COMP - 6521
Advanced Database Techniques and Application

Project Report
On

Two Phase Multiway Merge Sort (TPMWMS)

Professor
Dr. Nematolllaah Shiri

Team Members

Student Name	**Student ID**
Ruchit Dobariya	40232238
Akshit Rameshkumar Desai	40233041
Parth Sonani	40221824

Contents

1) Steps to run the program

- Place the data sets in the ‘inputfiles’ folder and set up the Eclipse environment.
- Configure the main memory size.
- Run the program using the ‘tpmwms.java’ file.
- The program will read the input files based on the configured memory size, sort them into sublists, merge those sublists into the final sorted relation, and perform the bag difference operation on the sorted relations.
- Check the ‘blocks’ folder to view the sublists generated by the program.
- The program will produce the final sorted list with no duplicates, which will be saved in the 'outputfiles' folder.

2) Program Description:

- The first phase of the program involves clearing the 'output' and 'sublist' folders. The program reads the 'T1(r1_large.txt)' and 'T2(r2_large.txt)' datasets one by one and sorts them. The sublists created during phase 1 can be viewed in the 'blocks' folder.
- In the second phase, the sublists are read iteratively into the two input buffers and compared against each other to identify the smallest one. Duplicate tuples are removed during the comparison process. The smallest sublist is then written to the output buffer. This process continues until there are no more sublists left, and the final output is written back to the disk.

3) Architecture Diagram:

