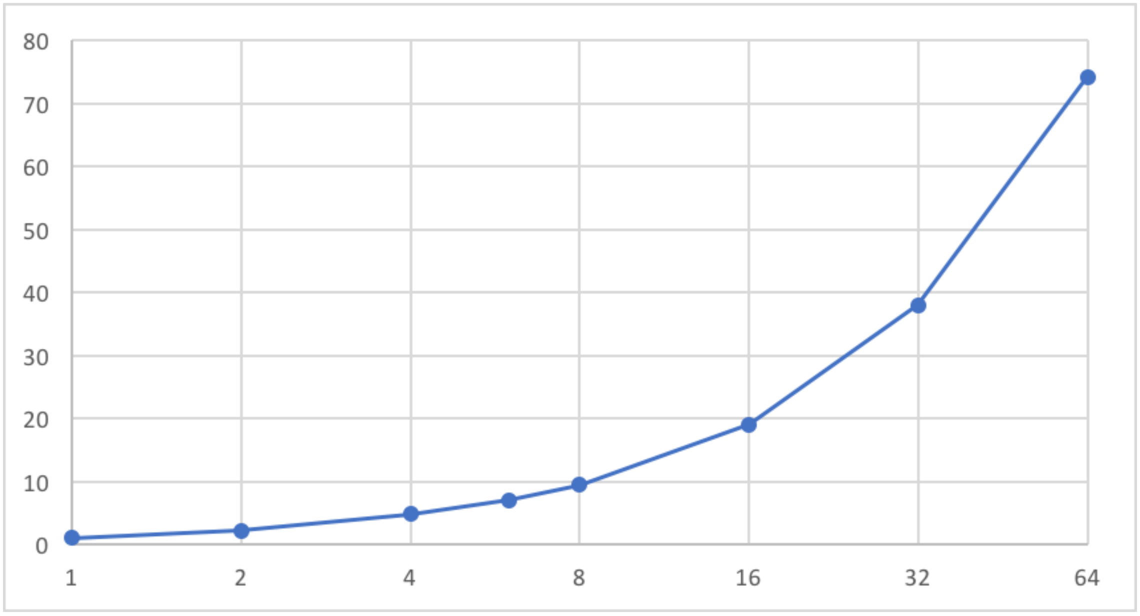


Figure 6: Volume [50,846,717] - 05/02/10 - 05/08/10 - Efficiency (T1/Tp)

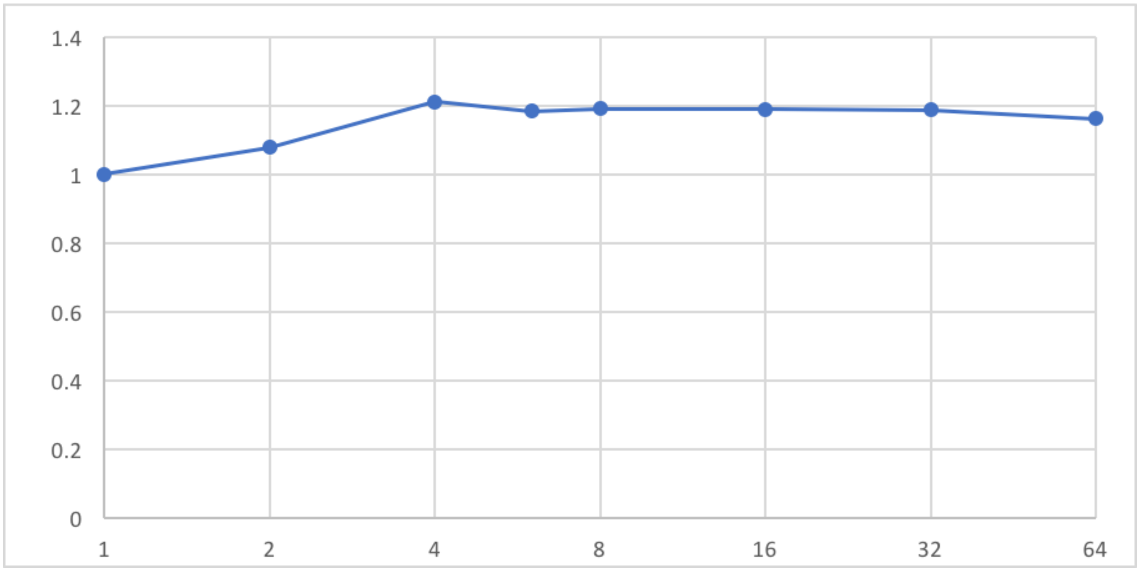


Figure 7: Read [43,593,190] - 05/09/10 - 05/15/10 - Speed-Up (T1/Tp)

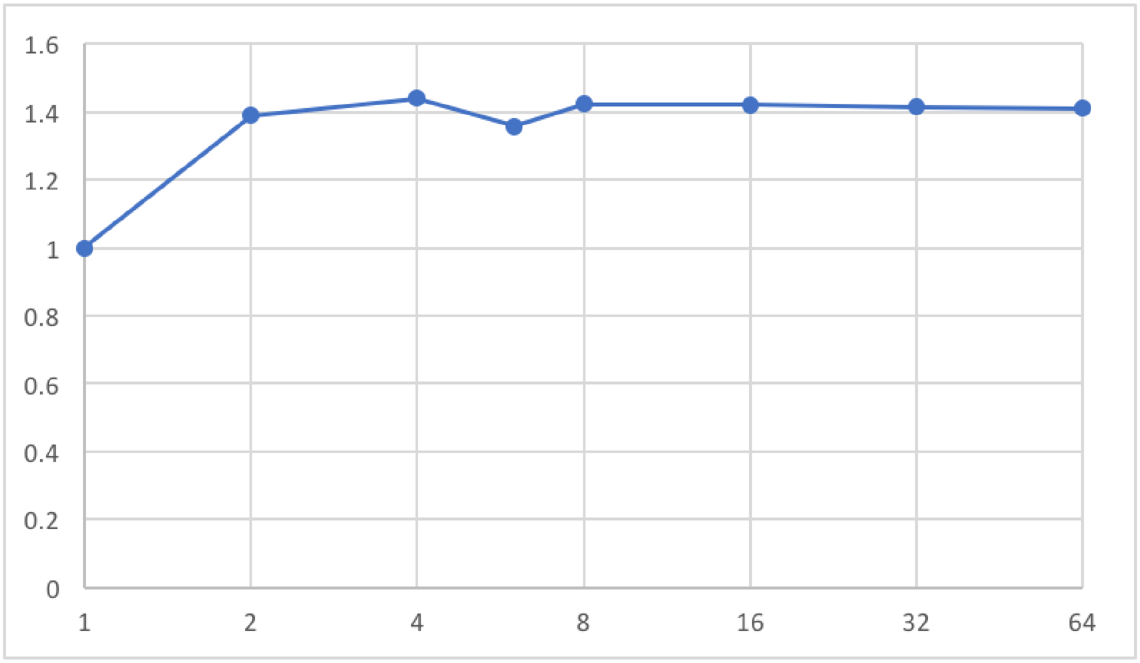


Figure 8: Read [43,593,190] - 05/09/10 - 05/15/10 - Efficiency (T1/Tp)

