NoSQL Systems - Assignment 2 Report

Siddharth Kothari (IMT2021019)

Sankalp Kothari (IMT2021028)

Vikas Kalyanapuram (IMT2021040)

M Srinivasan (IMT2021058)

February 27, 2025

Contents

Installation Instructions

1. For installing Hadoop on macOS, refer to the detailed guide available at: Installing Hadoop on macOS
2. For installing Hadoop on Ubuntu, refer to the detailed guide available at: Installing Hadoop on Ubuntu

1 Problem 1

1.1 M-Counter as a CRDT

Yes, M-Counter qualifies as a CRDT as it obeys the following properties:

1. **Associative:**

$$merge(merge(x,y),z) = merge(x, merge(y,z))$$

Proof. Let:

$$x = [x_1, x_2, ..x_n]$$
$$y = [y_1, y_2, ...y_n]$$
$$z = [z_1, z_2, ...z_n]$$

We can see that:

$$merge(x,y) = [max(x_1,y_1), max(x_2,y_2), ..max(x_n,y_n)]$$
$$merge(y,z) = [max(y_1,z_1), max(y_2,z_2), ..max(y_n,z_n)]$$

From here:

$$merge(merge(x,y),z) = [max(max(x_1,y_1),z_1), max(max(x_2,y_2),z_2), ..max(max(x_n,y_n),z_n)]$$
$$= [max(x_1,y_1,z_1), max(x_2,y_2,z_2), ...max(x_n,y_n,z_n)]$$

and

$$merge(x, merge(y,z)) = [max(x_1, max(y_1,z_1)), max(x_2, max(y_2,z_2)), ...max(x_n, max(y_n,z_n))]$$
$$= [max(x_1,y_1,z_1), max(x_2,y_2,z_2), ...max(x_n,y_n,z_n)]$$
$$= merge(merge(x,y),z)$$

Hence, we can see that M-Counter is Associative. □

2. **Commutative:**

$$merge(x,y) = merge(y,x)$$

Proof. Let:

$$x = [x_1, x_2, ..x_n]$$
$$y = [y_1, y_2, ...y_n]$$

We can see that:

$$merge(x,y) = [max(x_1,y_1), max(x_2,y_2), ..max(x_n,y_n)]$$
$$merge(y,x) = [max(y_1,x_1), max(y_2,x_2), ..max(y_n,x_n)]$$
$$= [max(x_1,y_1), max(x_2,y_2), ..max(x_n,y_n)]$$
$$= merge(x,y)$$

Hence, we can conclude that M-Counter is Commutative. □

3. **Idempotent:**

$$merge(x, x) = x$$

Proof. Let:

$$x = [x_1, x_2, ..x_n]$$

Now,

$$merge(x, x) = [max(x_1, x_1), max(x_2, x_2), ...max(x_n, x_n)]$$

$$= [x_1, x_2, ..x_n]$$

$$= x$$

Hence, we can conclude that M-Counter is Idempotent. $\square$

4. **Update** is **Increasing**: Let x be an arbitrary state-based object, and let $y = update(x, ...)$ be the result of applying an arbitrary update to x. The update method is increasing if:

$$merge(x, y) = y$$

Proof. Let:

$$x = [x_1, x_2, ..x_n]$$

Now, we obtain y by performing an update on x. So, let:

$$y = x.update(\alpha)$$

   (a) **Case-1:** $\alpha \geq 0$
   So,

$$y = [x_1, x_2, ..., x_i + \alpha, x_{i+1}, ..., x_n]$$

   From here:

$$merge(x, y) = [max(x_1, x_1), max(x_2, x_2), ..., max(x_i, x_i + \alpha), max(x_{i+1}, x_{i+1}), ..., max(x_n, x_n)]$$

$$= [x_1, x_2, ..., x_i + \alpha, x_{i+1}, ..., x_n] \text{ (as } \alpha \geq 0)$$

$$= y$$

   (b) **Case-2:** $\alpha < 0$
   So,

$$y = [x_1, x_2, ..., x_i, x_{i+1}, ..., x_n]$$

   From here:

