

Big Data Assignment 2 report.

Ruslan Izmailov, r.izmailov@innopolis.university, DS-01

Methodology

Changes made:

Docker compose file:

1. To be able to see logs I have exposed some ports on cluster-slave-node.
2. The same volume as for the master node was mounted to the slave-node.

Step 0. Starting services and setuping environment.

`app.sh` beginning

This step is done without any general changes from the provided template.

1. Start ssh service
2. Start hadoop and yarn services
3. Create environment, install dependences and pack

Step 1. Data preparation.

`prepare_data.sh`

To preprocess all data we have I have implemented the following steps:

1. Clears existing hdfs filesystem (to be able to write there without file collisions)
2. Runs `prepare_data.py` script
 - Samples N files (specified in script)
 - Creates documents in local file system
3. Writes created files to hdfs
4. Runs `prepare_data_indexing.py` script
 - Read files from `hdfs://data/`
 - Process each document (using RDD), so it can have the following structure:
document_id document_name document_body
 - Save preprocessed documents to `hdfs://index/data`

Step 2. Creating an Index.

`index/index.sh`

1. Prepare directories for storing indexes: clear `hdfs://index/output`
2. Run mapreduce job:
 - Mapper:
 - Extracts words from each document (which length ≥ 3)
 - Sends to stdout: `word \t document_id`
 - Reducer:
 - Extracts word & document_id from input
 - Stores data in dictionary with the following structure:
word \rightarrow (document_id, frequency)
 - After processing input sends to stdout: `word \t document_id \t frquency`
that would be placed in the

3. Initialise cassandra schemas

Table: vocabulary_index		
Column	Type	Key
term	TEXT	K
doc_id	TEXT	C ↑
tf	INT	

Table: document_info		
Column	Type	Key
doc_id	TEXT	K
title	TEXT	
doc_length	INT	

Table: collection_stats		
Column	Type	Key
key	TEXT	K
total_docs	INT	
avg_doc_length	DOUBLE	

Table: term_stats		
Column	Type	Key
term	TEXT	K
df	INT	

4. Store documents

- Stores all documents, titles & documents sizes to the document_info table

5. Store to cassandra

- Parses input generated by the mapreduce job and loading processed data to the vocabulary_index, term_stats
- Writes a global statistics to the collection_stats.

2.1 Indexing file from not default path

Script is also working in a mode of adding & indexing files to the search engine. To do it you need to run a system without an entrypoint specified in the docker-compose.yaml file, but with basic start setup. Then you connect to the container, run the `start-services.sh` & `prepare_data.sh` scripts, basically it means that indexing is only available, when system started.

To your own file to the indexing you run `bash index/index.sh 123_my_own_file.txt` then system would prepare it, index and finally add to the cassandra tables. In case such file already indexed (*term* & *doc_id* present in cassandra) it would just ignore it.

Step 3. Ranking documents.

`search.sh`

Hyperparameters of the BM25:

k1 = 1.5

b = 0.75

Other notes:

- Spark cassandra adapter is used (due to the warnings I obtained)
- I was not able to manage to make script send its result to stdout using `--deploy-mode cluster` so `query.py` writes result to the `hdfs://query` and script `search.sh` aggregates them (just sends results of all jobs to from `hdfs://query` to stdout, as it is stores retrieval results ordered by default).

In `--deploy-mode client` script just returns retrieval results in stdout without any other actions.

Steps to retrieve documents:

1. Retrieving main config

2. Loads the vocabulary index, term statistics, and document information from Cassandra into RDDs.
3. Filters the term statistics to include only the terms present in the query.
4. Computes the BM25 score for each document based on the tf & idf.
5. Ranks documents based on the scores, and cuts top 10 results.
6. Saves files to `hdfs://query` .