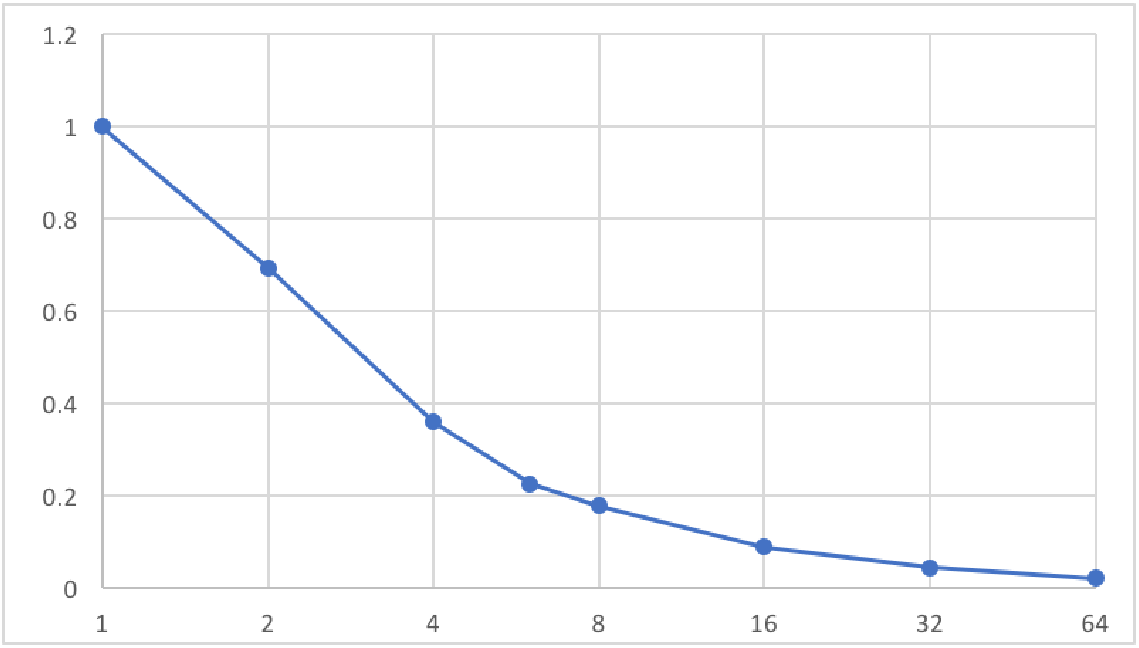


Figure 9: Search [43,593,190] - 05/09/10 - 05/15/10 - Speed-Up (T1/Tp)

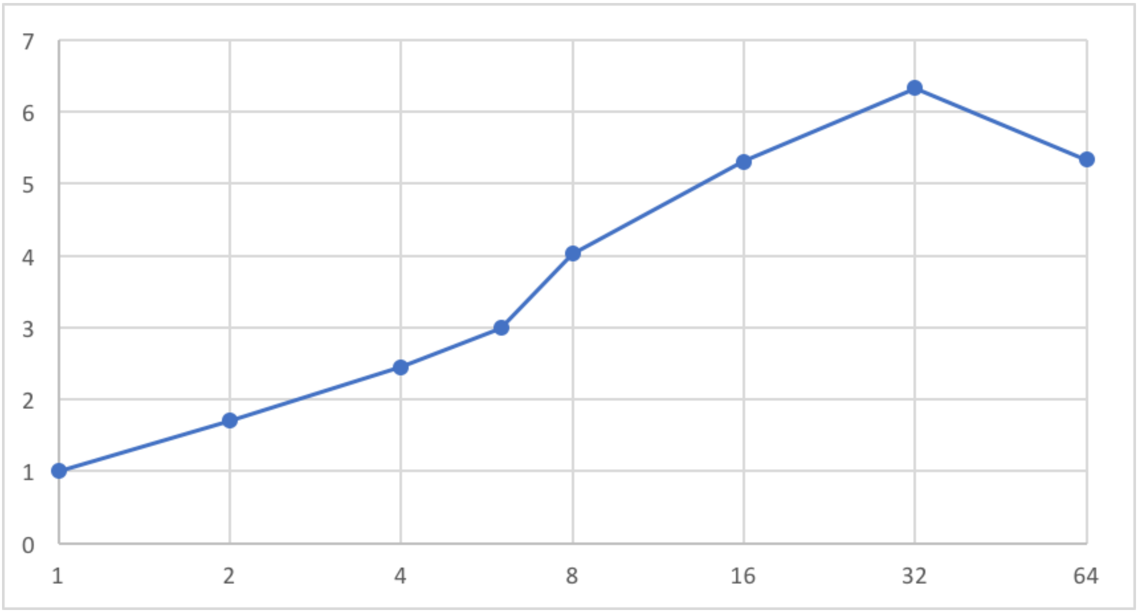


Figure 10: Search [43,593,190] - 05/09/10 - 05/15/10 - Efficiency (T1/Tp)

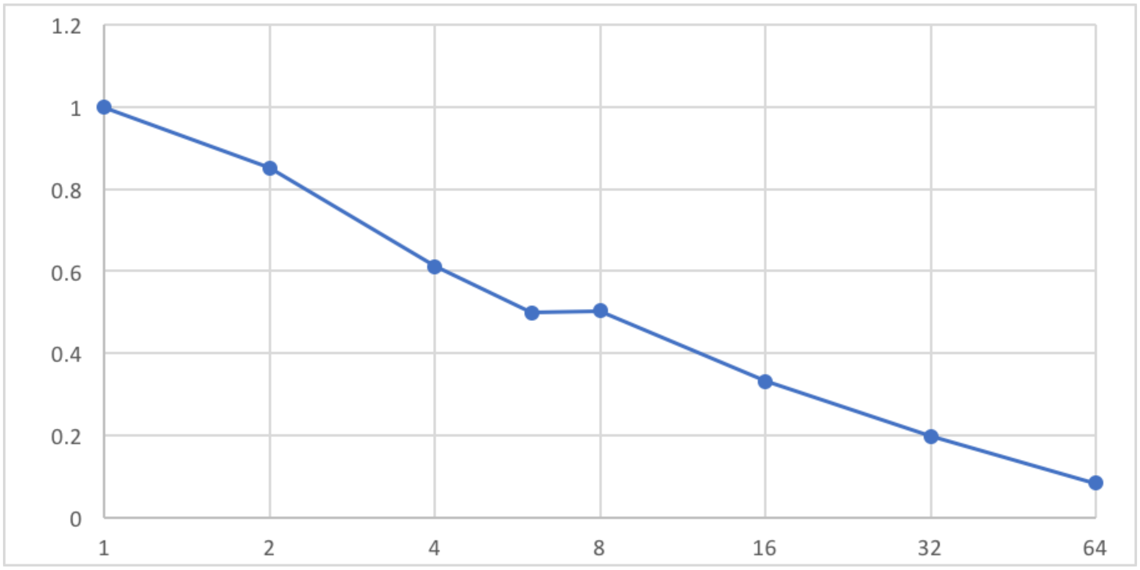


Figure 11: Volume [43,593,190] - 05/09/10 - 05/15/10 - Speed-Up (T1/Tp)

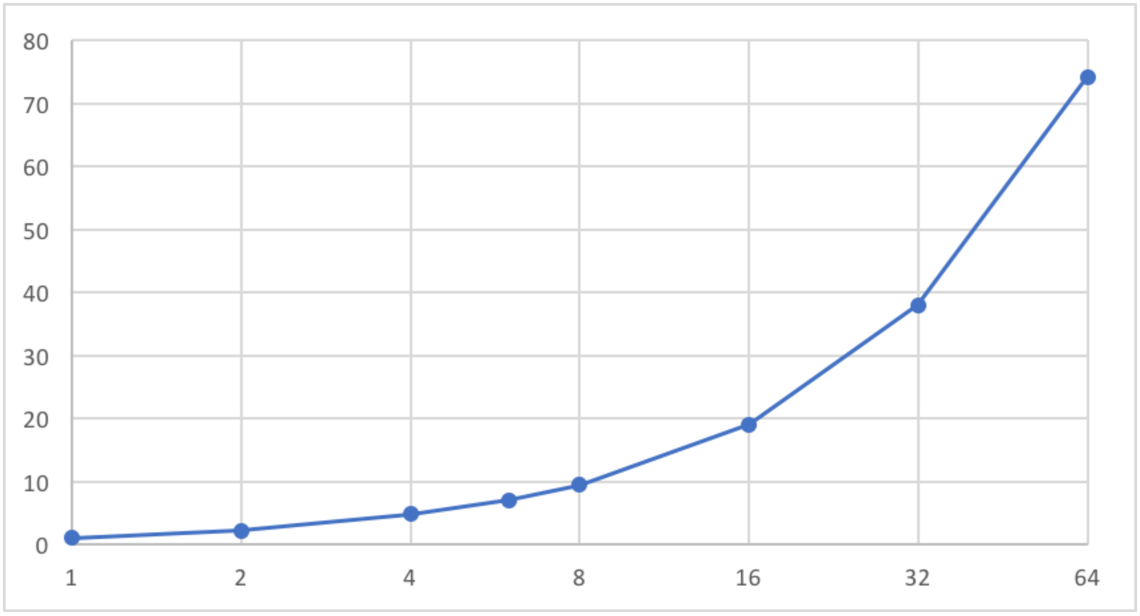


Figure 12: Volume [43,593,190] - 05/09/10 - 05/15/10 - Efficiency (T1/Tp)