$$merge(x, y) = [max(x_1, x_1), max(x_2, x_2), ..., max(x_i, x_i), max(x_{i+1}, x_{i+1}), ..., max(x_n, x_n)]$$

$$= [x_1, x_2, ..., x_i, x_{i+1}, ..., x_n]$$

$$= y$$

Hence, the update function is increasing. $\square$

1.2 State Table and Diagram

For this question, since we are not given the values of the server id for the servers a and b, and neither are we given the total number of servers, for making the state table, we assume only **2 servers**, with the server a being assigned server number 1, and server b is assigned number 2.
The state diagram is depicted in Figure 1, and the corresponding state table (with snapshots, state, queries and causal history) is provided in Table 1.

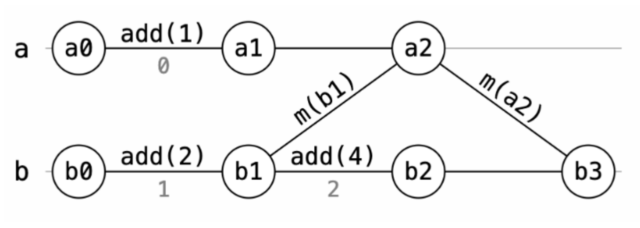


Figure 1: State Diagram

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

Table 1: State Table

Initially, the snapshots a0 and b0 start with both the array values 0, the query will return 0 as the sum, and both have an empty causal history.
Then the update with id 0 is applied on server a, taking it to snapshot a1, which has xs=[1,0] (as a is server 0). Similarly, server b is taken to snapshot b1, which has xs=[0,2] (as b is server 1), after applying the update with id 1. The causal history is updated to reflect the update done. The query on a1 will now reflect 1 as the output, and query on b1 will reflect 2 as the output.
We now merge a1 and b1 to obtain a2. This will result in a index-wise max of the arrays a.xs = [1,0] and b.xs = [0,2], to obtain a.xs = [1,2]. The causal history will now be updated to {0,1}, and the query will now reflect 3 as the output.
b2 is obtained by updating server b with the update with id 2 (this has xs=[0,6]). The causal history now changes to {1,2}, and the query changes to reflect the output as 6.
Finally, b3 will be obtained by merging a and b, to obtain b.xs=[1,6]. The query will now output 7, and the causal history is updated to reflect {1,2,0}.

1.3 Trade-Offs between Correctness and Application Semantics

Conflict-Free Replicated Data Types (CRDTs) are used in distributed systems to handle concurrent updates without requiring coordination mechanisms like locks. They must balance two critical aspects:

- **Formal Correctness (Convergence & Consistency):** Ensures all replicas eventually reach the same state regardless of update order by leveraging mathematical properties like commutativity, associativity, and idempotency.
- **Application Semantics (Intuitive Behavior):** Ensures that merged data aligns with user expectations, preventing results that, while formally correct, may be unintuitive or unusable.

Correctness must be explicitly enforced because:

- **Concurrency Complications:** Distributed systems involve multiple concurrent updates, leading to potential conflicts.
- **Eventual Consistency vs. Strong Consistency:** Unlike traditional databases, CRDTs operate under relaxed consistency models, necessitating correctness guarantees.

- **Data Integrity & Trust:** Applications such as financial transactions or medical records require correctness to avoid data corruption.
- **Avoiding Manual Conflict Resolution:** Without correctness, developers may need to manually resolve inconsistencies, increasing complexity.

Trade-offs Between Formal Correctness and Application Semantics

Aspect	Formal Correctness	Application Semantics
Definition	Ensures all replicas reach the same state	Ensures intuitive user behavior
Advantages	Predictable behavior, avoids divergence	Meaningful results, better user experience
Disadvantages	May not align with user expectations	May introduce inconsistencies
Example Where Critical	Shopping carts, financial transactions	Collaborative editing, task management
When to Prioritize	When correctness errors cause corruption	When user experience is key

Table 2: Trade-offs Between Formal Correctness and Application Semantics in CRDTs

Execution Instructions for Problems 2 and 3

Problem 2

```
hdfs dfs -mkdir -p /user/hadoop/input
hdfs dfs -put extracted_articles/*.txt /user/hadoop/input
javac -cp $(hadoop classpath) -d output/ WikipediaMapper.java WikipediaReducer.java
    WikipediaJob.java
jar -cvf WikipediaJob.jar -C output/ .
hadoop jar WikipediaJob.jar WikipediaJob /user/hadoop/input /user/hadoop/output
hdfs dfs -cat /user/hadoop/output/part-*
```

Now, note that we are reading from the output directory of Problem 2 for Problem 3.