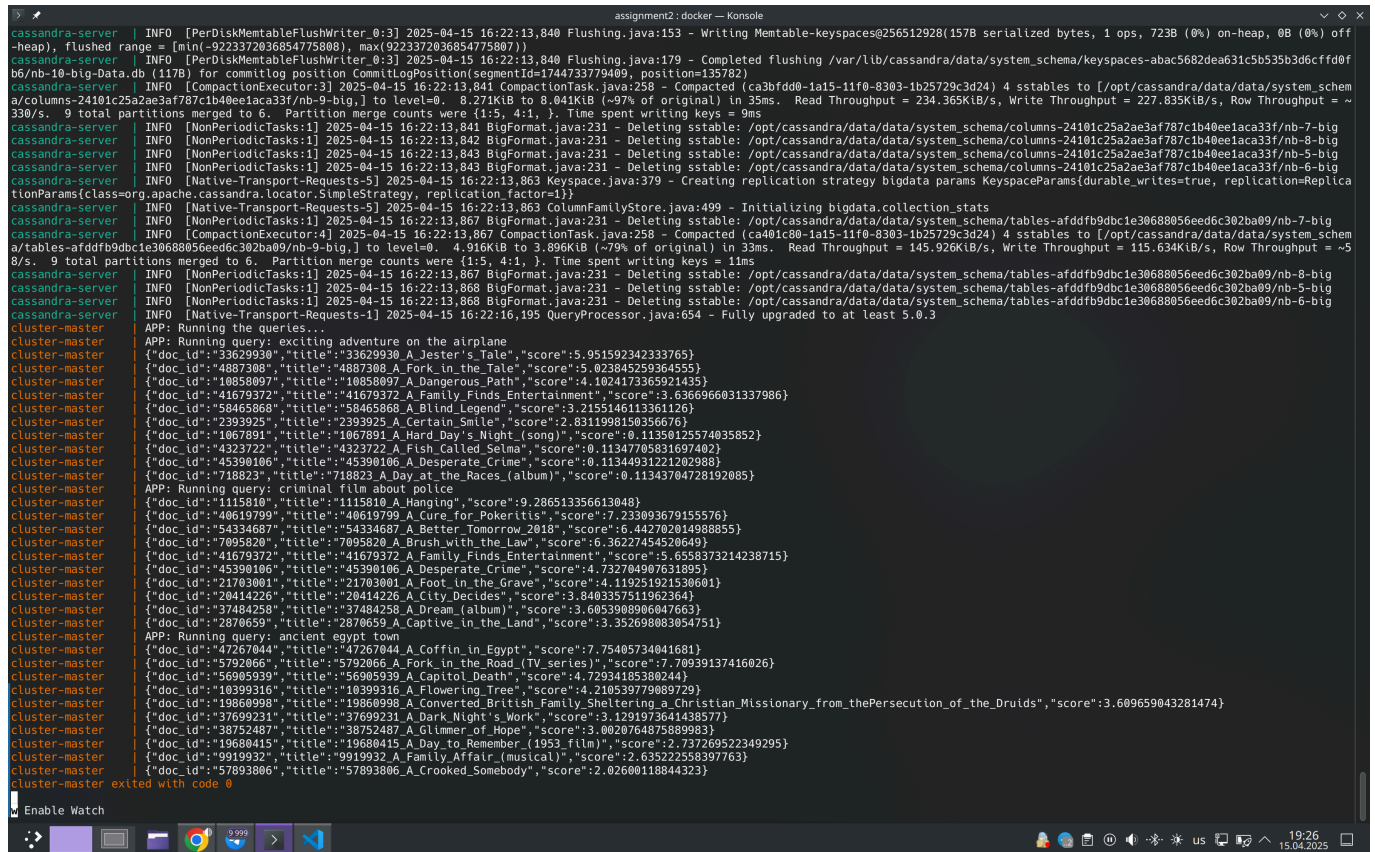
Demonstration

Guide how to run repository:

```
docker compose up --build
```

This will perform all needed actions to start, also it will run 3 predefined queries:

1. exciting adventure on the airplane
2. criminal film about police
3. ancient egypt town



