Comparison Time Execution and Memory Usage of Dual-Pivot Quick Sort and Parallel Merge Sort

Kevin Chayadi, Edward Lauwis, Vicky Frandy Lius, Kristien Margi Suryaningrum and Hanis Amalia Saputri

July 16, 2023

Comparison Time Execution and Memory Usage of Dual-Pivot Quick Sort and Parallel Merge Sort

Kevin Chayadi
Computer Science
Bina Nusantara University
Jakarta, Indonesia
kevin.chayadi@binus.ac.id

Edward Lauwis
Computer Science
Bina Nusantara University
Jakarta, Indonesia
edward.lauwis@binus.ac.id

Vicky Frandy LiuSs
Computer Science
Bina Nusantara University
Jakarta, Indonesia
vicky.lius@binus.ac.id

Kristien Margi Suryaningrum
Computer Science
Bina Nusantara University
Jakarta, Indonesia
kristien.s@binus.edu

Hanis Amalia Saputri
Computer Science
Bina Nusantara University
Jakarta, Indonesia
hanis.saputri@binus.ac.id

***Abstracts*- Various algorithms have been used to sort data. now it has very algorithms that have been optimized to be more efficient. We used the algorithms from quicksort and mergesort but modified versions, namely dual-pivot quicksort and parallel mergesort. In this research, we use a case study method to compare the performance of dual-pivot quicksort and parallel mergesort algorithms. We use Java programming language to implement both algorithms. Using a dataset of 10000 book ID data with the integer data type, we will compare the two algorithms in terms of execution time and memory usage. The comparison results show that dual-pivot quicksort is faster than parallel mergesort and also uses less memory than parallel mergesort. This shows that the algorithm of the advanced version of quicksort, namely dual-pivot quicksort, is better than the advanced version of mergesort, namely parallel mergesort. We also identified that dual-pivot on quicksort has significantly more impact than parallel mergesort. It can happen because parallel mergesort only speeds up the queue on recursive which while it helps but is not very significant. This research helps in choosing the best algorithm for sorting in terms of execution time and memory used so that in the future each algorithm is used according to its capacity so that each algorithm can work as efficiently as possible.**

Keywords—Dual-pivot Quicksort, Parallel Mergesort, Execution Time, Memory Usage.

I. Introduction

Sorting is one of the most essential things in computer science. Sorting helps us organize data so it's accessible when we want to find data or update data. currently, there are two most popular algorithms, they are quick sort and merge sort. but the focus of the algorithm is not the standard quick sort and merge sort but the advanced versions of the two algorithms which are the dual-pivot quick sort and multithreading mergesorts or so-called as parallel merge sort.

Since both algorithms are advanced, they also need resources for sorting. and that becomes a question to answer. Therefore, our question is, which one is faster for sorting? and also which one consumes less memory?

So in this research, we do a comparison of resources consumed by each algorithm. each algorithm will sort the data as much as 10000 data. from those 10000 data, we sort them starting from 2000, and slowly we add up to 10000 data. each sequence we analyze the time required and memory usage of each algorithm.

II. Theoretical Basis

A. Related Works

Several researchers have studied dual-pivot quicksort and parallel mergesort and made contributions to optimizing both algorithms.

like researchers in 2022 [4] are optimizing the quicksort algorithm by using dual parallel partitioning and a multi-swap algorithm to speed up the sorting time. The result is dual-pivot quicksort is much faster than the standard quicksort [5].

Also, research in 2019 [6], used parallel mergesort, and it is proven to work to improve the efficiency of mergesort. They improved the mergesort by implementing multi-thread on merge sort, they not only used 2 threads but 4 threads to improve the mergesort, and its works very efficiently.

based on both research, both algorithms can be advanced so the algorithms can work more efficiently [2]. So, we will analyze the result after sorting so that each algorithm can be used as efficiently as possible.