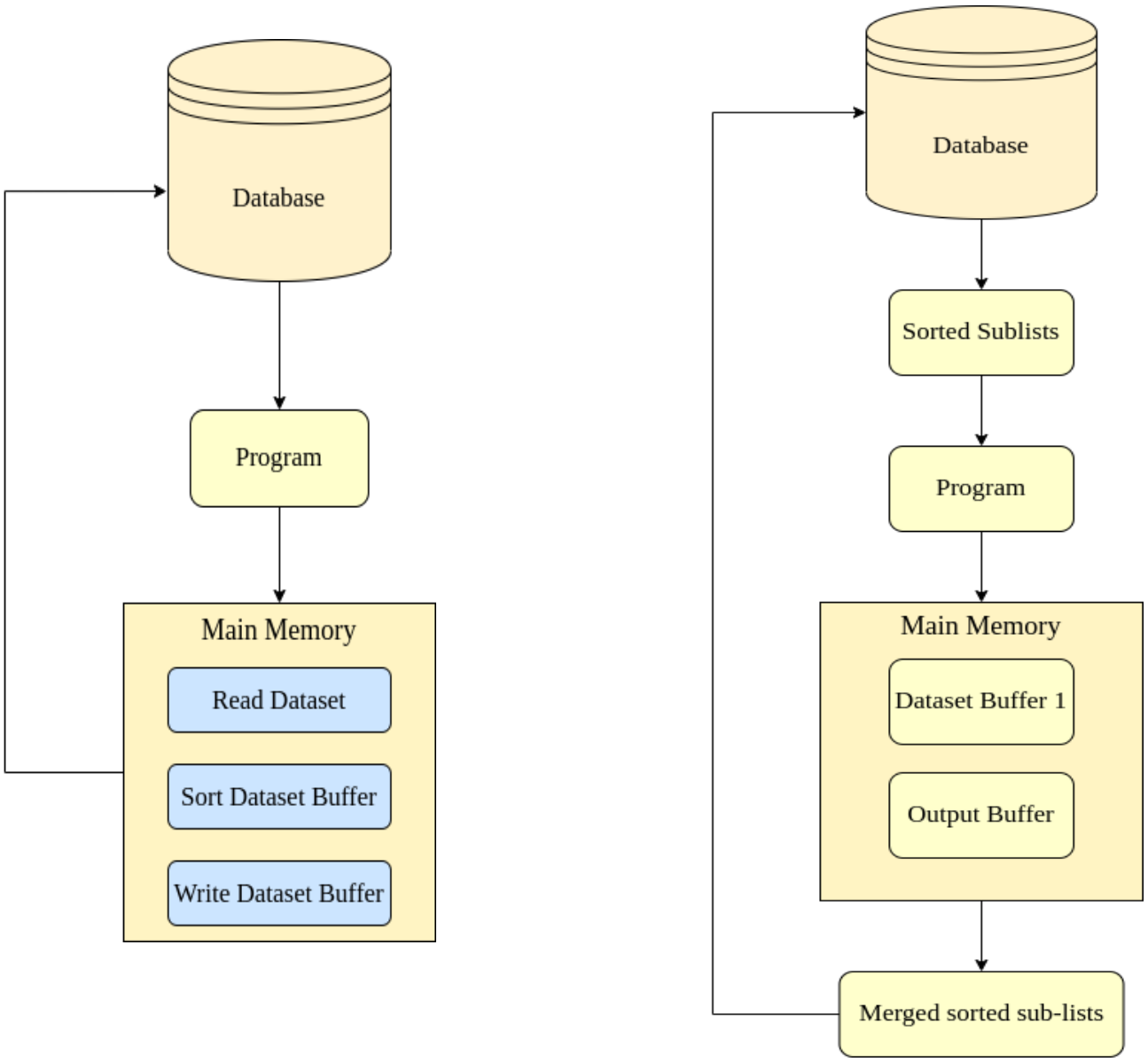


Fig 1 Phase-1 Sort & Phase-2 Merge

4) Algorithm:

Phase-1 (TPMWMS Sort Phase):

1. Start
2. Assign main memory to read sublists.
3. Reset output folder and sublist folder.
4. Load the input buffer (of M blocks) into the block of size 4K bytes.
5. Update the count of I/O operations
6. Apply Quick sort to sort the tuples within the input buffer.
7. Update the count of I/O operations.
8. Update the count of sublists created.
9. Repeat the above steps from 4 to 8 for a calculated number of memory fills.
10. Calculate the time takes to sort the relation
11. End

Phase-2 (TPMWMS Merge Phase):

1. Start
2. Read the input blocks into 2 string variables and check for their validity
3. Compare the whole tuple as per the record offset to maintain the count of each tuple
4. Write the record as well as the count into the output buffer
5. Update the disk I/O count and time taken for one merge iteration
6. Continue the operation until both the buffers are empty
7. End

5) Technical Details:

This program has two distinct phases:

- In Phase-1, the program reads data from an input file and sorts it into smaller sublists.
- In Phase-2, the program combines the sublists, merge duplicate entries, and maintain the total count of each tuple. The final result is then written to an output file.

The above process is followed for M = 51 blocks or 101 blocks, then the execution period is measured for executing the entire operation that involves the sorting and the duplicate count in the merging.

6) Program Results

```
====================Deleting Old Files From Directory====================
Successfully deleted and created Directory For storing blocks.
Successfully deleted and created Directory For Output.
--------------------------------------------------------------------------------
==========================TPMMS-Mini-Project-1==========================
Max Memory Size(Including memory consumed by JVM Processes) = 8.0 Megabyte
Tuple Size (In Bytes) = 100
--------------------------------------------------------------------------------
==============================T1 - Phase-1==============================
Time taken by Phase 1 for T1 : 138ms (0.138sec)
Number of records in T1 = 100000
Number of blocks for T1 = 2500
==============================T2 - Phase-1==============================
Time taken by Phase 1 for T2 : 139ms (0.139sec)
Number of records in T2 = 100000
Number of blocks for T2 = 2500
============================Output-Phase-1============================
Total time taken by Phase 1 (T1+T2) = 277ms (0.277sec)
Total number of records = 200000
Total number of Blocks = 5000
Sorted Disk IO = 10000
--------------------------------------------------------------------------------
================================Phase-2================================
Phase 2 merging time for iteration  0 : 202ms(~approx 0.202sec)
Phase 2 merging time for iteration  1 : 31ms(~approx 0.031sec)
Phase 2 merging time for iteration  2 : 13ms(~approx 0.013sec)
Phase 2 merging time for iteration  3 : 7ms(~approx 0.007sec)
Phase 2 merging time for iteration  4 : 4ms(~approx 0.004sec)
Phase 2 merging time for iteration  5 : 3ms(~approx 0.003sec)
Phase 2 merging time for iteration  6 : 1ms(~approx 0.001sec)

Phase 2 Total time = 261ms (0.261 sec)
Phase 2 Prepare Output Time = 36ms (0.036 sec)
Phase 2 Write Output Time = 225ms (0.225 sec)
Merge Phase - Number of I/O = 9949
Phase 2 Number of blocks written  = 2520
Phase 2 number of output records (including all iteration) = 100927
============================Output-Phase-2============================
Total time Phase 1 & Phase 2 = 538ms
Total time Phase 1 & Phase 2 = 0.538 sec
Total Number of I/O = 19949
Total Number of records in final output = 1000
Sum of frequency of all records in final output = 200000
Which should be same as Number of given input = 200000
--------------------------------------------------------------------------------
```