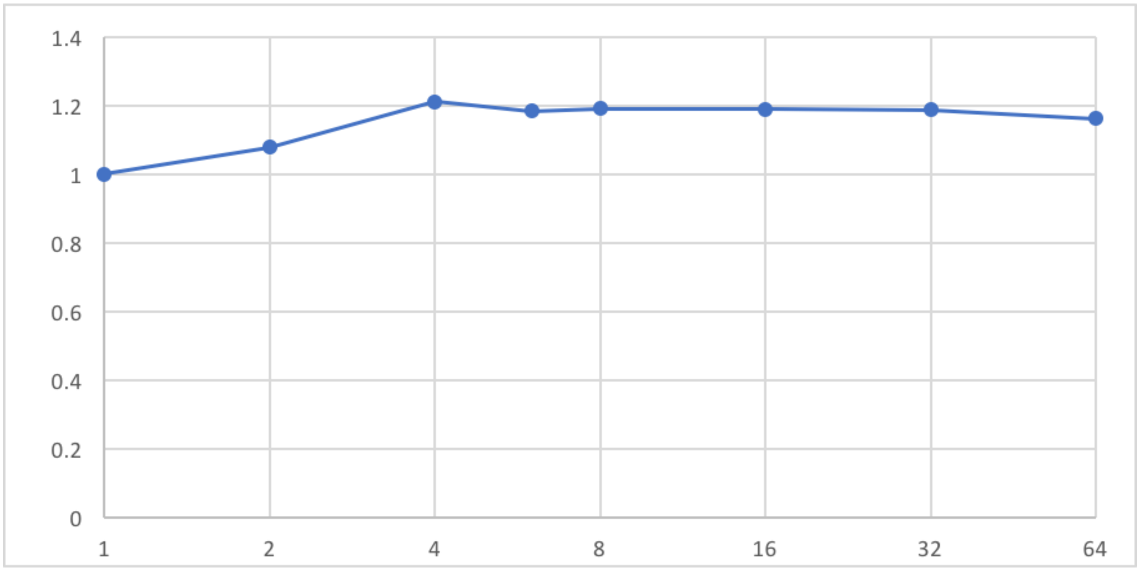


Figure 13: Read [62,586,111]- 05/16/10 - 05/22/10 - Speed-Up (T1/Tp)

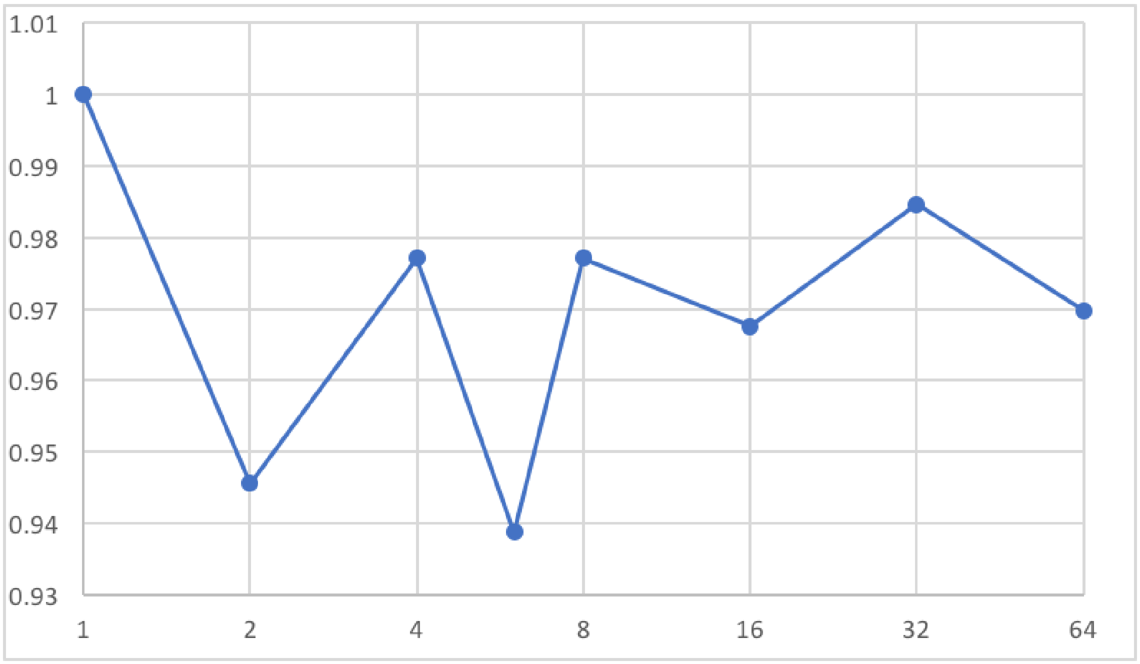


Figure 14: Read [62,586,111]- 05/16/10 - 05/22/10 - Efficiency (T1/Tp)

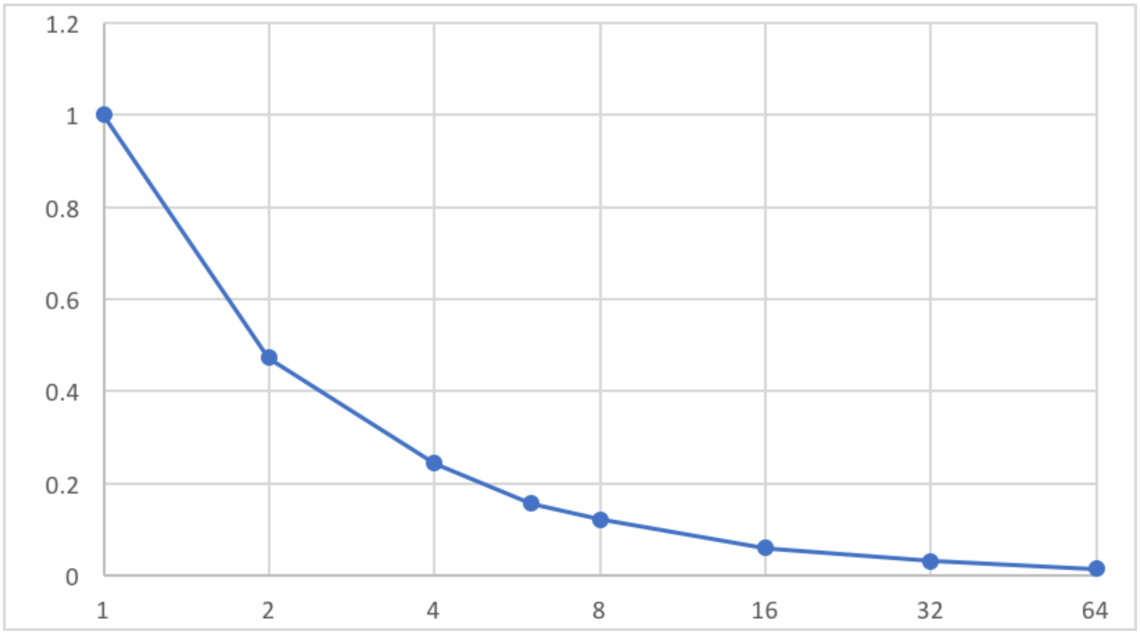


Figure 15: Search [62,586,111]- 05/16/10 - 05/22/10 - Speed-Up (T1/Tp)