III. Methodology

This research uses the case study method. We use book list data which amounts to more than 10000 ID books that we get from Kaggle, which will be sorted. here of the flowchart of what we are going to do in this research:

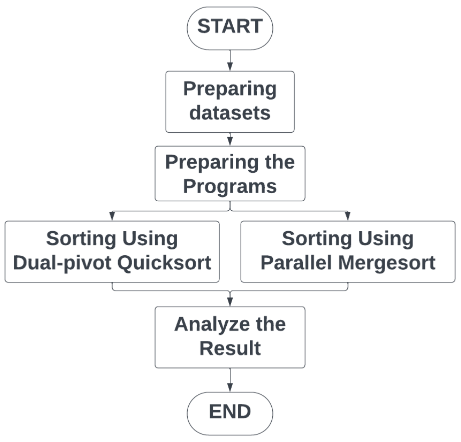
START
Preparing datasets
Preparing the Programs
Sorting Using Dual-pivot Quicksort
Sorting Using Parallel Mergesort
Analyze the Result
END

Figure 3.1 Flowchart research

in figure 3.1, first we will prepare our dataset which is 10000 ID books from Kaggle. after that, we prepare the programs for sorting. when it's sorting, we enter the input in increments. as originally 2000, then we record the results. each sorting of the data stage will be repeated 100 times, we do this to see the algorithm performance for each input. after that, we are going to analyze the result.

1. Dual-Pivot Quicksort

Dual-pivot quicksort works in similar ways to standard quicksort, but dual-pivot uses two pivots instead of one [4]. dual-pivot works by dividing the arrays into 3 subarrays to be sorted, then every subarray becomes a new array that will divide into 3 subarrays to be sorted. this process is repeated recursively until one element is left because if one element is left, it can be considered as already sorted.

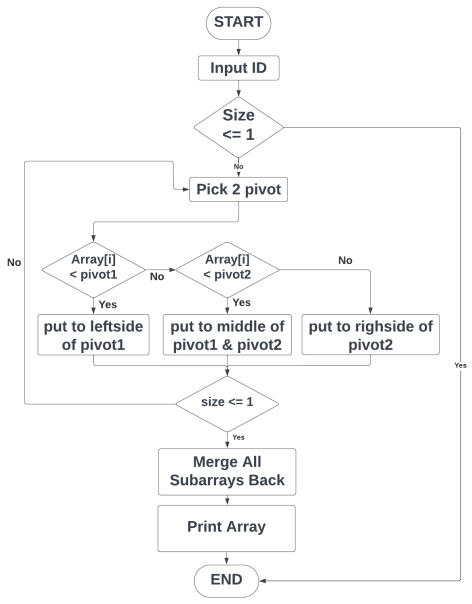

Figure 3.1.1 Flowchart dual-pivot quicksort

Figure 3.1.1 shows how the dual-pivot algorithms work, it divides the arrays into 3 subarrays, and it is done recursively until one element is left. after one element is left, all the subarrays are merged back together. after that, it prints out the result after sorting.

2. Parallel Mergesort

Parallel Mergesort works just the same as standard mergesort, just when dividing the array and when comparing each element [5]. in standard mergesort, dividing will be done after the program is done with another subarray, so the other subarrays are waiting until it's done, same thing when comparing each other, some elements are awaiting to be compared [12]. but when in parallel mergesort, two tasks can be done in a time, and that's affected by the process of sorting.

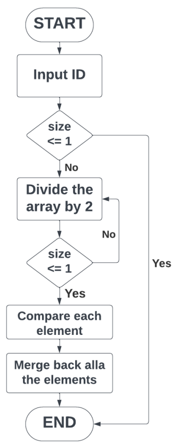

Figure 3.2.1 Flowchart parallel mergesort.

Figure 3.2.1 shows how the parallel mergesort works. flowchart parallel mergesort can be said the same as a standard mergesort flowchart because the only difference is in the divide and compare each element part. parallel mergesort divides two arrays at the same time, also 2 compares at the same time.

IV. Result

We did a comparison of each algorithm based on the time needed for sorting and memory used by using our dataset which is 10000 book IDs. this is the result:

Table 4.1 Result comparison table.

input	Dual-Pivot Quicksort		Parallel Mergesort	
	Time (in nanoseconds)	Memory Usage (in kilobytes)	Time (in nanoseconds)	Memory Usage (in kilobytes)
2000	86088	1048.4	189032	1130.16
4000	194193	2097.128	383519	3227.344
6000	303544	3145.6	525608	5338.552
8000	407302	4194.168	631496	7344.064
10000	512850	5242.720	829925	8470.192

table 4.1 shows the comparison of both algorithms are based on the time and

memory used. The number in the table that is in the time column, represents the time needed for sorting that has already been repeated 100 times for each input to show its performance each input.