Result for 51 Blocks

```
======================Deleting Old Files From Directory====================
Successfully deleted and created Directory For storing blocks.
Successfully deleted and created Directory For Output.
-----------------------------------------------------------------------------
============================TPMMS-Mini-Project-1=============================
Max Memory Size(Including memory consumed by JVM Processes) = 8.0 Megabyte
Tuple Size (In Bytes) = 100
-----------------------------------------------------------------------------
================================T1 - Phase-1=================================
Time taken by Phase 1 for T1 : 147ms (0.147sec)
Number of records in T1 = 100000
Number of blocks for T1 = 2500
================================T2 - Phase-1=================================
Time taken by Phase 1 for T2 : 149ms (0.149sec)
Number of records in T2 = 100000
Number of blocks for T2 = 2500
==============================Output-Phase-1=================================
Total time taken by Phase 1 (T1+T2) = 296ms (0.296sec)
Total number of records = 200000
Total number of Blocks = 5000
Sorted Disk IO = 10000
-----------------------------------------------------------------------------
================================Phase-2======================================
Phase 2 merging time for iteration  0 : 148ms(~approx 0.148sec)
Phase 2 merging time for iteration  1 : 15ms(~approx 0.015sec)
Phase 2 merging time for iteration  2 : 7ms(~approx 0.007sec)
Phase 2 merging time for iteration  3 : 4ms(~approx 0.004sec)
Phase 2 merging time for iteration  4 : 3ms(~approx 0.003sec)
Phase 2 merging time for iteration  5 : 1ms(~approx 0.001sec)

Phase 2 Total time = 178ms (0.178 sec)
Phase 2 Prepare Output Time = 82ms (0.082 sec)
Phase 2 Write Output Time = 96ms (0.096 sec)
Merge Phase - Number of I/O = 7501
Phase 2 Number of blocks written  = 1298
Phase 2 number of output records (including all iteration) = 51986
==============================Output-Phase-2=================================
Total time Phase 1 & Phase 2 = 474ms
Total time Phase 1 & Phase 2 = 0.474 sec
Total Number of I/O = 17501
Total Number of records in final output = 1000
Sum of frequency of all records in final output = 200000
Which should be same as Number of given input = 200000
-----------------------------------------------------------------------------
```

Result for 101 Blocks

7) Coding Standards:

The codes were developed by adhering to the most commonly accepted coding conventions.

- The class name begins with an uppercase word.
- E.g.: TPMWMS.java
- Constants are called with characters in the upper case
- The variable name is descriptive and is rendered in lower case including a capital letter to separate words.
- The procedure name begins with a lowercase character and uses the uppercase characters to separate words.

8) Class Description:

- **Constants.java:** This class stores the constant values required for program execution like file IO paths, block size etc.
- **PhaseOne.java:** This class has the methods for reading tuples into main memory, sorting them and creating the sublists. It also handles the IO calculation for phase one.
- **PhaseTwo.java:** This class has the methods for combining the sublists into a final sorted list.
- **TPMPWS.java:** This class has the main method from where program execution starts. It builds and clears the output directories and prints the timing outputs. It also creates the objects for other classes and triggers their execution.
- **QuickSort.java:** This class performs the quicksort algorithm on the blocks read into memory
- **Validator.java:** to validate the output generated by TPMPWS.