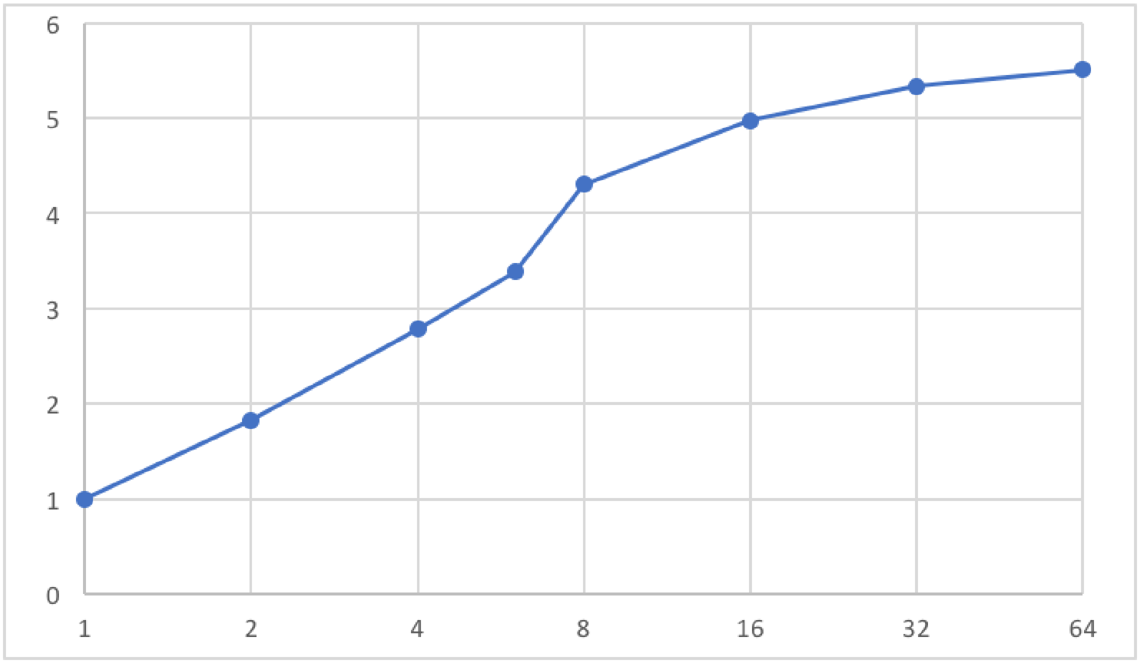


Figure 16: Search [62,586,111]- 05/16/10 - 05/22/10 - Efficiency (T1/Tp)

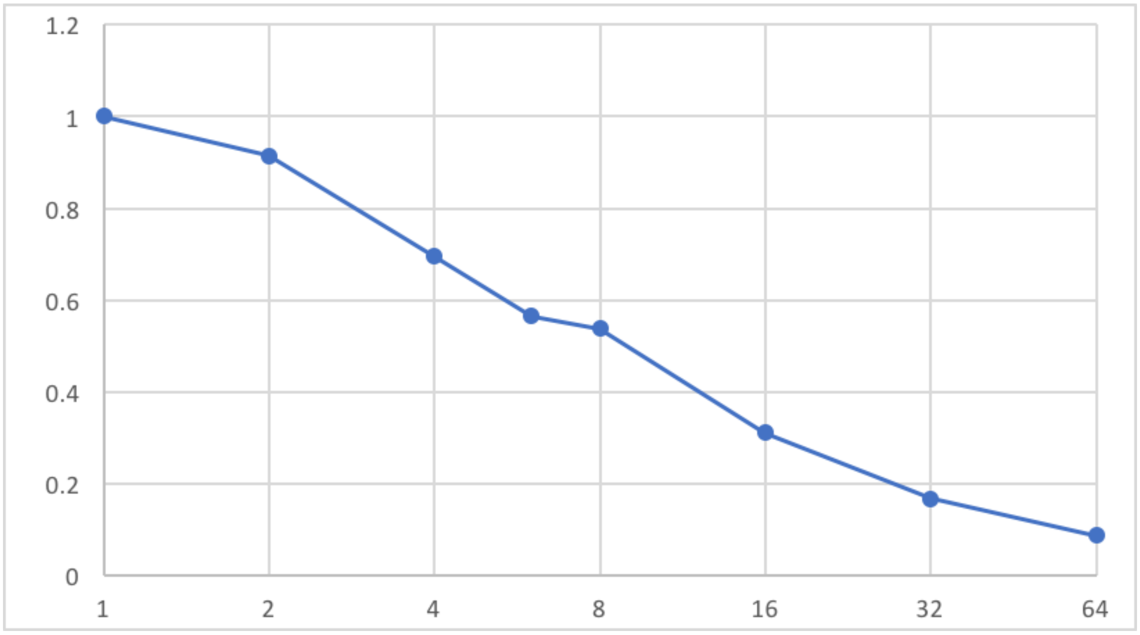


Figure 17: Volume [62,586,111]- 05/16/10 - 05/22/10 - Speed-Up (T1/Tp)

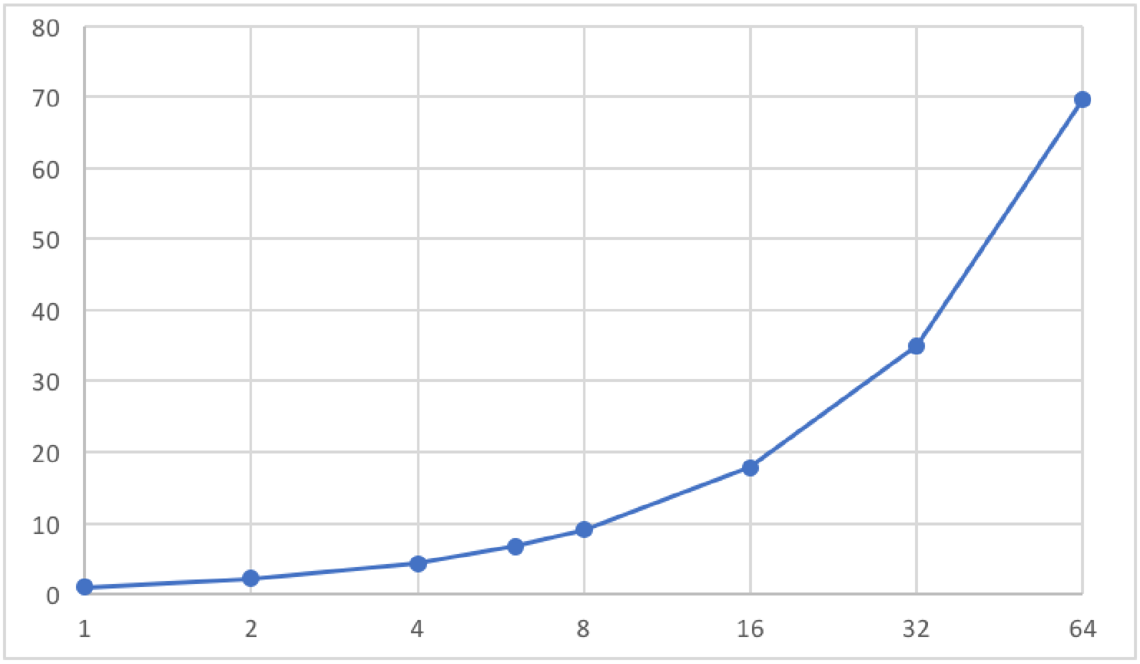


Figure 18: Volume [62,586,111]- 05/16/10 - 05/22/10 - Efficiency (T1/Tp)