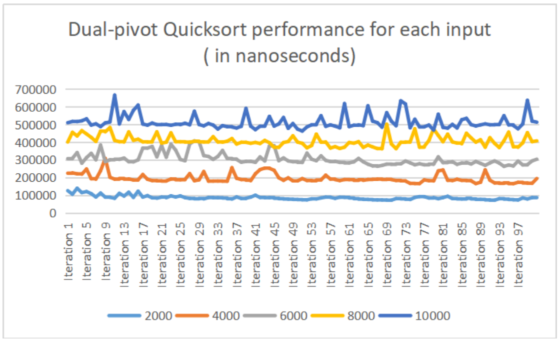

Figure 4.2 Dual-pivot performance for each input.

As it’s shown in figure 4.2, the dual pivot quicksort works pretty efficiently, but it can be seen that the larger the input, the performance of the algorithm is not very efficient. figure 4.2 also shows dual pivot quicksort is not too efficient when the input size is too big. We do not make a diagram for the memory usage for each input because it will produce the output, because of the same input.

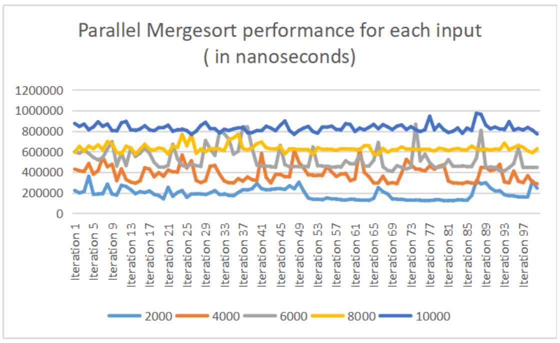

Figure 4.3 Parallel Mergesort performance for each input

meanwhile figure 4.3 shows the performance of parallel mergesort. figure 4.3 shows us that the bigger the input, the more stable the algorithms were. figure 4.2 and figure 4.3 also show the general difference between the two algorithms, namely merge sort is strong for large amounts of data, while quicksort is strong for small amounts of data, but still, both can sort data very quickly. again we do not make a diagram for the memory usage for each input because it will produce the same output.

As it's shown in the table, both algorithms work efficiently, it is proved by 10000 data sorted in less than a second. it is also shown that mergesort uses memory about twice that of quicksort. It can also be seen in the table that the time of quicksort is a little bit faster than mergesort.

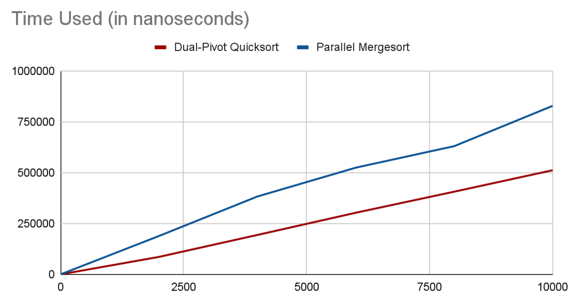

Figure 4.4 Comparison time used (the x-axis represents an input size, and the y-axis represents the time).

Figure 4.4, was taken from table 4.1. it shows that dual-pivot is faster than parallel mergesort. it can happen because using two pivots is affected significantly the way it is sorted instead of using one pivot [3]. but, also when too much data is to be sorted, there are too many partitions to be done, which can slow the algorithms down [3].

whereas parallel mergesort, the mechanism does not change when using multi-threading, it's just that the queue while waiting to be sorted is cut faster, this is also the reason why parallel mergesort is still not very good for dealing with small inputs even it is using multi-threading. even so, the two algorithms are still efficient if used according to the size of the input [13].

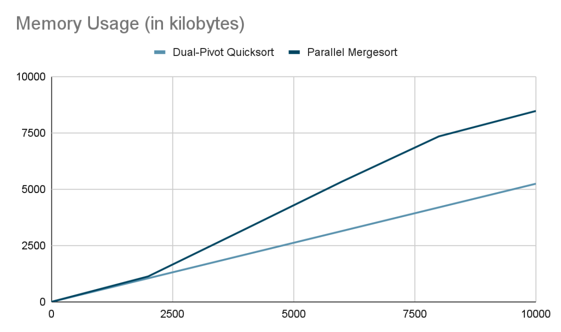

Figure 4.5 Comparison of memory usage (the x-axis represents an input size, and the y-axis represents the memory).