Problem 3 Part 1

```
javac -cp $(hadoop classpath) -d classes/ TimestampMapper.java
    MaxTimestampReducer.java WikipediaTimestampDriver.java
jar -cvf WikipediaTimestampJob.jar -C classes/ .
hadoop jar WikipediaTimestampJob.jar WikipediaTimestampDriver /user/hadoop/output
    /user/hadoop/output1
hdfs dfs -cat /user/hadoop/output1/part-*
```

Problem 3 Part 2

```
javac -cp $(hadoop classpath) -d classes_new/ LatestTimestampMapper.java
    LatestTimestampReducer.java LatestDocumentDriver.java
jar -cvf LatestDocumentDriver.jar -C classes_new/ .
```

```
hadoop jar LatestDocumentDriver.jar LatestDocumentDriver /user/hadoop/output
    /user/hadoop/output2
hdfs dfs -cat /user/hadoop/output2/part-*
```

2 Problem 2

2.1 Wikipedia Mapper

The `WikipediaMapper` class extends the Hadoop `Mapper` class and processes text files to generate key-value pairs, where:

- The key is the document ID (extracted from the filename).
- The value is a combination of the word's position (index) and the word itself.
- **map Method -**
  1. Extracts the document ID from the filename by removing the `.txt` extension.
  2. Tokenizes the input text line by splitting it into words.
  3. Iterates through each word, removing non-alphabetic and non-numeric characters and converting it to lowercase.
  4. If the word is not empty, it pairs the index with the word and writes the result as (`docID`, `index,word`).

2.2 Wikipedia Reducer

The `WikipediaReducer` class extends the Hadoop `Reducer` class and processes intermediate key-value pairs generated by the mapper. It groups words by document ID, sorts them by their position in the document, and outputs key-value pairs where:

- The key is the index (word position in the document).
- The value is a formatted pair containing the document ID and the word.
- **reduce Method**:
  1. Collects all words and their positions from the given `values` iterable.
  2. Sorts the words based on their index (position in the document).
  3. Iterates over the sorted words, extracts the index and word, and formats the value as `(doc-ID, word)`.
  4. Writes the output as (`index`, `(doc-ID, word)`).

2.3 Wikipedia Job

The `WikipediaJob` class configures and runs a the mapper and reducer. It specifies the mapper and reducer classes, input and output formats, and other job parameters.

- **Main Method:**
  - Validates that exactly two command-line arguments (input and output paths) are provided.
  - Initializes a Hadoop `Configuration` object and creates a new `Job` instance with the name Wikipedia Collaborative Editing.
- **Job Configuration:**
  - Sets the main job class using `setJarByClass(WikipediaJob.class)`.

  - Assigns `WikipediaMapper` as the mapper and `WikipediaReducer` as the reducer.
  - Defines the output key-value types:
    * Mapper Output: `Text` (document ID) as the key and `Text` (index-word pair) as the value.
    * Reducer Output: `IntWritable` (index) as the key and `Text` (document ID, word pair) as the value.
  - Sets the number of reducer tasks to at least 3 using `setNumReduceTasks(3)`.

- **Input and Output Paths:**
  - Reads the input path from the first command-line argument and assigns it using FileInputFormat.addInputPath().
  - Reads the output path from the second argument and assigns it using FileOutputFormat.setOutputPath().

- **Job Execution:**
  - Submits the job using `job.waitForCompletion(true)`.
  - Terminates the program with an exit code of 0 (success) or 1 (failure).

3 Problem 3

The outputs of the reducer of Problem 2 are passed as inputs to both the mappers in Problem 3.