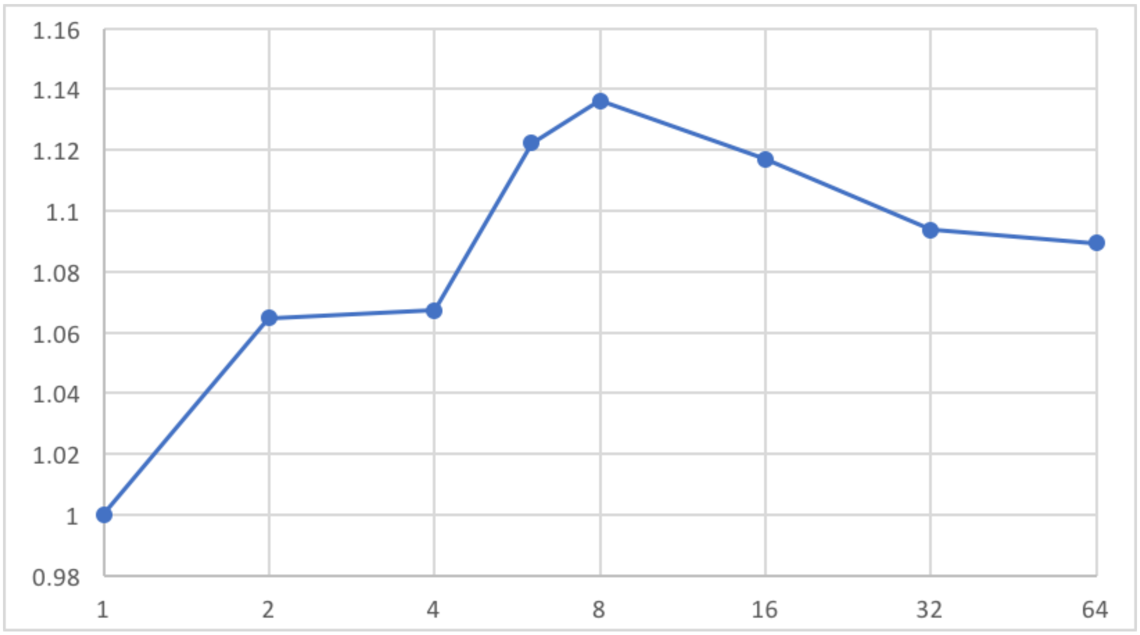


Figure 19: Read [41,407,982]- 05/23/10 - 05/30/10 - Speed-Up (T1/Tp)

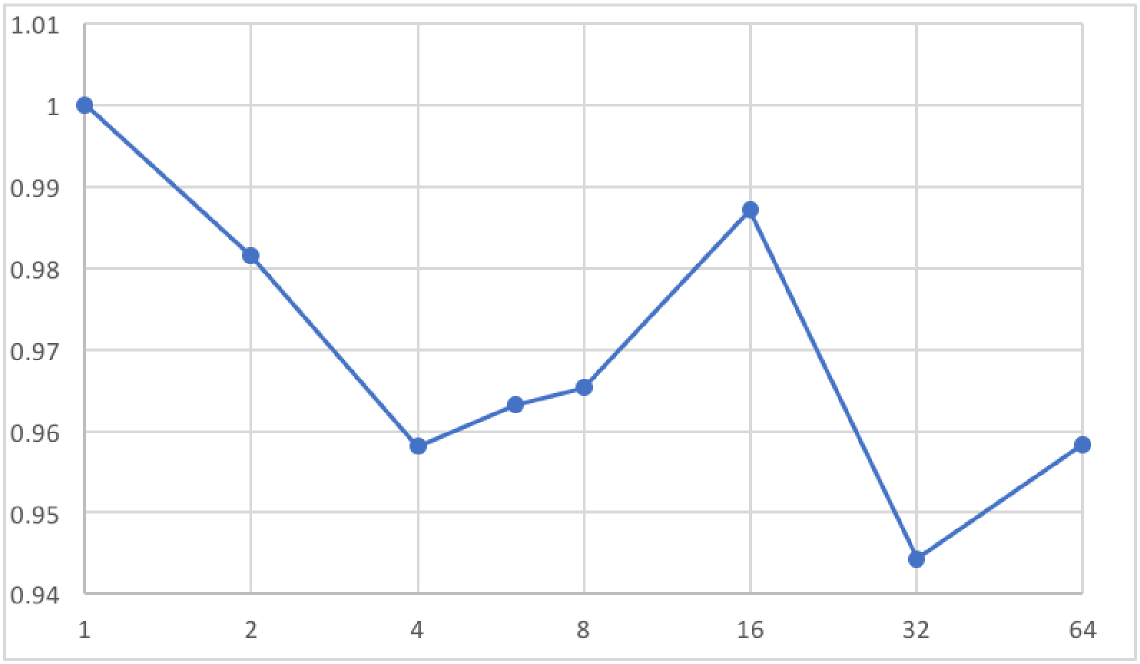


Figure 20: Read [41,407,982]- 05/23/10 - 05/30/10 - Efficiency (T1/Tp)

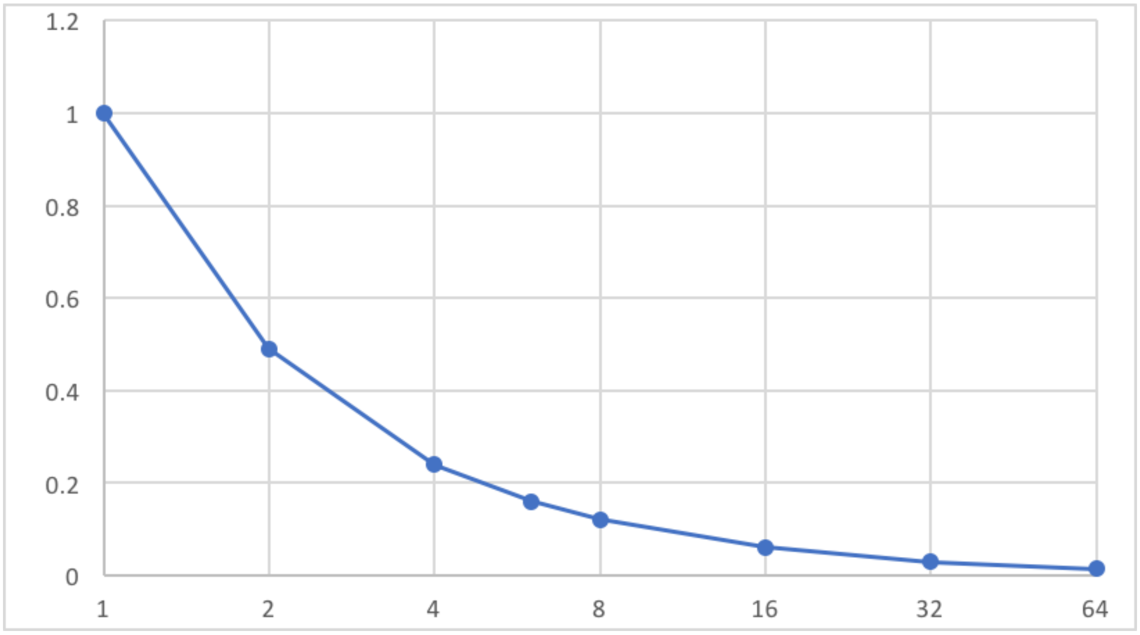


Figure 21: Search [41,407,982]- 05/16/10 - 05/22/10 - Speed-Up (T1/Tp)

