NoSQL Assignment 2 – Part A

IMT2021008 – Sheikh Muteeb

IMT2021003 – Keshav Chandak

IMT2021007 – Sunny Kaushik

IMT2021076 – Devendara Rishi Nelapati

Problem 1

1. Yes, the M-Counter is a CRDT.

Justification:
The M-Counter is a CRDT as it satisfies all the four mentioned properties.

- **Merge is associative**
  - Let x, y, z be 3 M-counter objects represented by their internal state as follows:
  - x = [x1, ….. , xn], y = [y1, ….., yn], z = [z1,….. Zn]
  - merge(merge(x, y), z) = merge([max(x1, y1), ..., max(xn, yn)], [z1,, … zn])

    = [max(max(x1, y1), z1), ..., max(max(xn, yn), zn)]

- merge(x, merge(y, z)) = merge([x1, .. xn], [max(y1, z1) ..., max(yn, zn)])

  = [max(x1, max(y1, z1)), ..., max(xn, max(yn, zn))]

- By associativity of max operation, max(max(xi, yi), zi) = max(xi, max(yi, zi)) for all i from 1 to n
- Thus merge(merge(x, y), z) = merge(x, merge(y, z))
- Hence proved
- **Merge is commutative**
  - Let x, y be 2 M-counter objects represented by their internal state as follows:
  - x = [x1, ….. , xn], y = [y1, ….., yn]
  - merge(x, y) = [max(x1, y1), ..., max(xn, yn)]
  - merge(y, x) = [max(y1, x1), ..., max(yn, xn)]
  - By commutativity of max operation, max(xi, yi) = max(yi, xi) for all i from 1 to n
  - Thus merge(x, y) = merge(y, x)
  - Hence proved
- **Merge is idempotent**
  - Let x be an M-counter object represented by its internal state as follows:
  - x = [x1, ….. , xn]
  - merge(x, x) = [max(x1, x1), …., max(xn, xn)]
  - We know max(xi, xi) = xi for all i from 1 to n
  - Thus merge(x, x) = x
  - Hence proved
- **Update is increasing**
  - Let x be an M-counter object on server i represented by its internal state as follows:
  - x = [x1, ….. , xn]
  - Let y = Add(x, c) = [x1, … xi + c, …, xn] where c is a positive integer.
  - merge(x, y) = [max(x1, y1), ., max(xi, yi), .., max(xn, yn)]

= [max(x1, x1), ., max(xi, xi + c), .., max(xn, xn)]
= [x1, … xi + c, …, xn] (Since c is positive, xi + c > xi)
= y

- Thus if y = add(x, ..), then merge(x, y) = y
- Hence proved

2. Let *a* have server id = 0 and *b* have server id = 1

	State	Query	History
a0	i : 0, n : 2, xs : [0, 0]	0	{}
a1	i : 0, n : 2, xs : [1, 0]	1	{0}
a2	i : 0, n : 2, xs : [1, 2]	3	{0, 1}
b0	i : 1, n : 2, xs : [0, 0]	0	{}
b1	i : 1, n : 2, xs : [0, 2]	2	{1}
b2	i : 1, n : 2, xs : [0, 6]	6	{1, 2}
b3	i : 1, n : 2, xs : [1, 6]	7	{0, 1, 2}

3. CRDTs should ideally balance **formal correctness** with **intuitive application semantics**, but there is often a trade-off between the two.

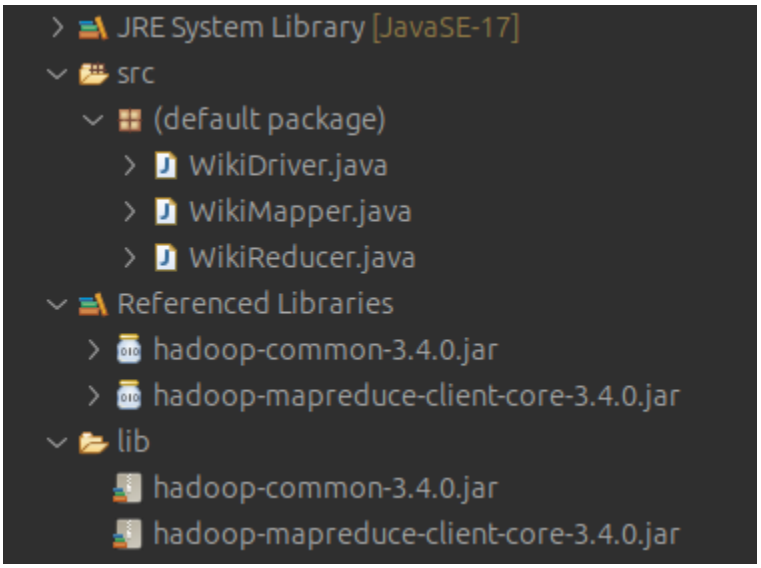
Trade-offs:

1. **Prioritizing Correctness**
    a. **Pros:** Ensures deterministic, reliable behavior in all scenarios.
    b. **Cons:** Can lead to unintuitive outcomes that do not match user expectations.
2. **Prioritizing Application Semantics**
    a. **Pros:** Improves user experience by aligning with expected behavior, meets domain specific expectations.
    b. **Cons:** Might require deviations from strict mathematical correctness, possibly leading to unpredictable behaviour and inconsistencies.

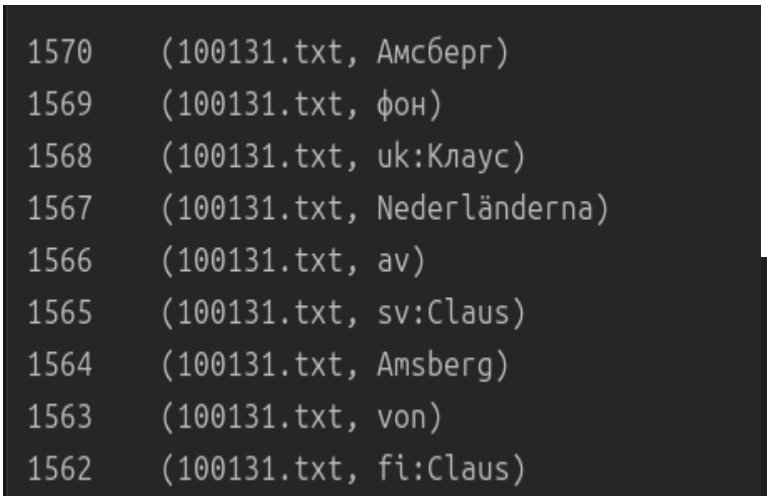
The ideal balance depends on the use case: critical financial transactions prioritize formal correctness, whereas collaborative note-taking apps may need more responsiveness and flexibility and allow small inconsistencies.

Problem 2

File structure:



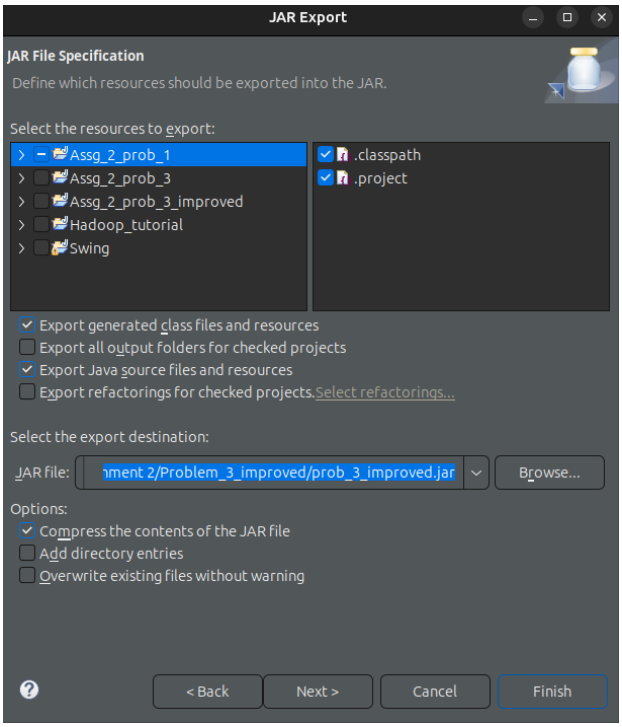
1. Mapper(WikiMapper.java)
    a. Reads a Wikipedia article, **tokenizes words**, and **emits (index, (docID, word))**.
    b. docID is extracted from the filename.
2. Reducer(WkikReducer.java)
    a. Groups words **by index** and **selects the word from the highest docID**.