3.1 Timestamp Mapper

The `TimestampMapper` class is a custom Mapper implementation in Hadoop MapReduce. It processes input text lines and extracts a timestamp from a document ID while associating it with a corresponding word.

3.1.1 Input Format

The Mapper receives:

- **Key:** A line number (`LongWritable`).
- **Value:** A text line (`Text`), containing three parts separated by whitespace: an index, a document ID (docID), and a word.

3.1.2 Processing Steps

The Mapper processes each line as follows:

1. Trim the input line and split it using whitespace.
2. If the number of parts is less than three, log an error and skip the line.
3. Extract the **index** (first element) and convert it into an integer.
4. Extract the **docID** (second element), removing non-numeric characters.
5. Convert the docID into a timestamp:
   - Interpret the docID as the number of days since the Unix epoch (January 1, 1970).
   - Convert days to milliseconds (`1 day = 86400000` milliseconds).
   - Create a Java `Date` object from the timestamp.
6. Extract the **word** (third element) while removing any non-alphabetic characters.
7. Emit a key-value pair where:
   - Key: `index` (`IntWritable`).
   - Value: A concatenation of the `timestamp` and `word`, separated by a comma (`Text`).

3.1.3 Error Handling

- If the input does not contain at least three parts, the Mapper logs an error and skips the line.
- If the docID cannot be converted to a valid number, a warning message is printed, and a default timestamp of `-1` is assigned.

3.2 Max Timestamp Reducer

The `MaxTimestampReducer` processes key-value pairs emitted by the Mapper to identify the word associated with the latest (maximum) timestamp for each key.

3.2.1 Input Format

The Reducer receives:

- **Key:** An integer (`IntWritable`).
- **Values:** A list of comma-separated strings (`Text`), where each value follows the format `"timestamp,word"`.

3.2.2 Processing Steps

The Reducer follows these steps:

1. Initialize `latestTimestamp` to the smallest possible value (`Long.MIN_VALUE`) and `latestWord` as an empty string.
2. Iterate over the list of values:
   - Split each value using `","` to extract the timestamp and word.
   - If the format is incorrect (i.e., missing timestamp or word), skip the entry.
   - Convert the timestamp string into a `long` value.
   - Extract the word while removing any non-alphabetical characters.
   - If the extracted timestamp is greater than `latestTimestamp`, update both `latestTimestamp` and `latestWord`.
3. After processing all values for a key, emit the key along with the word corresponding to the maximum timestamp.

3.2.3 Error Handling

If a value cannot be parsed into a valid number, the Reducer logs an error and skips the entry, ensuring robustness.

3.3 Timestamp Driver

The `WikipediaTimestampDriver` class configures and runs the mapper and reducer mentioned above.

- **Main Method:**
  - Initializes a Hadoop `Configuration` object and creates a new `Job` instance with the name Wikipedia Timestamp Processing.
- **Job Configuration:**
  - Sets the main job class using `setJarByClass(WikipediaTimestampDriver.class)`.
  - Assigns `TimestampMapper` as the mapper and `MaxTimestampReducer` as the reducer.
  - Defines the output key-value types:

    * Mapper Output: `IntWritable` (some integer key, e.g., year) as the key and `Text` (timestamp information) as the value.
    * Reducer Output: `IntWritable` (same integer key) as the key and `Text` (processed timestamp information) as the value.
  - Sets the number of reducer tasks to 3 using `setNumReduceTasks(3)`.

- **Input and Output Paths:**
  - Reads the input path from the first command-line argument and assigns it using `FileInputFormat.addInputPath()`.
  - Reads the output path from the second argument and assigns it using `FileOutputFormat.setOutputPath()`.

- **Job Execution:**
  - Submits the job using `job.waitForCompletion(true)`.
  - Terminates the program with an exit code of 0 (success) or 1 (failure).

3.4 Discrepancies

These MapReduce functions did not match the latest Wikipedia article as the current reducer will find out the latest document ID for each index and put the word corresponding to it in the output. This will result in some indices which were not present in the latest document to have values from previous documents. So until and unless the latest document is also the longest one, we will have an output which has the words from the latest documents followed by words from previous documents.