9) Analysis

a) Why is it consuming more memory than we specified in the -Xmx argument.

i) Both values of M=51(=514096 bytes) and M=101(=1014096 bytes) will consume very less memory(<1 MB).

ii) So, to run out we can code, we can even use -Xmx2m which will limit max heap memory to 2 MB. And it should more than enough to perform the TPMWMS.

iii) Here, When we try to log memory available at run time using Runtime.getRuntime().maxMemory(), We can see that it will consume more memory than we specified in Xmx argument.

iv) This is because JVM consumes memory for it's processes and objects other than Heap Memory consumed by Java Objects.

b) Stats for M=51 & M=101 blocks. (Answers Q-1,2,3)

T1=r1_large.txt(100000 records) & T2=r2_large.txt(records) Total blocks = (100000*2)/40 = 5000 blocks

For Phase-1:-

Memory Limit	No. of sublists	Time taken	No. of Disk I/O
101 Blocks	50	280ms	10000
51 Blocks	99	277ms	10000

For Phase-2:

Memory Limit	Total iteration	Total time taken	Time taken to prepare output	Time taken to write output	No of Disk I/O	Total output records	No of blocks to be written
101 Blocks	6	167ms	94ms	73ms	7501	51986	1298
51 Blocks	7	261ms	36ms	225ms	9949	100927	2520

Total:-

Memory Limit	Total Time taken	No of Disk I/O	No of records in final output
101 Blocks	447ms	17501	1000
51 Blocks	538ms	19949	1000

c) The scalability issue as the number of records in each of R1 and R2 increases to 500,000 and 1000,000.

i) When records in both R1 and R2 are increased to 500,000 or 1,000,000 then the number of sublists produced by Phase-1 will also increase.

ii) And that will also increase the number of iterations in phase two.

iii) It can cause hard disk failures,
   (1) If disk memory is not sufficient to store intermediate sublists, we need to delete the files of previous iteration after current iteration is complete, so that space can be reused by overwriting.
   (2) This may result in a dirty bit for the hard disk. And a higher number of disk writes can also lead to disk failure.

iv) If we can increase the main memory size to fit more number of blocks to output less number of sublists from phase-1.
   (1) Size of intermediate sublists will increase.
   (2) Less number of iteration will be required.
   (3) Number of disk I/O will reduce.
   (4) It will take less time to produce output.

v) But if we can not increase the main memory than it is difficult to scale it to handle 500,000 or 1000,000 records of R1 and R2.

vi) Stats for 500,000 and 1000,000 records in R1 and R2:- (M=101 blocks)

For Phase-1:-

No. of Records	No. of sublists	Time taken	No. of Disk I/O
500,000	247	935ms	50000
1,000,000	495	1629ms (1.629sec)	100000