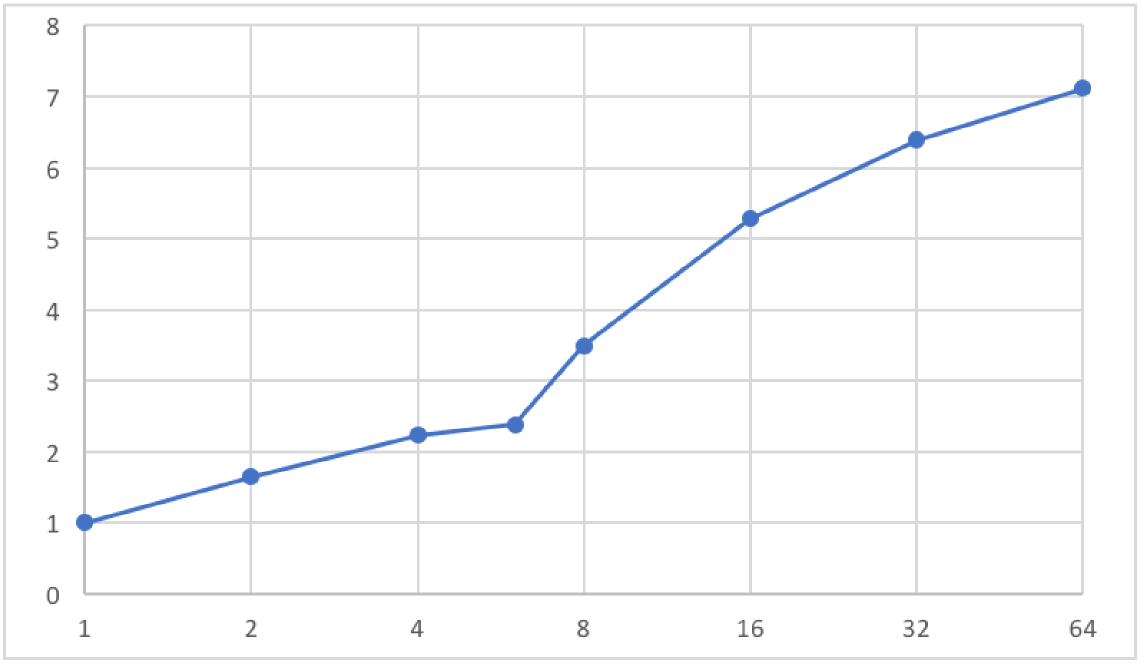


Figure 22: Search [41,407,982]- 05/16/10 - 05/22/10 - Efficiency (T1/Tp)

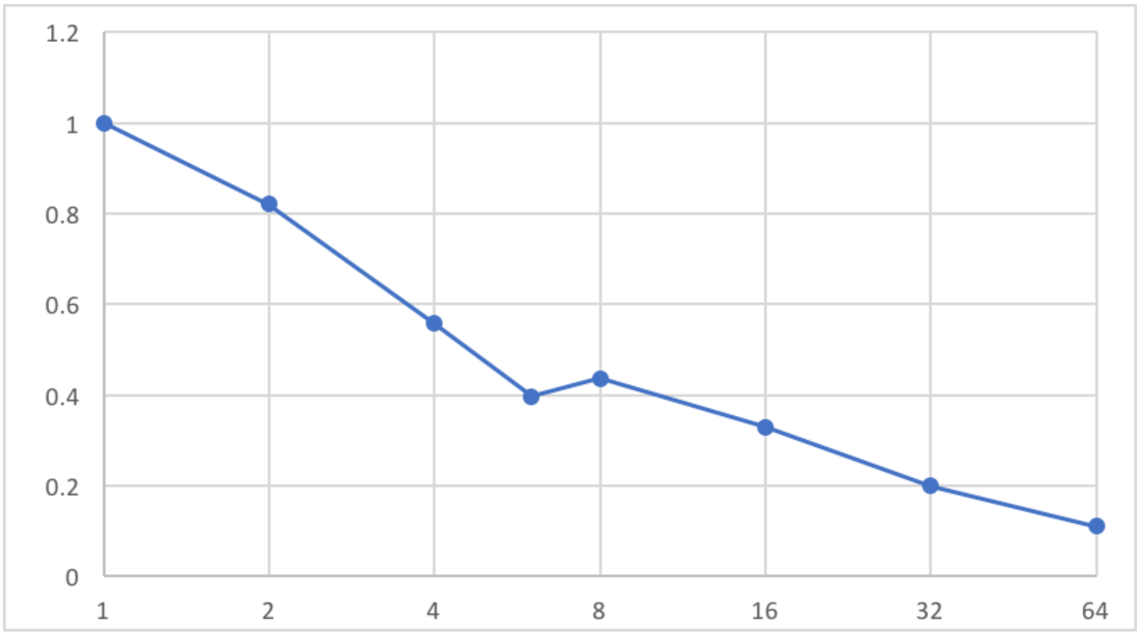


Figure 23: Volume [41,407,982]- 05/16/10 - 05/22/10 - Speed-Up (T1/Tp)

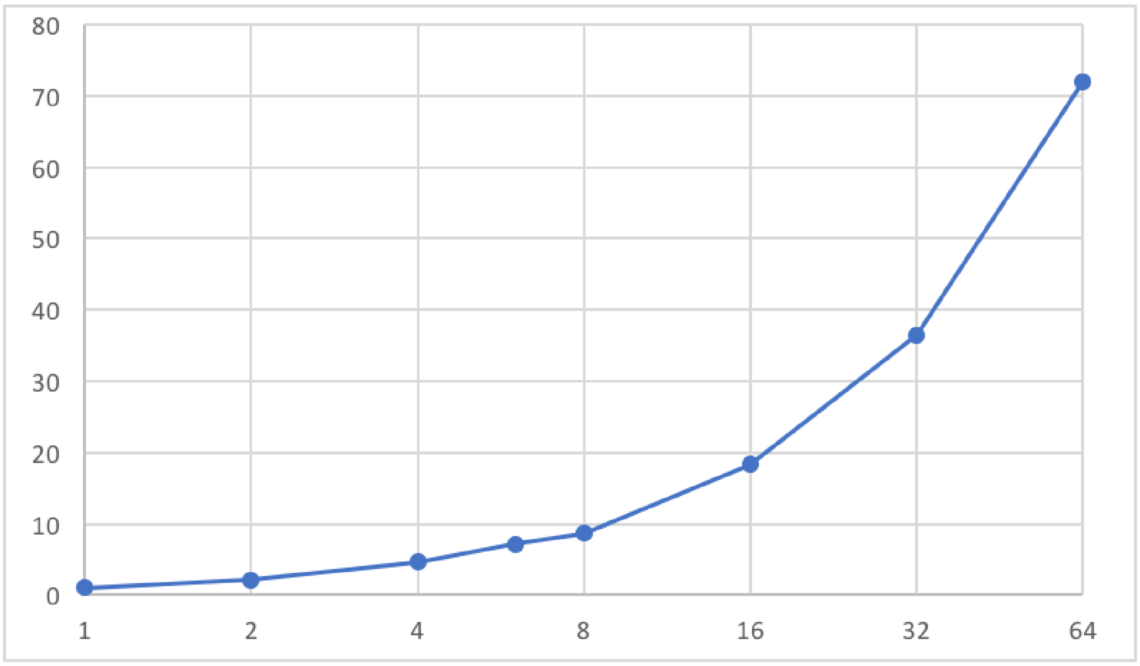


Figure 24: Volume [41,407,982]- 05/16/10 - 05/22/10 - Efficiency (T1/Tp)

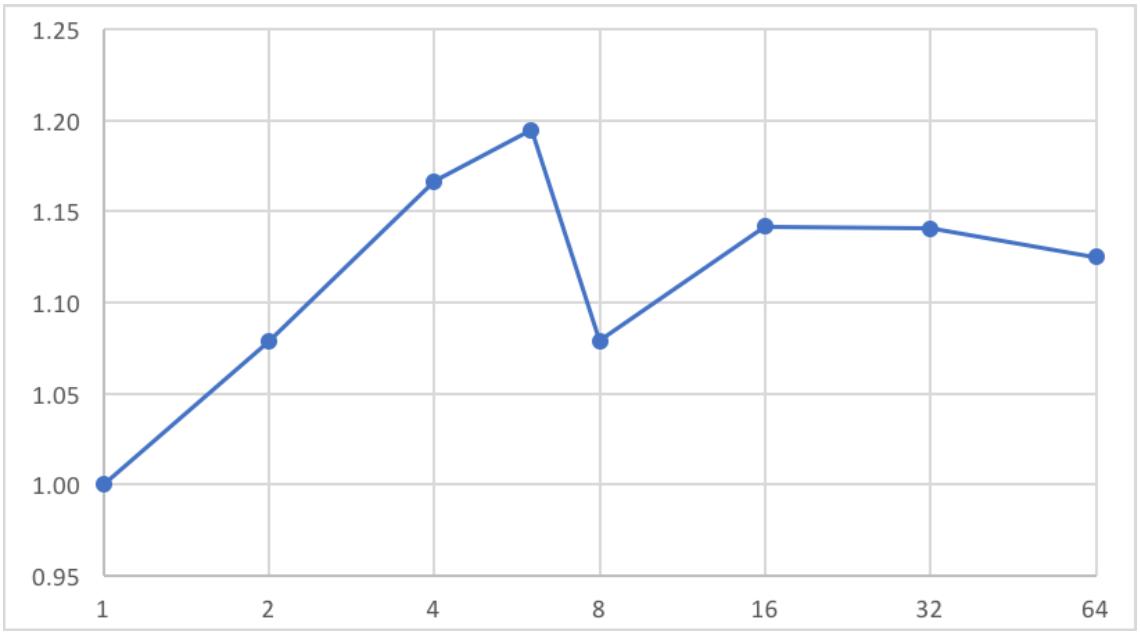


Figure 25: Read [31,720,486]- 05/30/10 - 06/03/10 - Speed-Up (T1/Tp)