3.5 Modifications

As the generated file does not match the actual Wikipedia article, we implemented another MapReduce function with the necessary modifications to address it:

1. The mapper remains the same.
2. The reducer will now keep track of a maximum document ID and will check the current document ID with the maximum document ID.
3. If they're equal, the word is appended to the set of latest words required, otherwise if the current doc ID is greater than the max doc ID, the max doc ID is updated and the latest words considered so far are discarded (as we have found a new latest document).
4. Finally, after we have constructed the latest words required, we write them to the output file using a cleanup function.

3.6 LatestTimestampReducer

The `LatestTimestampReducer` processes key-value pairs emitted by the Mapper to identify the words associated with the latest (maximum) document ID for each key. It also ensures that any indices having a document id less than the maximum are not reflected in the final input.

3.6.1 Input Format

The Reducer receives:

- **Key:** An integer (`IntWritable`) representing the document ID.
- **Values:** A list of comma-separated strings (`Text`), where each value follows the format ”index, word”.

3.6.2 Processing Steps

The Reducer follows these steps:

1. **Initialize Tracking Variables**
   - `maxDocID` is initialized to the smallest possible value (`Integer.MIN_VALUE`), tracking the latest document ID encountered.
   - `latestDocWords`, a `TreeMap<Integer, String>`, is initialized to store words from the latest document in index order.
2. **Iterate Over Input Values**
   - Extract the document ID from the key.
   - If the document ID is greater than `maxDocID`, update `maxDocID` and clear `latestDocWords`, ensuring only the latest document's data is stored.
   - If the document ID matches `maxDocID`, process each value:
     - Split the value using "," to extract `index` and `word`.
     - If the format is incorrect (i.e., missing required fields), skip the entry.
     - Convert `index` into an integer and trim the word.
     - Store the word in `latestDocWords`, maintaining order by index.
3. **Emit Ordered Words from the Latest Document**
   - In the `cleanup` method, iterate over `latestDocWords` in index order and emit each (`index`, `word`) pair as output.

3.6.3 Error Handling

- If a value cannot be parsed correctly (e.g., `index` is not a valid integer), the Reducer catches the `NumberFormatException` and skips the entry.
- The use of a `TreeMap` ensures words are emitted in sorted order based on index, maintaining structured output.

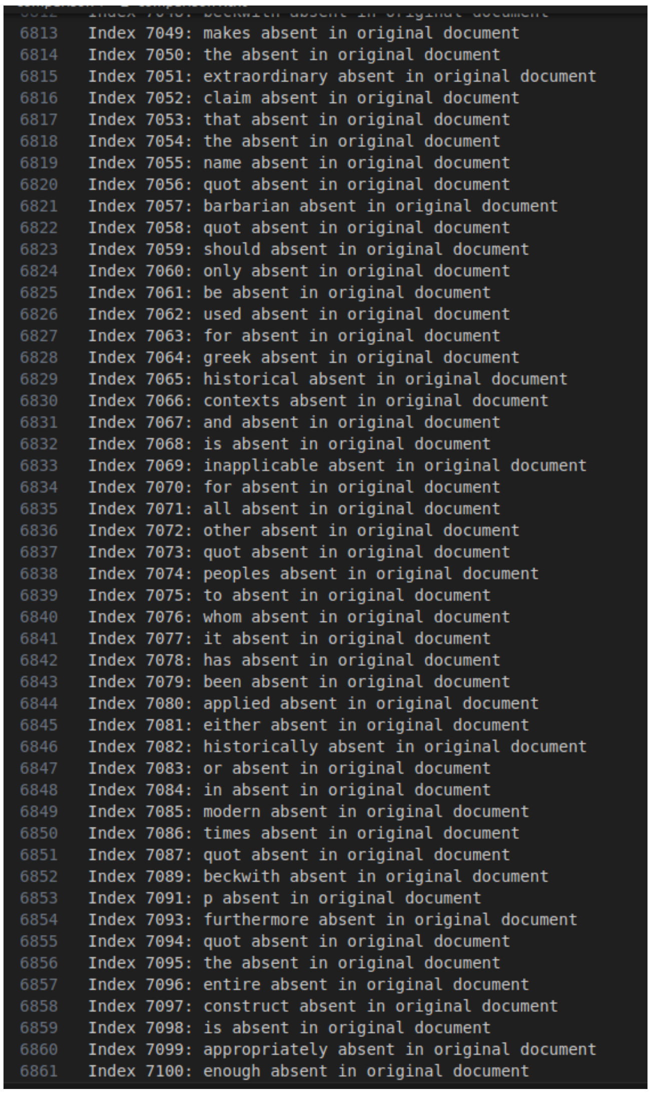
This Reducer effectively filters and emits words only from the latest document encountered in the dataset.