The screenshot shows a terminal window titled "assignment2: docker — Konsole". It displays logs from a Cassandra cluster and the results of three queries. The logs show various background tasks like flushing, compacting, and deleting sstables. The queries are executed by the 'cluster-master' and return JSON arrays of document scores.

```
assignment2: docker — Konsole
cassandra-server INFO [PerDiskMemtableFlushWriter_0:3] 2025-04-15 16:22:13,840 Flushing.java:153 - Writing Memtable-keyspaces@25651928(157B serialized bytes, 1 ops, 723B (0%) on-heap, 0B (0%) off-heap), flushed range = [min(-9223372036854775808), max(9223372036854775807)]
cassandra-server INFO [PerDiskMemtableFlushWriter_0:3] 2025-04-15 16:22:13,840 Flushing.java:179 - Completed flushing /var/lib/cassandra/data/system_schema/keyspaces-abac5682dea631c5b535b3d6cfd0f66/nb-10-big-data.db (117B) for commitlog position CommitLogPosition(segmentId=1744733779409, position=135782)
cassandra-server INFO [CompactionExecutor:3] 2025-04-15 16:22:13,841 CompactionTask.java:258 - Compacted (ca3bfdd0-1a15-11f0-8303-1b25729c3d24) 4 sstables to [/opt/cassandra/data/data/system_schema/columns-24101c25a2ae3af787c1b40ee1aca33f/nb-9-big] to level=0. 8.271KiB to 8.041KiB (~97% of original) in 35ms. Read Throughput = 234.365KiB/s, Write Throughput = 227.835KiB/s, Row Throughput = ~330/s. 9 total partitions merged to 6. Partition merge counts were {1:5, 4:1, }. Time spent writing keys = 9ms
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,841 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/columns-24101c25a2ae3af787c1b40ee1aca33f/nb-7-big
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,842 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/columns-24101c25a2ae3af787c1b40ee1aca33f/nb-8-big
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,843 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/columns-24101c25a2ae3af787c1b40ee1aca33f/nb-5-big
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,843 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/columns-24101c25a2ae3af787c1b40ee1aca33f/nb-6-big
cassandra-server INFO [Native-Transport-Requests-5] 2025-04-15 16:22:13,863 Keyspace.java:379 - Creating replication strategy bigdata.params KeyspaceParams{durable_writes=true, replication=ReplicationParams{class=org.apache.cassandra.locator.SimpleStrategy, replication_factor=1}}
cassandra-server INFO [Native-Transport-Requests-5] 2025-04-15 16:22:13,863 ColumnFamilyStore.java:499 - Initializing bigdata.collection_stats
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,867 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/tables-afddf9bdbc1e30688056eed6c302ba09/nb-7-big
cassandra-server INFO [CompactionExecutor:4] 2025-04-15 16:22:13,867 CompactionTask.java:258 - Compacted (ca01e68-1a15-11f0-8303-1b25729c3d24) 4 sstables to [/opt/cassandra/data/data/system_schema/tables-afddf9bdbc1e30688056eed6c302ba09/nb-9-big] to level=0. 4.916KiB to 3.896KiB (~79% of original) in 33ms. Read Throughput = 145.926KiB/s, Write Throughput = 115.634KiB/s, Row Throughput = ~58/s. 9 total partitions merged to 6. Partition merge counts were {1:5, 4:1, }. Time spent writing keys = 11ms
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,867 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/tables-afddf9bdbc1e30688056eed6c302ba09/nb-8-big
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,868 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/tables-afddf9bdbc1e30688056eed6c302ba09/nb-5-big
cassandra-server INFO [NonPeriodicTasks:1] 2025-04-15 16:22:13,868 BigFormat.java:231 - Deleting sstable: /opt/cassandra/data/data/system_schema/tables-afddf9bdbc1e30688056eed6c302ba09/nb-6-big
cassandra-server INFO [Native-Transport-Requests-1] 2025-04-15 16:22:16,195 QueryProcessor.java:654 - Fully upgraded to at least 5.0.3