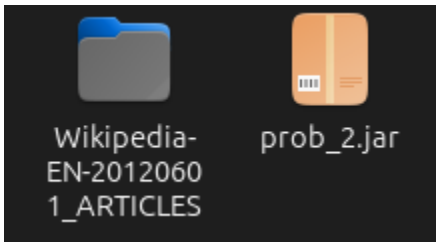


Example output from the MapReduce program and the files generated

Implementation:

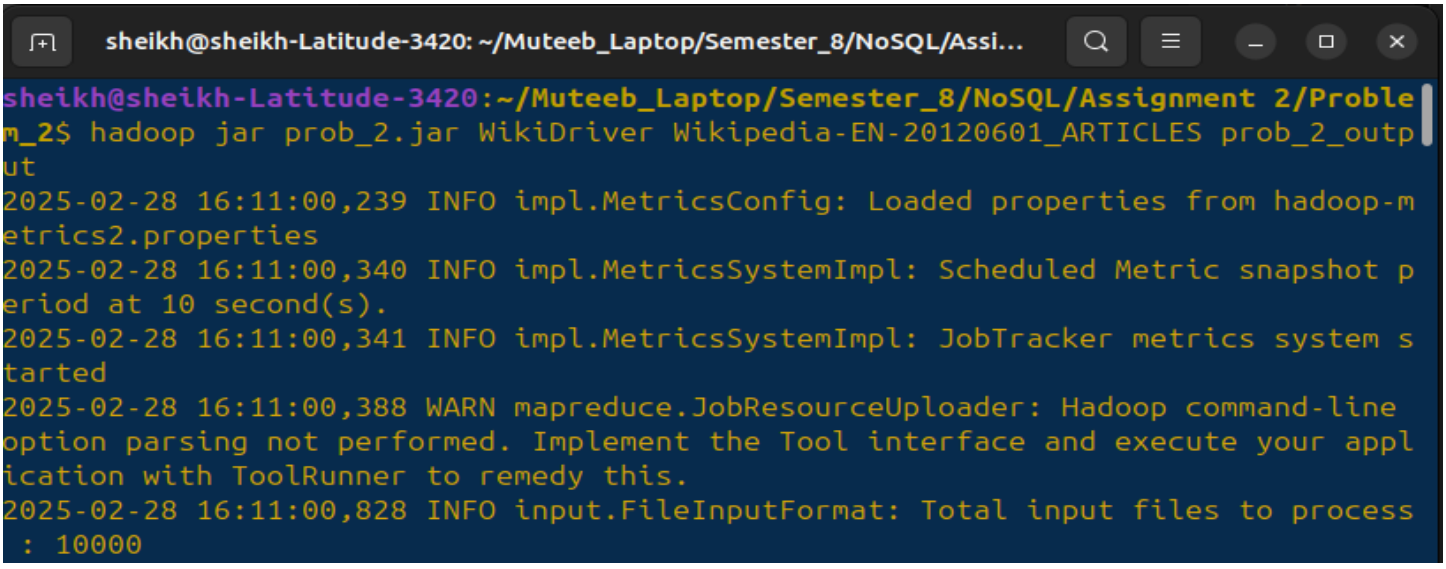
1. Export the whole project as a JAR file to a local directory.





prob_2.jar is the whole project exported to a local directory

2. Then in the terminal run the following command:



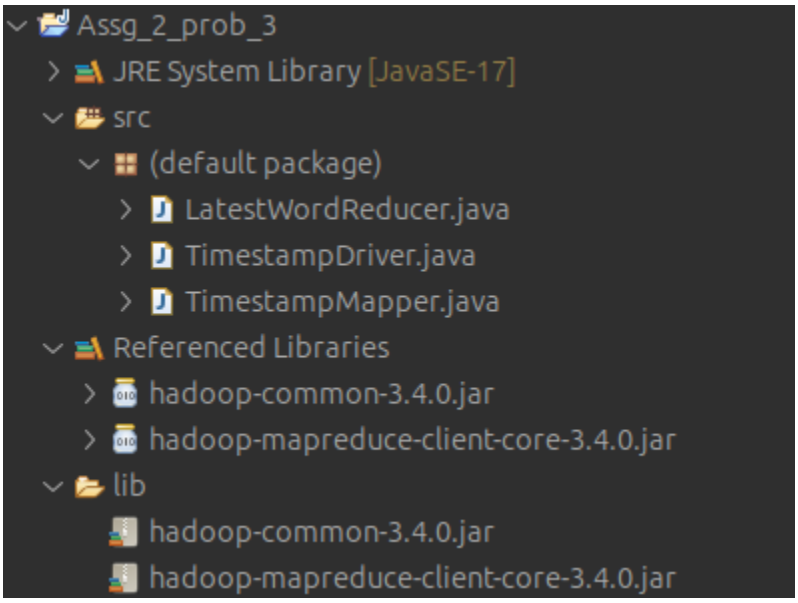
The output gets stored in the prob_2_output folder

prob_2_output folder contents

Directory after running hadoop mapreduce

Problem 3

File Structure:



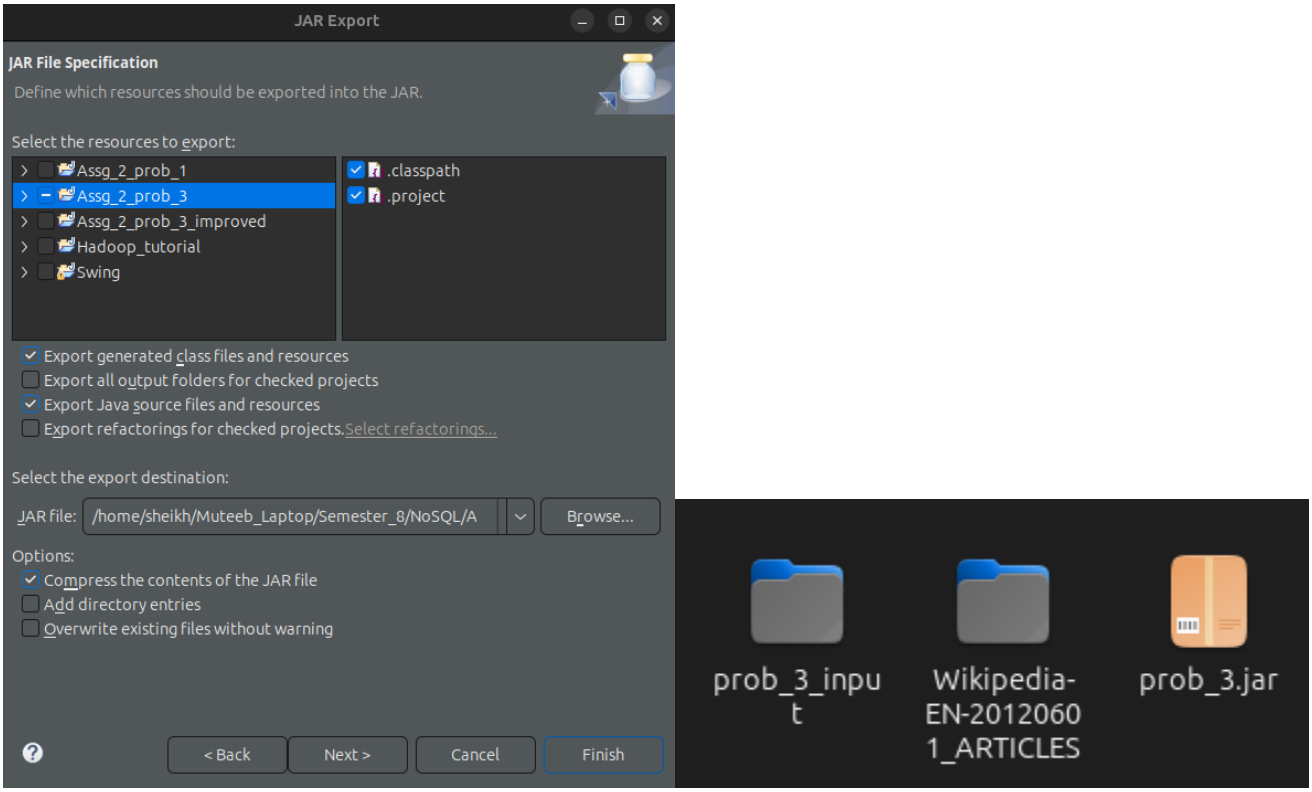
1. Mapper (TimestampMapper.java) :
    a. Reads **Problem 2 output** and emits **(index, (timestamp, word))**.
2. Reducer (LatestWordReducer.java):
    a. Retains only words **from the max timestamp (latest revision).**

```
0       [
3       ]
6       07
9       {
12      {
15      }
18      }
21      Memorial
24      {
27      New
30      {
33      (1981-1985
```