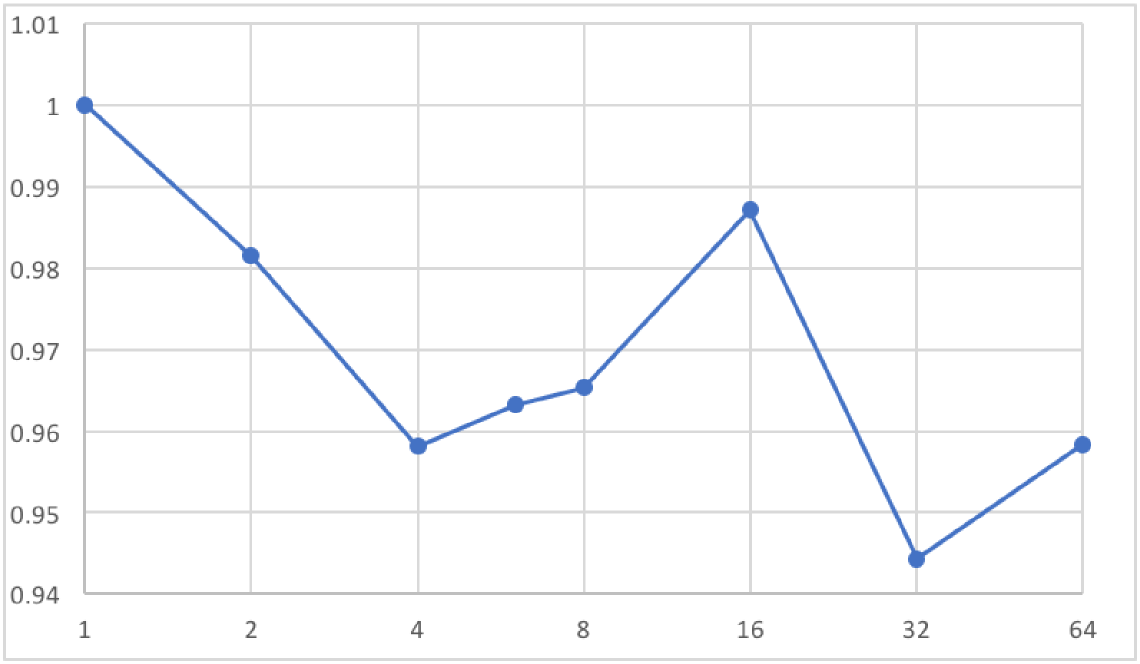


Figure 26: Read [31,720,486]- 05/30/10 - 06/03/10 - Efficiency (T1/Tp)

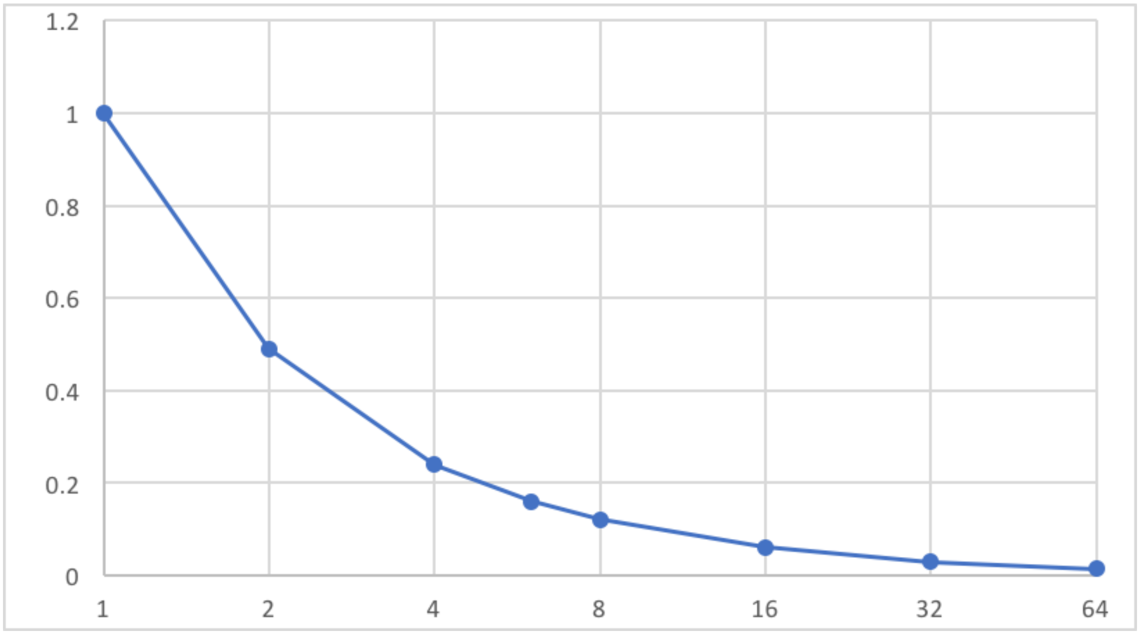


Figure 27: Search [31,720,486]- 05/30/10 - 06/03/10 - Speed-Up (T1/Tp)

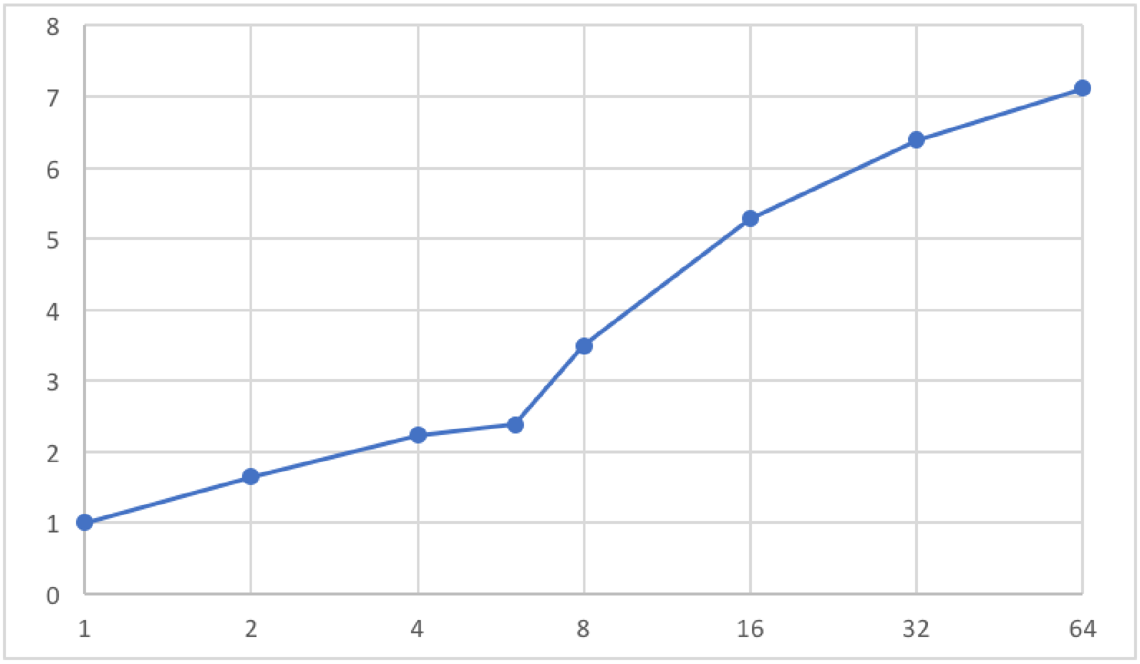


Figure 28: Search [31,720,486]- 05/30/10 - 06/03/10 - Efficiency (T1/Tp)