Figure 4.5 taken from the table 4.1 shows the memory used. figure 4.5 also shows the memory used for both algorithms. it shows that dual-pivot quicksort takes less memory while parallel mergesort takes a lot of memory because even in advanced mergesort, it still needs additional memory to store up the sorted array [5][9]. that is why parallel mergesort takes more memory than dual-pivot quicksort.

V. Conclusion

We conduct this research to see the difference based on time used and memory used for both algorithms.

By the experiment that we have done, we conclude that dual-pivot quicksort is faster than the parallel mergsort and takes less memory usage to use. dual-pivot quicksort takes less time can happen because using two pivots is affected significantly the way it is sorted instead of using one pivot [3]. on the other hand, using parallel mergesort can indeed speed up the algorithm but not as significant as dual pivot on quicksort [12].

So both algorithms are very efficient for sorting, but still, there are advantages and disadvantages for both algorithms. each algorithm has its own good conditions to be more efficient so it is important to choose what algorithms based on its conditions.

References

1. A. Gautam, and A. Naman, (2021). Divide and Conquer Sorting Techniques (Vol. 8). IRJET (International Research Journal of Engineering and Technology).
2. Aljabri, N., Al-Hashimi, M., Saleh, M., & Abulnaja, O. (2019). Investigating power efficiency of mergesort. The Journal of Supercomputing, 75, 6277-6302.
3. Hossain, M. S., Mondal, S., Ali, R. S., & Hasan, M. (2020, July). Optimizing complexity of quick sort. In Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020, Revised Selected Papers (pp. 329-339). Singapore: Springer Singapore.
4. Ketchaya, S., & Rattanatranurak, A. (2022). Analysis and optimization of Dual Parallel Partition Sorting with OpenMP. Applied Computing and Informatics, (ahead-of-print).
5. Altarawneh, M., Inan, U., & Elshqeirat, B. (2022). Empirical Analysis Measuring the Performance of Multi-threading in Parallel Merge Sort. International Journal of Advanced Computer Science and Applications, 13(1).
6. Huang, X., Liu, Z., & Li, J. (2019). Array sort: an adaptive sorting algorithm on multi‐thread. The Journal of Engineering, 2019(5), 3455-3459.
7. Klaib, M. F., Sara, M. R. A., & Hasan, M. (2020). A Parallel Implementation of Dual-Pivot Quick Sort for Computers with Small Number of Processors. Indonesia Journal on Computing (Indo-JC), 5(2), 81-90.
8. Buss, S., & Knop, A. (2019). Strategies for stable merge sorting. In Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (pp. 1272-1290). Society for Industrial and Applied Mathematics.
9. Taiwo, O. E., Christianah, A. O., Oluwatobi, A. N., & Aderonke, K. A. (2020). Comparative study of two divide and conquer sorting algorithms: quicksort

and mergesort. Procedia Computer Science, 171, 2532-2540.
10. Neininger, R., & Straub, J. (2018). Probabilistic Analysis of the Dual-Pivot Quicksort “Count”. In 2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO) (pp. 1-7). Society for Industrial and Applied Mathematics.
11. Hossain, M. S., Mondal, S., Ali, R. S., & Hasan, M. (2020, July). Optimizing complexity of quick sort. In Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020, Revised Selected Papers (pp. 329-339). Singapore: Springer Singapore.
12. Marszałek, Z., Woźniak, M., & Połap, D. (2018). Fully flexible parallel merge sort for multicore architectures. Complexity, 2018, 1-19.
13. Blelloch, G. E., Fineman, J. T., Gu, Y., & Sun, Y. (2020, July). Optimal parallel algorithms in the binary-forking model. In Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures (pp. 89-102).
14. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.
15. Marszałek, Z. (2018). Performance tests on merge sort and recursive merge sort for big data processing. Technical Sciences/University of Warmia and Mazury in Olsztyn, (21 (1), 19-35.