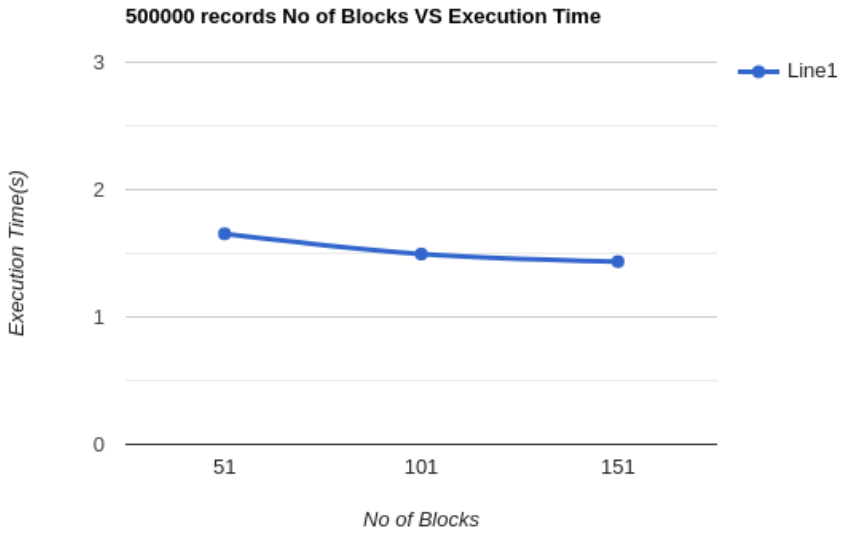
For Phase -2:-

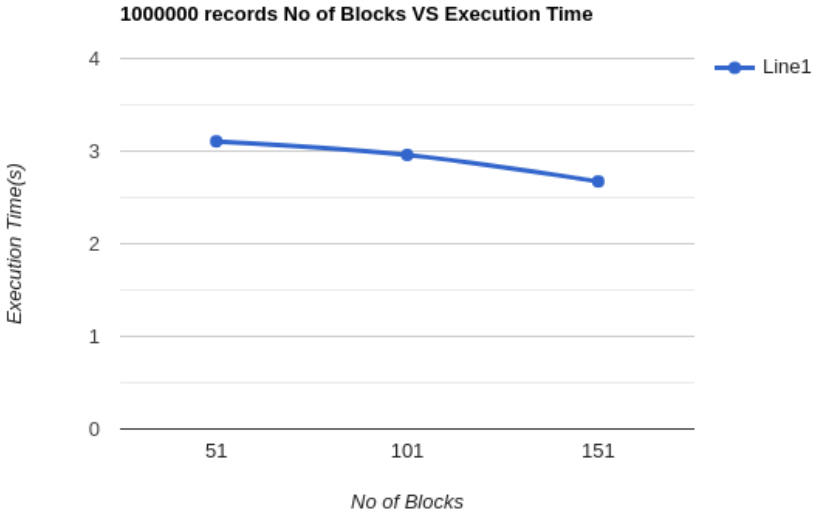
No or records	Total iteration	Total time taken	Time taken to prepare output	Time taken to write output	No of Disk I/O	Total output records	No of blocks to be written
101 Blocks	8	500ms	82ms	463ms	37359	247956	6192
51 Blocks	9	1330 ms	116ms	1214 ms	74772	495912	12385

Total:-

Memory Limit	Total Time taken	No of Disk I/O	No of records in final output
500,000	1435ms (1.435 sec)	87359	1000
1,000,000	2959ms (2.959 sec)	174772	1000

vii) Comparison of No of Blocks with Execution time:





10) Summary

Two phase multiway merge sort performs good if below points are followed.

- Maximize the duration of initial operations.
- Try to perform input, processing, and output simultaneously in all phases of the operation.
- Utilize as much working memory as possible, as increasing memory usage usually results in faster processing. This is especially true for external sorting, where a faster CPU will not provide significant performance improvement, since disk I/O speed is the primary bottleneck.
- Use multiple disk drives to enhance the overlap of processing with input/output operations, and to enable sequential file processing.

11) References

a) https://www.baeldung.com/java-quicksort
b) https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html
c) https://docs.oracle.com/javase/8/docs/api/java/io/BufferedWriter.html
d) https://www.edureka.co/community/5621/default-parameters-of-xms-and-xmx-in-jvm
e) https://coderanch.com/t/654510/java/Xmx-MB-Max-heap-size
f) https://github.com/sagarvetal/ADB_Project_1_TPMMS
g) https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html
h) https://docs.oracle.com/javase/tutorial/essential/io/copy.html
i) https://docs.oracle.com/javase/tutorial/essential/io/delete.html
j) https://docs.oracle.com/javase/tutorial/essential/io/dirs.html
k) https://codereview.stackexchange.com/questions/188716/merge-sorted-lists-removing-duplicates
l) https://www.tutorialspoint.com/how-to-measure-execution-time-for-a-java-method
m) https://www.baeldung.com/java-memory-beyond-heap
n) https://stackoverflow.com/questions/48798024/java-consumes-memory-more-than-xmx-argument
o) https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html
p) https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/ExternalSort.html
q) https://www.geeksforgeeks.org/external-sorting/