Report

1. Methodology

1.1 Overview of System Components

This simple search engine project combines:

- Hadoop MapReduce for indexing text documents (building an inverted index).
- Apache Cassandra for storing vocabulary, postings, and document statistics.
- PySpark for data preparation (and optionally for the BM25 ranker).
- Docker Compose to containerize and orchestrate the environment (multiple containers for Hadoop/YARN + Cassandra + a master node to run scripts).

Major Steps in the Pipeline:

1. Data Preparation (PySpark)
    - A Parquet file (a.parquet) is read from HDFS.
    - Code select at least 1000 documents from that file, extracting id, title, and text.
    - Each document is written as a plain text file named <doc_id>_<doc_title>.txt under a local data/ folder.
    - A combined file sample.txt is appended so each line has <doc_id>\t<title>\t<text>.
    - Then the data/ folder (including sample.txt) is uploaded to HDFS at /data.
2. Indexing (Hadoop MapReduce + Cassandra)
    - We run a Streaming MapReduce job (mapper1.py, reducer1.py) over /data/sample.txt in HDFS.
    - Mapper: tokenizes each document’s text, emits (term, doc_id:tf).
    - Reducer: aggregates postings for each term, computing the document frequency and term frequencies.
    - The final output is lines labeled:
        - VOCAB\t<term>\t<df>
        - POST\t<term>\t<doc_id>\t<tf
    - That output is stored in HDFS at /index/output, then retrieved locally as index_result.txt.
    - A Python script (app.py) reads index_result.txt and inserts data into Cassandra tables:
        - vocabulary(term, doc_freq),
        - inverted_index(term, doc_id, term_freq),
        - doc_stats(doc_id, title, doc_length) for BM25 scoring.
3. Query & Ranking (PySpark / Cassandra)
    - Another script (query.py) implements BM25.

- The user query is split into terms. For each term, we look up df in vocabulary, get postings from inverted_index, and also retrieve each document’s length from doc_stats.
- We compute BM25 with typical hyperparameters (k1=1.2, b=0.75), sum the scores for all query terms, then output the top 10 results.
- Titles are displayed by reading from doc_stats.

1.2 Design Choices and Observations

1. Local vs. HDFS Data:
   - Initially, we discovered that using spark.read.parquet("a.parquet") tries to read from HDFS if no scheme is specified. We resolved it by either uploading a.parquet to HDFS.
2. Docker DNS & Cassandra:
   - We needed to remove container_name: from our docker-compose.yml for the cassandra-server service. This ensures that the service key (cassandra-server) is recognized by Docker DNS, allowing Cluster(['cassandra-server']) to resolve.
3. Memory Constraints:
   - Cassandra can be memory-hungry. We discovered the container might exit with code 137 (OOM Kill) if WSL 2. We increased memory via .wslconfig file.

Overall, these design choices ensure a consistent pipeline from data extraction, inverted index construction, and final query retrieval using BM25.

