MapReduce Program+ Full Inverted Index

Snit Daniel Zerea
05/06/2024

Table of Content

Introduction

➢ **Objective:**

Develop a MapReduce program to create a Full Inverted Index from multiple text files.

➢ **Steps Overview:**

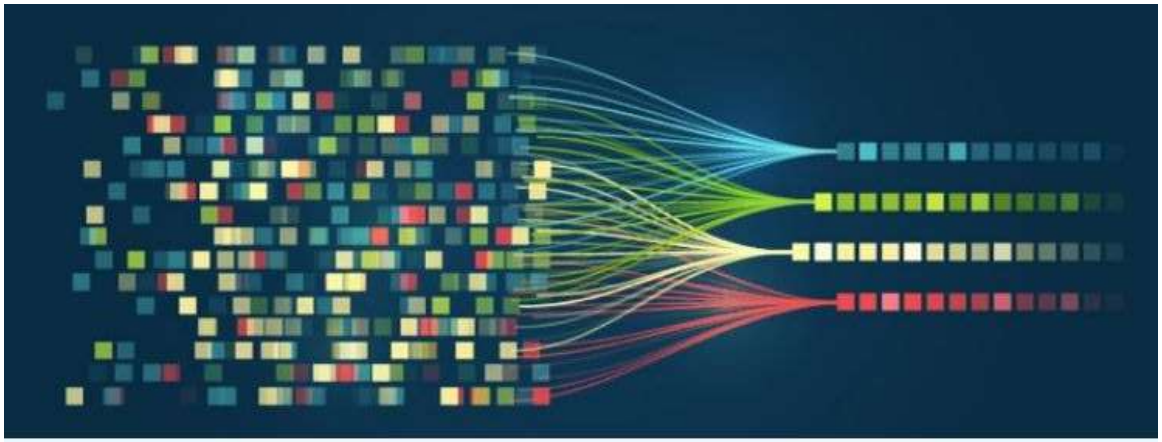
1. Draw tables showing the processes of mapper, combiner, and reducer for a Full Inverted Index.
2. Implement the Partial Inverted Index MapReduce program.
3. Convert the Partial Inverted Index to a Full Inverted Index MapReduce program.

Design

- **Problem Identification:**
  - Efficiently creating an inverted index for large datasets.
- **Solution Investigation:**
  - **Alternatives:**
    - Single-threaded processing: Simpler to implement but less efficient for large datasets.
    - MapReduce: Designed for parallel processing, better for large-scale data.

- **Comparison:**
  - **Scalability:** MapReduce can handle large data volumes by distributing the workload across multiple nodes.
  - **Performance:** MapReduce leverages parallel processing, reducing the overall processing time.
  - **Fault Tolerance:** MapReduce provides built-in fault tolerance by re-executing failed tasks on other nodes.
- **Selected Solution:**
  - MapReduce for handling large text datasets efficiently due to its scalability, performance, and fault tolerance.

- **Full Inverted Index Process:**
  - **Mapper:** Reads input files, processes each word, and emits intermediate key-value pairs.
  - **Combiner:** Local aggregation of data to reduce network load.
  - **Reducer:** Aggregates all values associated with a key.



Implementation and Testing

- Three Input Files Used in this project:
  - File 0: "it is what it is"
  - File 1: "what is it"
  - File 2: "it is a banana"

Step 1: Draw three tables to show the processes done by mapper, combiner, and reducer to show the Full Inverted Index of these three files.

Job: Full Inverted Index											
Map Task								Reduce Task			
Map()				Combine()				Reduce()			
Input (Given)		Output (Program)		Input (Given)		Output(program)		Input (Given)		Output (Program)	
Key	value	Key	Value	Key	Value	key	value	Key	Value	Key	Value
File 0	it is what it is	it	(0,0)	it	{(0,0), (0,3)}	it	{(0,0), (0,3)}	a	{(2,2)}	a	{(2,2)}
		is	(0,1)	is	{(0,1), (0,4)}	is	{(0,1), (0,4)}	banana	{(2,3)}	banana	{(2,3)}
		what	(0,2)	what	{(0,2)}	what	{(0,2)}	is	{(0,1), (0,4), (1,1), (2,1)}	is	{(0,1), (0,4), (1,1), (2,1)}
		it	(0,3)					it	{(0,0), (0,3), (1,2), (2,0)}	it	{(0,0), (0,3), (1,2), (2,0)}
		is	(0,4)					what	{(0,2), (1,0)}	what	{(0,2), (1,0)}
File 1	what is it	what	(1,0)	what	{(1,0)}	what	{(1,0)}				
		is	(1,1)	is	{(1,1)}	is	{(1,1)}				
		it	(1,2)	it	{(1,2)}	it	{(1,2)}				
File 2	it is a banana	it	(2,0)	it	{(2,0)}	it	{(2,0)}				
		is	(2,1)	is	{(2,1)}	is	{(2,1)}				
		a	(2,2)	a	{(2,2)}	a	{(2,2)}				
		banana	(2,3)	banana	{(2,3)}	banana	{(2,3)}				

Step 2: Convert a WordCount MapReduce program into a Partial Inverted Index MapReduce program.