cluster-master APP: Running the queries...
cluster-master APP: Running query: exciting adventure on the airplane
cluster-master {
  "doc_id": "33629930", "title": "33629930_A_Jester's Tale", "score": 5.95159234233765}
cluster-master {
  "doc_id": "4887308", "title": "4887308_A_Fork_in_the Tale", "score": 5.023845259364555}
cluster-master {
  "doc_id": "10858097", "title": "10858097_A_Dangerous Path", "score": 4.1024173365921435}
cluster-master {
  "doc_id": "41679372", "title": "41679372_A_Family Finds Entertainment", "score": 3.6366966031337986}
cluster-master {
  "doc_id": "58465868", "title": "58465868_A_Blind Legend", "score": 3.2155146113361126}
cluster-master {
  "doc_id": "2393925", "title": "2393925_A_Certain Smile", "score": 2.8311998150356676}
cluster-master {
  "doc_id": "1067891", "title": "1067891_A_Hard Day's Night (song)", "score": 0.11358125574035852}
cluster-master {
  "doc_id": "4323722", "title": "4323722_A_Fish Called Selma", "score": 0.11347705831697402}
cluster-master {
  "doc_id": "45390106", "title": "45390106_A_Desperate Crime", "score": 0.1134931221202988}
cluster-master {
  "doc_id": "718023", "title": "718023_A Day at the Races (album)", "score": 0.11343704728192085}
cluster-master APP: Running query: criminal film about police
cluster-master {
  "doc_id": "1115810", "title": "1115810_A_Hanging", "score": 9.286513356613048}
cluster-master {
  "doc_id": "40619799", "title": "40619799_A_Cure_for Pokeritis", "score": 7.233093679155576}
cluster-master {
  "doc_id": "54334687", "title": "54334687_A_Better Tomorrow 2018", "score": 6.442702014988855}
cluster-master {
  "doc_id": "7095820", "title": "7095820_A_Brush with the Law", "score": 6.36227454520649}
cluster-master {
  "doc_id": "41679372", "title": "41679372_A_Family Finds Entertainment", "score": 5.6558373214238715}
cluster-master {
  "doc_id": "45390106", "title": "45390106_A_Desperate Crime", "score": 4.732704907631805}
cluster-master {
  "doc_id": "21703001", "title": "21703001_A_Foot in the Grave", "score": 4.119251921530601}
cluster-master {
  "doc_id": "20414226", "title": "20414226_A_City Decides", "score": 3.8403357511962364}
cluster-master {
  "doc_id": "37484258", "title": "37484258_A_Dream (album)", "score": 3.6053908906047663}
cluster-master {
  "doc_id": "2870659", "title": "2870659_A_Captive in the Land", "score": 3.352698083054751}
cluster-master APP: Running query: ancient egypt town
cluster-master {
  "doc_id": "47267044", "title": "47267044_A_Coffin in Egypt", "score": 7.75405734041681}
cluster-master {
  "doc_id": "5792066", "title": "5792066_A_Fork in the Road (TV series)", "score": 7.70939137416026}
cluster-master {
  "doc_id": "56905939", "title": "56905939_A_Capitol Death", "score": 4.72934185380244}
cluster-master {
  "doc_id": "10399316", "title": "10399316_A_Flowering Tree", "score": 4.210539779089729}
cluster-master {
  "doc_id": "1980998", "title": "1980998_A_Converted British Family Sheltering a Christian Missionary from the Persecution of the Druids", "score": 3.609659043281474}
cluster-master {
  "doc_id": "37699231", "title": "37699231_A_Dark Night's Work", "score": 3.1291973641438577}
cluster-master {
  "doc_id": "38752487", "title": "38752487_A_Glimmer of Hope", "score": 3.0020764875889983}
cluster-master {
  "doc_id": "19680415", "title": "19680415_A Day to Remember (1953 film)", "score": 2.737269522349295}
cluster-master {
  "doc_id": "9919932", "title": "9919932_A_Family Affair (musical)", "score": 2.635222558397763}
cluster-master {
  "doc_id": "57893806", "title": "57893806_A_Crooked Somebody", "score": 2.02600118844323}
cluster-master exited with code 0
```

I have explored top 1-3 documents for each query and they all contained all/most of words from the initial query. Also the score is highly grows, when target word (word from query) present several times in the documents.