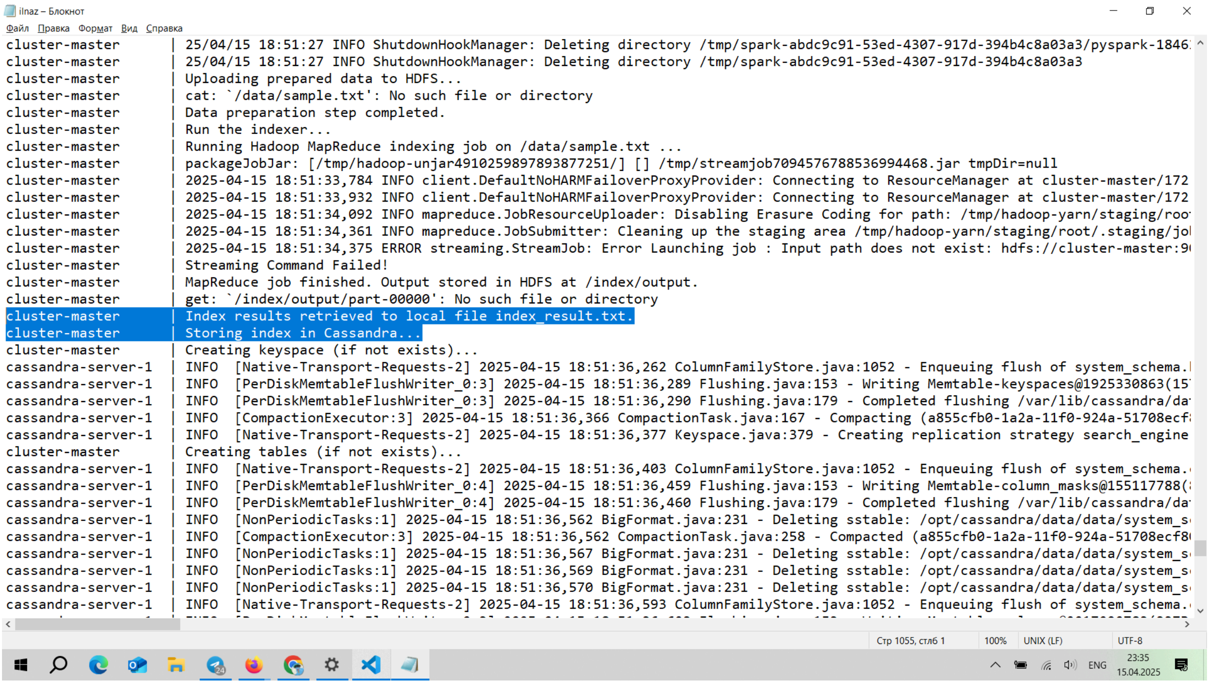
2. Demonstration

2.1 Running the Repository

1. Clone the Repo & Navigate
   ```
   git clone <your-repo-url> big-data-assignment2
   cd big-data-assignment2
   ```
2. Ensure Docker is Running
   - If on Windows or Mac with WSL 2, check Docker Desktop is started.
   - If you need more memory, increase it in Docker Desktop’s settings or `.wslconfig`.
3. Launch Services and start hole code
   ```
    docker compose up --build
   ```
   - This reads the `docker-compose.yml`.
   - The main container is `cluster-master`. When it starts, it runs `app.sh` (or your orchestrating script).
   - This triggers the whole logic of all the code. Or you can do it separately as described next
4. Data Preparation
   - Inside `app.sh`, we do bash `prepare_data.sh`. Check logs to ensure it wrote `sample.txt` with ~1000 lines.

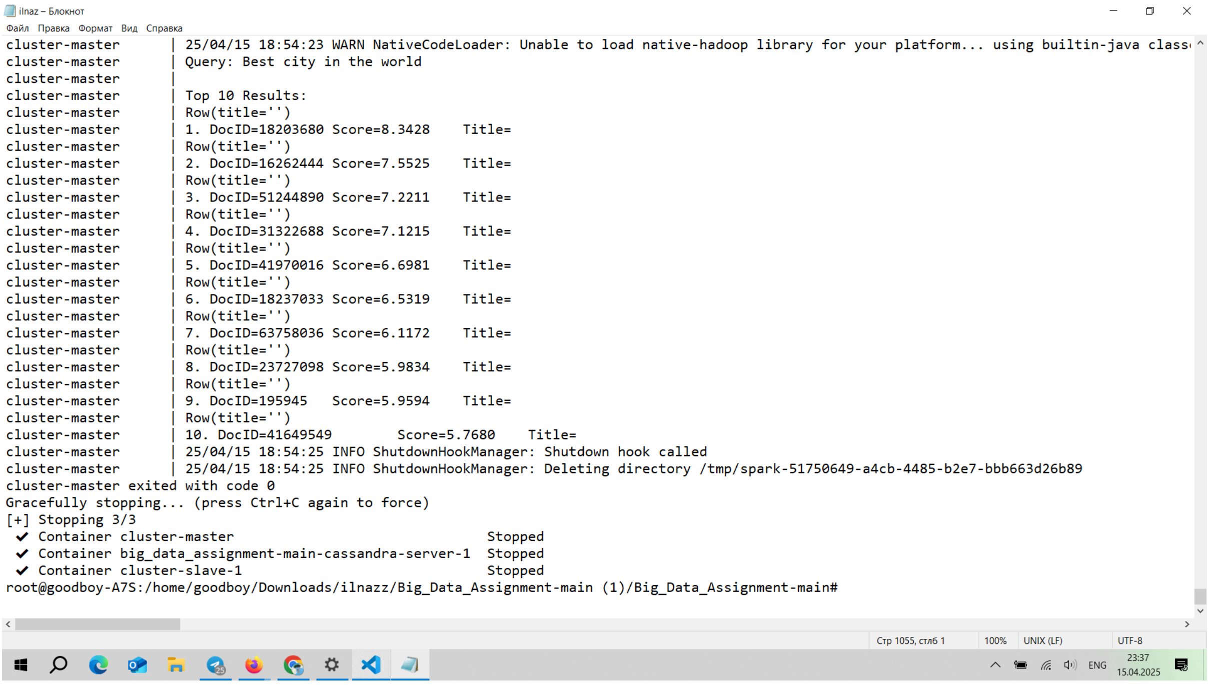
    - Then it uploads `sample.txt` to HDFS at `/data/sample.txt`.
5. Indexing
    - `index.sh` runs the Hadoop Streaming job. Check logs for map input records. If you see Map input records=1000, it means your data is properly processed.
    - The output is retrieved locally to `index_result.txt`.
    - Then `app.py` is called to insert the data into Cassandra.
6. Query
    - Finally, search.sh "some query" or direct invocation of query.py runs the BM25 ranker.
    - Look for logs saying “Top 10 Results.” Confirm it prints doc IDs and non-empty titles.