3.7 Verification of Results

We wrote a Python program to compare the document with the highest document id with the output obtained by the reducer for Q3. We observe that the output from the reducer for Q3 part 2 exactly aligns with the document, while the output for Q3 part 1 has extra lines, as expected. Figures 2 and 3 show the outputs obtained.

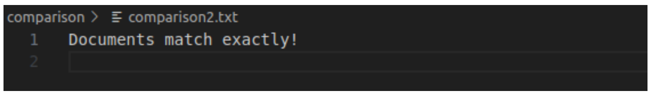
4 Runtime Analysis

We initially tried out the map and reducer in Q2 with the entire dataset. Uploading the dataset to HDFS took 30 minutes (Figure 4), and when we tried to run the map reducer with this dataset, we had to abort the operation as it had taken 43 minutes, and still was not close to completion (Figure 6).
Hence, we were forced to look at only the Wikipedia 50 dataset for our work. Uploading the dataset to HDFS took 4.3 seconds (Figure 5). Running the map reduce task for Q2 took 4.99 seconds (Figure 7), while the map and reduce tasks for Q3 part 1 and part 2 took 4.84 and 3.84 seconds respectively (Figures 8, 9).



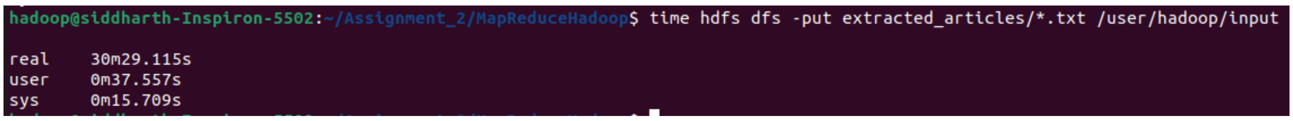
```
Index 7049: makes absent in original document
Index 7050: the absent in original document
Index 7051: extraordinary absent in original document
Index 7052: claim absent in original document
Index 7053: that absent in original document
Index 7054: the absent in original document
Index 7055: name absent in original document
Index 7056: quot absent in original document
Index 7057: barbarian absent in original document
Index 7058: quot absent in original document
Index 7059: should absent in original document
Index 7060: only absent in original document
Index 7061: be absent in original document
Index 7062: used absent in original document
Index 7063: for absent in original document
Index 7064: greek absent in original document
Index 7065: historical absent in original document
Index 7066: contexts absent in original document
Index 7067: and absent in original document
Index 7068: is absent in original document
Index 7069: inapplicable absent in original document
Index 7070: for absent in original document
Index 7071: all absent in original document
Index 7072: other absent in original document
Index 7073: quot absent in original document
Index 7074: peoples absent in original document
Index 7075: to absent in original document
Index 7076: whom absent in original document
Index 7077: it absent in original document
Index 7078: has absent in original document
Index 7079: been absent in original document
Index 7080: applied absent in original document
Index 7081: either absent in original document
Index 7082: historically absent in original document
Index 7083: or absent in original document
Index 7084: in absent in original document
Index 7085: modern absent in original document
Index 7086: times absent in original document
Index 7087: quot absent in original document
Index 7089: beckwith absent in original document
Index 7091: p absent in original document
Index 7093: furthermore absent in original document
Index 7094: quot absent in original document
Index 7095: the absent in original document
Index 7096: entire absent in original document
Index 7097: construct absent in original document
Index 7098: is absent in original document
Index 7099: appropriately absent in original document
Index 7100: enough absent in original document
```