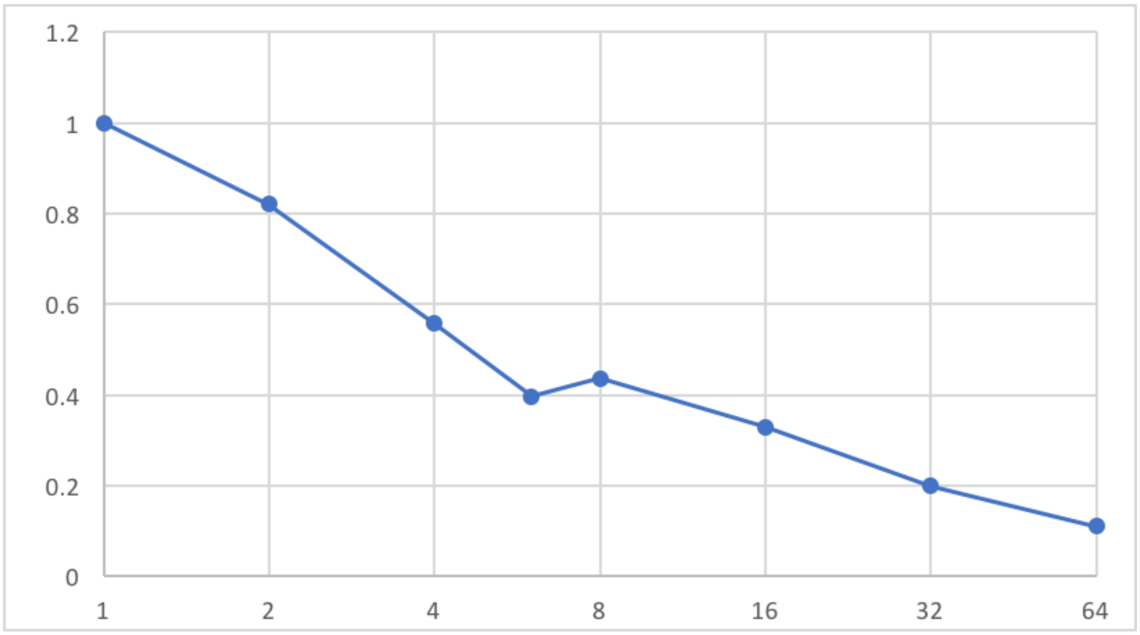


Figure 29: Volume [31,720,486]- 05/30/10 - 06/03/10 - Speed-Up (T1/Tp)

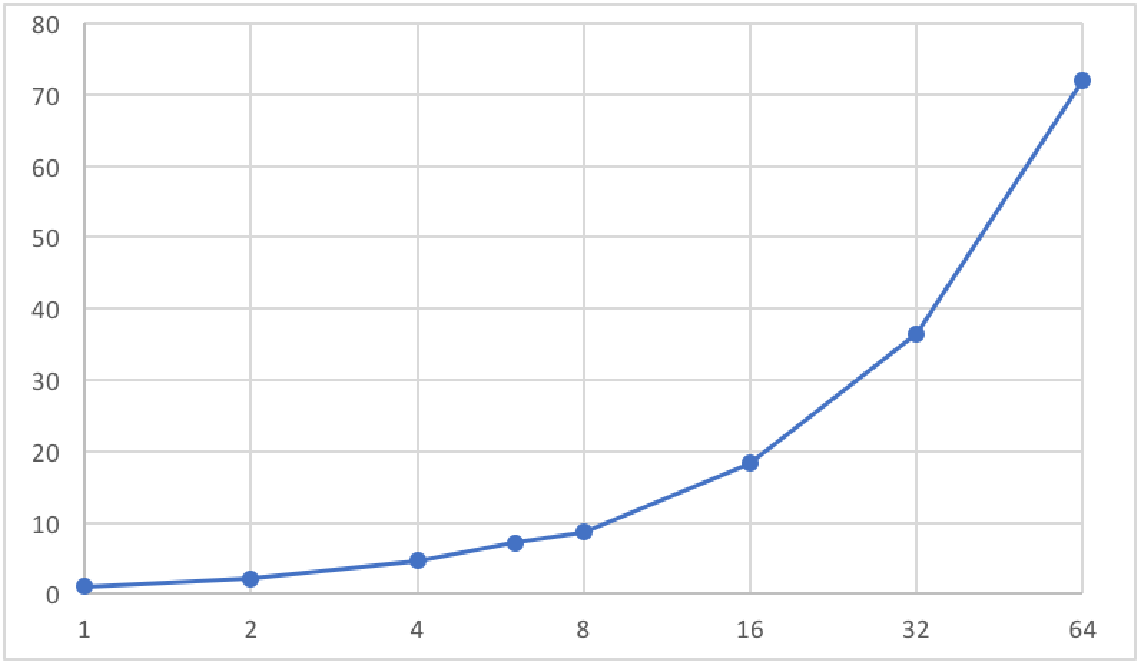
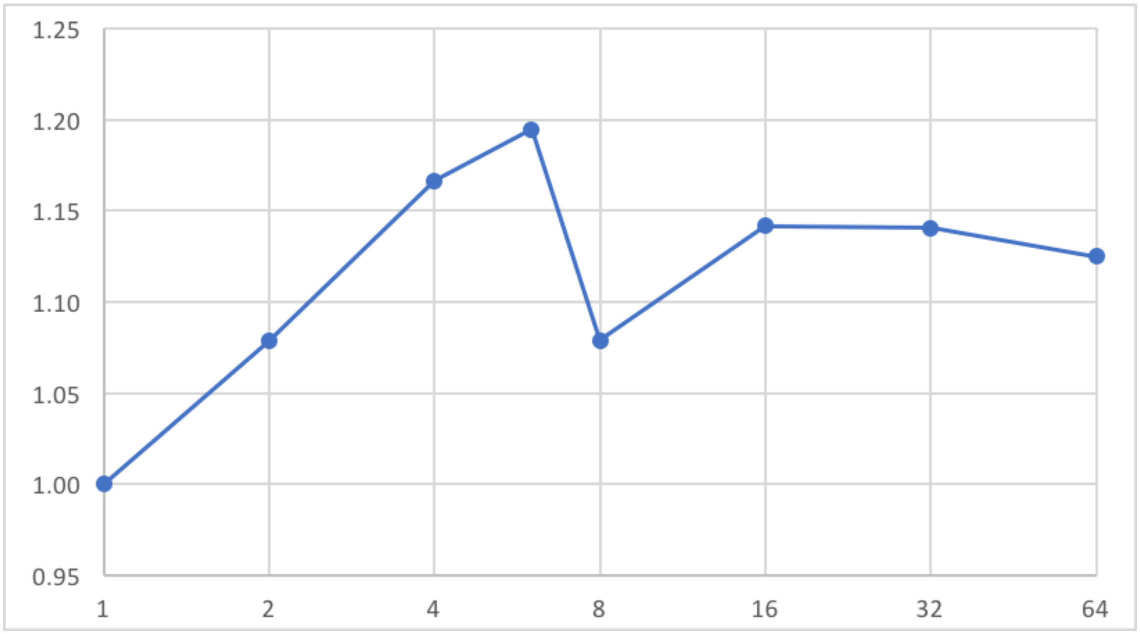


Figure 30: Volume [31,720,486]- 05/30/10 - 06/03/10 - Efficiency (T1/Tp)


1.25
1.20
1.15
1.10
1.05
1.00
0.95
1
2
4
8
16
32
64