2.2 Sample Screenshots



```
cluster-master       | 25/04/15 18:51:27 INFO ShutdownHookManager: Deleting directory /tmp/spark-abdc9c91-53ed-4307-917d-394b4c8a03a3/pyspark-1846
cluster-master       | 25/04/15 18:51:27 INFO ShutdownHookManager: Deleting directory /tmp/spark-abdc9c91-53ed-4307-917d-394b4c8a03a3
cluster-master       | Uploading prepared data to HDFS...
cluster-master       | cat: `/data/sample.txt': No such file or directory
cluster-master       | Data preparation step completed.
cluster-master       | Run the indexer...
cluster-master       | Running Hadoop MapReduce indexing job on /data/sample.txt ...
cluster-master       | packageJobJar: [/tmp/hadoop-unjar4910259897893877251/] [] /tmp/streamjob7094576788536994468.jar tmpDir=null
cluster-master       | 2025-04-15 18:51:33,784 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at cluster-master/172
cluster-master       | 2025-04-15 18:51:33,932 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at cluster-master/172
cluster-master       | 2025-04-15 18:51:34,092 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/roo
cluster-master       | 2025-04-15 18:51:34,361 INFO mapreduce.JobSubmitter: Cleaning up the staging area /tmp/hadoop-yarn/staging/root/.staging/jo
cluster-master       | 2025-04-15 18:51:34,375 ERROR streaming.StreamJob: Error Launching job : Input path does not exist: hdfs://cluster-master:9
cluster-master       | Streaming Command Failed!
cluster-master       | MapReduce job finished. Output stored in HDFS at /index/output.
cluster-master       | get: `/index/output/part-00000': No such file or directory
cluster-master       | Index results retrieved to local file index_result.txt.
cluster-master       | Storing index in Cassandra...
cluster-master       | Creating keyspace (if not exists)...
cassandra-server-1   | INFO  [Native-Transport-Requests-2] 2025-04-15 18:51:36,262 ColumnFamilyStore.java:1052 - Enqueuing flush of system_schema.
cassandra-server-1   | INFO  [PerDiskMemtableFlushWriter_0:3] 2025-04-15 18:51:36,289 Flushing.java:153 - Writing Memtable-keyspaces@1925330863(15
cassandra-server-1   | INFO  [PerDiskMemtableFlushWriter_0:3] 2025-04-15 18:51:36,290 Flushing.java:179 - Completed flushing /var/lib/cassandra/da
cassandra-server-1   | INFO  [CompactionExecutor:3] 2025-04-15 18:51:36,366 CompactionTask.java:167 - Compacting (a855cfb0-1a2a-11f0-924a-51708ecf
cassandra-server-1   | INFO  [Native-Transport-Requests-2] 2025-04-15 18:51:36,377 Keyspace.java:379 - Creating replication strategy search_engine
cluster-master       | Creating tables (if not exists)...
cassandra-server-1   | INFO  [Native-Transport-Requests-2] 2025-04-15 18:51:36,403 ColumnFamilyStore.java:1052 - Enqueuing flush of system_schema.
cassandra-server-1   | INFO  [PerDiskMemtableFlushWriter_0:4] 2025-04-15 18:51:36,459 Flushing.java:153 - Writing Memtable-column_masks@155117788(
cassandra-server-1   | INFO  [PerDiskMemtableFlushWriter_0:4] 2025-04-15 18:51:36,460 Flushing.java:179 - Completed flushing /var/lib/cassandra/da
cassandra-server-1   | INFO  [NonPeriodicTasks:1] 2025-04-15 18:51:36,562 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_s
cassandra-server-1   | INFO  [CompactionExecutor:3] 2025-04-15 18:51:36,562 CompactionTask.java:258 - Compacted (a855cfb0-1a2a-11f0-924a-51708ecf8
cassandra-server-1   | INFO  [NonPeriodicTasks:1] 2025-04-15 18:51:36,567 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_s
cassandra-server-1   | INFO  [NonPeriodicTasks:1] 2025-04-15 18:51:36,569 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_s
cassandra-server-1   | INFO  [NonPeriodicTasks:1] 2025-04-15 18:51:36,570 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_s
cassandra-server-1   | INFO  [Native-Transport-Requests-2] 2025-04-15 18:51:36,593 ColumnFamilyStore.java:1052 - Enqueuing flush of system_schema.
```