Contents from the recieved output files

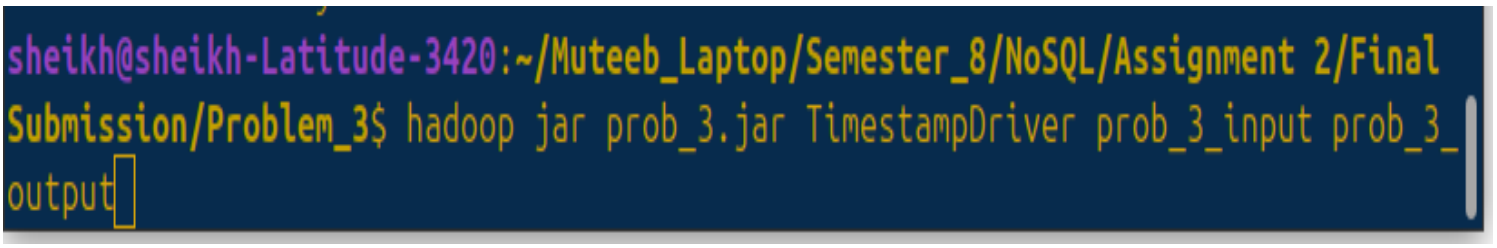
Implementation:

1. Export the whole project as a JAR file to a local directory.



prob_3.jar is the whole project exported to a local directory with all other required files

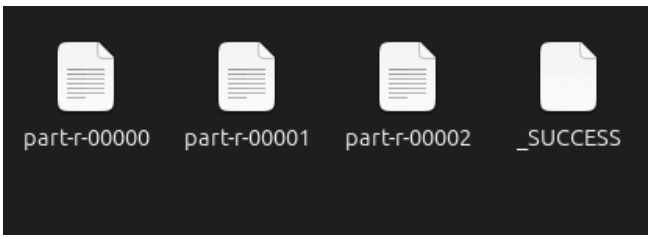
3. Then in the terminal run the following command:



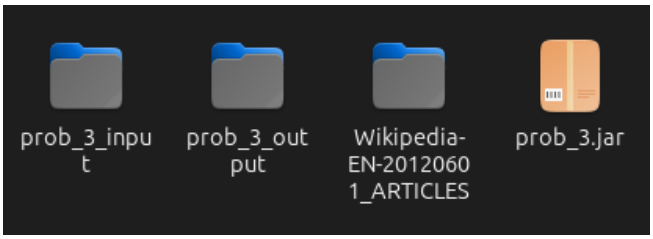
```
2025-02-28 17:47:03,501 INFO mapred.Task: Final Counters for attempt_local107794375_0001_r_000002_0: Counters: 24
        File System Counters
                FILE: Number of bytes read=1616006554
                FILE: Number of bytes written=983448222
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Combine input records=0
                Combine output records=0
                Reduce input groups=9663
                Reduce shuffle bytes=162689084
                Reduce input records=8582815
                Reduce output records=9663
                Spilled Records=8582815
                Shuffled Maps =21
                Failed Shuffles=0
                Merged Map outputs=21
                GC time elapsed (ms)=86
                Total committed heap usage (bytes)=92274688
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Output Format Counters
                Bytes Written=116911
2025-02-28 17:47:03,502 INFO mapred.LocalJobRunner: Finishing task: attempt_local107794375_0001_r_000002_0
2025-02-28 17:47:03,502 INFO mapred.LocalJobRunner: reduce task executor complete.
```

```
                FILE: Number of bytes written=8478614695
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Map input records=25759343
                Map output records=25759343
                Map output bytes=436688713
                Map output materialized bytes=488215344
                Input split bytes=3759
                Combine input records=0
                Combine output records=0
                Reduce input groups=28990
                Reduce shuffle bytes=488215344
                Reduce input records=25759343
                Reduce output records=28990
                Spilled Records=51518686
                Shuffled Maps =63
                Failed Shuffles=0
                Merged Map outputs=63
                GC time elapsed (ms)=337
                Total committed heap usage (bytes)=7795113984
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=633446219
```

Above is the successfull execution after running.



prob_3_output folder contents



Directory after running hadoop mapreduce

4. Running python program to check for differences with the latest wikipedia article:

```
sheikh@sheikh-Latitude-3420:~/Muteeb_Laptop/Semester_8/NoSQL/Assignment 2/
Final Submission/Problem_3$ python3 merge_and_compare.py prob_3_output Wik
ipedia-EN-20120601_ARTICLES
✅ Merged output saved to: merged_output.txt
✅ Latest Wikipedia article: 567579.txt (ID: 567579)

🔍 **Comparison Results:**

❌ Differences found =  27281
```