Algorithm Engineering
WS 22/23

Sanket Datta
Toru Tsukamoto

# External Memory Merge Sort

## Abstract

We implement and compare the practical performance of two different variants of a well known sorting algorithms, namely, Classic merge sort and External merge sort algorithm. We first explain the basic algorithms, then test them with different size of data and finally compare the performance of each of them. We test both methods with different sizes of randomly generated numbers. The result show that External merge sort is not as efficient as classic merge sort, but it was able to sort data beyond the memory size, which does not work with classic merge sort.

## 1 Introduction

Many algorithms exist for sorting like bubble sort, insertion sort, quick sort, merge sort etc. In real world sorting data is a very big part of life where tech giant strive to sort billions of data every moment for getting various insights from the data. As time progressed and data became larger, new sorting methods came up to tackle with the enormous amount of data. In this report we compare classic merge sort method and the external memory merge sort method with a focus of their practical performance with different kind of data loads.
The remainder of this work is structured as follows. Section 2 will explain the necessary background information regarding the experiment. Section 3 will explain the two algorithms, provides information about the used algorithm engineering techniques and gives some implementation details. In Section 4, we describe the experimental setup and provide empirically results. In addition, we discuss the Data and Hardwares used for this experiment. Then we compare the stats using data visualization. Finally, in section 5, we discuss the outcomes.

## 2 Preliminaries

We consider the following problem: Given a set of randomly generated integers, How does the merge sort and External merge sort algorithm performs. We will briefly recap these two algorithms in the following.
Merge Sort: The Merge Sort algorithm is a sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two equal halves again and again and then they are combined in a sorted manner.
External merge sort: External sorting is a term for a class of sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted does not fit into the main memory of a computing device (usually RAM) and instead, must reside in the slower external memory (usually a hard drive).

## 3 Algorithm & Implementation

At first we generate a series of random integers and create a file with that. For generating random integers we used numpy library.
We start testing with classic merge sort algorithm by reading the file at first and storing all numbers in an array. After that, the merge sort process takes theoretical running time(O(nlogn)) to finish the sorting and stores them back in another array. Finally, It is written to the output file.
For External merge sort, After file generation, the file is read in particular size of chunks and it is sorted using quick sort algorithm(In place). After sorting the chunks it is written in the output file buffers. After that, we picked up two chunks and started doing two way merge in those two chunks of integers. We kept pushing the integer output of merge to an empty array. After output buffer reaches a certain limit. It

directly writes in the output file. This process is carried out for all chunks in sequence. Then double the size of the chunks and repeat the same process. When the chunk size is equal to the original data, the sorting is complete.
We used memory profiler to count the time taken to calculate various test runs for the whole execution of merge sort and external merge sort.

# 4 Experimental Evaluation

## 4.1 Computer Specifications

Table 1 shows the specifications of the computers used in the experiment.

Table 1: Computer Specifications

| | |
|---|---|
| Chip | Apple M1 Pro |
| Core | 8 CPU, 14GPU, 16 Neural Engine |
| Memory | LPDDR5 16GB |
| Storage | APPLE SSD |

Then used docker virtual machine to replicate smaller ram size to test the data. For the first experiment, The memory size was set to 64 MB for the first experiment and later one tested with 512 MB of memory. We couldn't test it with lower ram size because a certain amount of memory was necessary to avoid failures in the environment construction.

## 4.2 Results

Data sizes of 4MB, 8MB, 32MB, 64MB and 128MB were tried. For the first run(64 MB RAM): As the calculation time was taking more than 20 minutes at 128 MB for the external merge sort, we decided not to increase the data size any further. Using the memory profiler further increased the calculation time by a factor of 10 or more, so we could only measure memory usage to a certain extent.
On the other hand, internal sorting only worked with 4 MB (5 percent). We use Python lists for storing data, but the amount of data may easily become large. Memory is also used to run Python and other libraries. Also, classic merge sort needs extra n space to run the algorithm for which It couldn't run with more than that data size.
For the second run(512 MB RAM): Sorting could be performed with 32 MB (5 percent), but not with 64 MB due to insufficient memory.
As shown in Fig 1, for 4 MB of data, Classic merge sort was about eight times faster than External marge sort. However, for larger data, Classic merge sort could not be performed due to insufficient memory. External marge sort, on the other hand, was able to perform sorting on 128 MB of data, twice the memory size. For increasing amounts of data, the computation time increased slightly faster than linear, but slower than nlogn.
Figure 2 illustrates that the memory consumed by the sorting was 38 MB for the Classic merge sort when the amount of data was 4 MB. On the other hand, the External merge sort consumed less than 2 MB of memory in all cases where the amount of data was 4 MB, 8 MB and 32 MB.
For visualizing the output we used matplotlib and plotted the results of both classic and external merge sort.

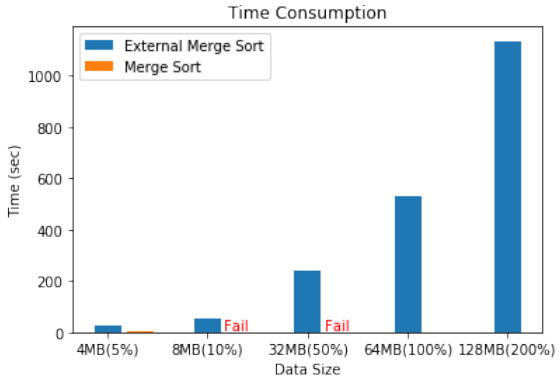


Figure 1: Time Consumption in Sorting with 64MB memory

# 5 Discussion and Conclusion

In this work, we reviewed two sorting algorithms and discussed how they can be compared for better prac-

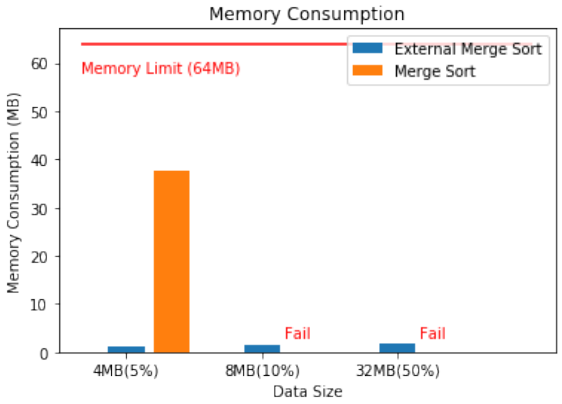


Figure 2: Memory Consumption in Sorting with 64MB memory

tical performance. According to the experiment classic merge sort works better and more efficiently with small data, But it fails to execute when used more than 10 percent of memory. External merge sort works even when data size is huge.