Figure 2: Comparison between output of Q3 part 1 with the document with the highest id



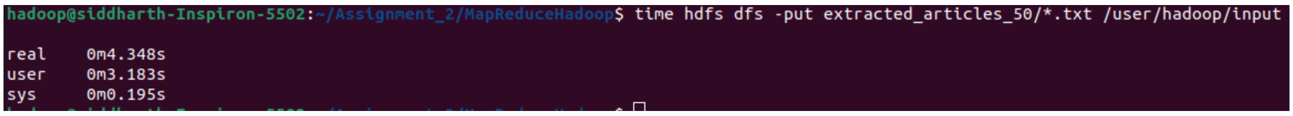
```
Documents match exactly!
```

Figure 3: Comparison between output of Q3 part 2 with the document with the highest id



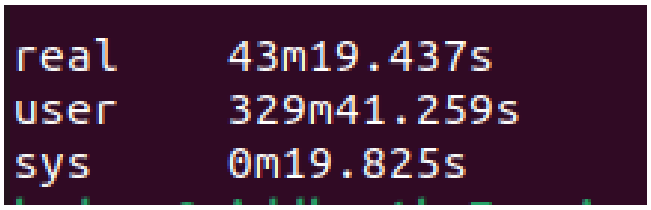
```
hadoop@siddharth-Inspiron-5502:~/Assignment_2/MapReduceHadoop$ time hdfs dfs -put extracted_articles/*.txt /user/hadoop/input

real    30m29.115s
user    0m37.557s
sys     0m15.709s
```

Figure 4: Time Required to Upload the entire dataset to HDFS



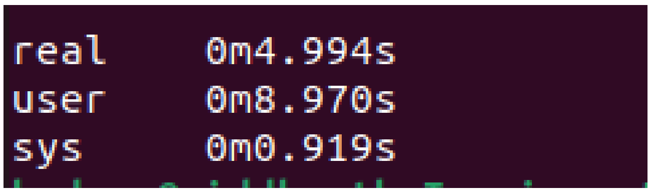
```
hadoop@siddharth-Inspiron-5502:~/Assignment_2/MapReduceHadoop$ time hdfs dfs -put extracted_articles_50/*.txt /user/hadoop/input

real    0m4.348s
user    0m3.183s
sys     0m0.195s
```

Figure 5: Time Required to Upload the Wikipedia 50 Dataset to HDFS



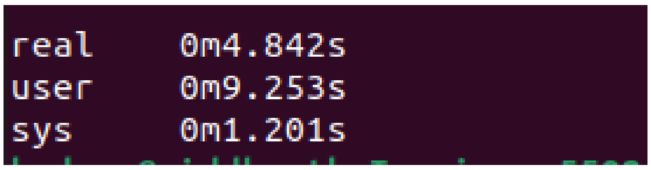
```
real	43m19.437s
user	329m41.259s
sys	0m19.825s
```

Figure 6: Map Reduce Time for Q2 with the entire dataset



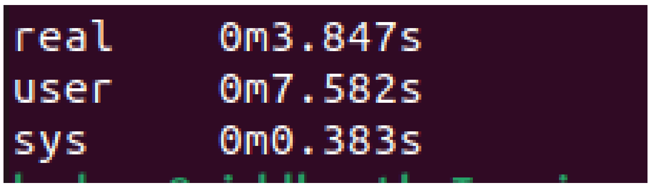
```
real	0m4.994s
user	0m8.970s
sys	0m0.919s
```

Figure 7: Map Reduce Time for Q2 with the Wikipedia 50 Dataset



```
real	0m4.842s
user	0m9.253s
sys	0m1.201s
```

Figure 8: Map Reduce Time for Q3 Part 1 with the Wikipedia 50 Dataset



```
real	0m3.847s
user	0m7.582s
sys	0m0.383s
```

Figure 9: Map Reduce Time for Q3 Part 2 with the Wikipedia 50 Dataset