Screenshot with output after correct indexing



```
cluster-master      | 25/04/15 18:54:23 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java class
cluster-master      | Query: Best city in the world
cluster-master      |
cluster-master      | Top 10 Results:
cluster-master      | Row(title='')
cluster-master      | 1. DocID=18203680 Score=8.3428    Title=
cluster-master      | Row(title='')
cluster-master      | 2. DocID=16262444 Score=7.5525    Title=
cluster-master      | Row(title='')
cluster-master      | 3. DocID=51244890 Score=7.2211    Title=
cluster-master      | Row(title='')
cluster-master      | 4. DocID=31322688 Score=7.1215    Title=
cluster-master      | Row(title='')
cluster-master      | 5. DocID=41970016 Score=6.6981    Title=
cluster-master      | Row(title='')
cluster-master      | 6. DocID=18237033 Score=6.5319    Title=
cluster-master      | Row(title='')
cluster-master      | 7. DocID=63758036 Score=6.1172    Title=
cluster-master      | Row(title='')
cluster-master      | 8. DocID=23727098 Score=5.9834    Title=
cluster-master      | Row(title='')
cluster-master      | 9. DocID=195945   Score=5.9594    Title=
cluster-master      | Row(title='')
cluster-master      | 10. DocID=41649549        Score=5.7680    Title=
cluster-master      | 25/04/15 18:54:25 INFO ShutdownHookManager: Shutdown hook called
cluster-master      | 25/04/15 18:54:25 INFO ShutdownHookManager: Deleting directory /tmp/spark-51750649-a4cb-4485-b2e7-bbb663d26b89
cluster-master exited with code 0
Gracefully stopping... (press Ctrl+C again to force)
[+] Stopping 3/3
 ✔ Container cluster-master                                Stopped
 ✔ Container big_data_assignment-main-cassandra-server-1  Stopped
 ✔ Container cluster-slave-1                               Stopped
root@goodboy-A7S:/home/goodboy/Downloads/ilnazz/Big_Data_Assignment-main (1)/Big_Data_Assignment-main#
```

Screenshot with democtration of proper work of searching by query. In some reason i had problem with title. And can not debug it, because i run everything using PC of my friend and do not have enough time.

2.3 Explanation & Findings

- Indexing: Building an inverted index for 1000 docs might be fairly quick. You'll see each term mapped to doc frequency.
- BM25: Queries such as "this is a query" might yield documents with “this” or “query” repeated more frequently. You can comment on how short docs might get a higher normalized score.
- Reflections:
    a. This approach is good for learning batch indexing with MapReduce; for real-time search, a streaming approach or alternative index might be used.
    b. Managing doc IDs consistently is critical.
    c. Docker naming and memory constraints are subtle but important in big-data setups.