- First, created the input files on my local filesystem.
  - Mkdir -p input_files
  - echo "it is what it is" > input_files/file0.txt
  - echo "what is it" > input_files/file1.txt
  - echo "it is a banana" > input_files/file2.txt
- Started HDFS
  - sbin/start-dfs.sh

```
szerea56456@cs-570-instance:~/hadoop-3.3.5$ mkdir -p input_files
szerea56456@cs-570-instance:~/hadoop-3.3.5$ echo "it is what it is" > input_files/file0.txt
szerea56456@cs-570-instance:~/hadoop-3.3.5$ echo "what is it" > input_files/file1.txt
szerea56456@cs-570-instance:~/hadoop-3.3.5$ echo "it is a banana" > input_files/file2.txt
szerea56456@cs-570-instance:~/hadoop-3.3.5$
```

- Created Input Directory in HDFS:
  - hdfs dfs -mkdir /input
- Created Input Directory in HDFS:
  - hdfs dfs -mkdir /input
- Uploaded the input files to the HDFS input directory:
  - hdfs dfs -put input_files/* /input

```
szerea56456@cs-570-instance:~/hadoop-3.3.5$ hdfs dfs -put input_files/* /input
```

```
szerea56456@cs-570-instance:~/hadoop-3.3.5$ mkdir partial_inverted_index
```

```
szerea56456@cs-570-instance:~/hadoop-3.3.5$ cd partial_inverted_index
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$
```

- To implemented the Partial Inverted Index Program, first I created the Java Program in the partial_inverted_index directory I created.
  - mkdir partial_inverted_index
  - cd partial_inverted_index
  - vi partialInvertedIndex.java

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$ vi PartialInvertedIndex.java
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$
```

SSH-in-browser UPLOAD

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class PartialInvertedIndex {

    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
        private Text word = new Text();
        private IntWritable fileIndex = new IntWritable();

        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
            FileSplit fileSplit = (FileSplit) reporter.getInputSplit();
            String fileName = fileSplit.getPath().getName();
            int fileNumber = Integer.parseInt(fileName.replaceAll("[^0-9]", ""));
            fileIndex.set(fileNumber);
```

- I Compiled it and Create JAR:
  - javac -classpath `hadoop classpath` -d . PartialInvertedIndex.java
  - jar -cvf partialinvertedindex.jar org/

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$ javac -classpath `hadoop classpath` -d . PartialInvertedIndex.java
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$ jar -cvf partialinvertedindex.jar org/
added manifest
adding: org/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/PartialInvertedIndex.class(in = 1550) (out= 746)(deflated 51%)
adding: org/myorg/PartialInvertedIndex$Map.class(in = 2463) (out= 1043)(deflated 57%)
adding: org/myorg/PartialInvertedIndex$Reduce.class(in = 2253) (out= 980)(deflated 56%)
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$
```

- Then I run the MapReduce job:
  - hadoop jar partialinvertedindex.jar org.myorg.PartialInvertedIndex /input /partial_output

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$ hadoop jar partialinvertedindex.jar org.myorg.Part
2024-06-04 18:42:45,371 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-06-04 18:42:45,671 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-06-04 18:42:45,672 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-06-04 18:42:45,701 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
Exception in thread "main" org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory hdfs://localhost:900
        at org.apache.hadoop.mapred.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:131)
        at org.apache.hadoop.mapreduce.JobSubmitter.checkSpecs(JobSubmitter.java:279)
        at org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:143)
        at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1678)
        at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1675)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1899)
        at org.apache.hadoop.mapreduce.Job.submit(Job.java:1675)
```

- Then got the partial inverted index output:
  - hdfs dfs -cat /partial_output/part-00000

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/partial_inverted_index$ hdfs dfs -cat /partial_output/part-00000
a:      {2}
banana: {2}
is:     {0, 1, 2}
it:     {0, 1, 2}
what:   {0, 1}
```

Step 3: Convert a Partial Inverted Index MapReduce program into a Full Inverted Index MapReduce program.

- First, I created a directory "full_inverted_index" then created a Java Program there.
  - mkdir full_inverted_index
  - cd full_inverted_index
- Created and Saved the FullInvertedIndex.java with the following content:

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/full_inverted_index$ vi FullInvertedIndex.java
```

SSH-in-browser

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class FullInvertedIndex {

    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {
        private Text word = new Text();
        private Text fileAndPosition = new Text();

        public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter) throws IOException {
            FileSplit fileSplit = (FileSplit) reporter.getInputSplit();
            String fileName = fileSplit.getPath().getName();
            int fileNumber = Integer.parseInt(fileName.replaceAll("[^0-9]", ""));
            String line = value.toString();
            String[] words = line.split("\\s+");

            for (int i = 0; i < words.length; i++) {
                word.set(words[i]);
                fileAndPosition.set(fileNumber + "," + i);
                output.collect(word, fileAndPosition);
            }
        }
    }
```

- Then compiled it and created JAR:
  - javac -classpath `hadoop classpath` -d . FullInvertedIndex.java
  - jar -cvf fullinvertedindex.jar org/

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/full_inverted_index$ javac -classpath `hadoop classpath` -d . FullInvertedIndex.java
szerea56456@cs-570-instance:~/hadoop-3.3.5/full_inverted_index$ jar -cvf fullinvertedindex.jar org/
added manifest
adding: org/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/FullInvertedIndex.class(in = 1497) (out= 725)(deflated 51%)
adding: org/myorg/FullInvertedIndex$Reduce.class(in = 1848) (out= 780)(deflated 57%)
adding: org/myorg/FullInvertedIndex$Map.class(in = 2533) (out= 1068)(deflated 57%)
szerea56456@cs-570-instance:~/hadoop-3.3.5/full_inverted_index$
```

- Finally I run the Job:
  - hadoop jar fullinvertedindex.jar org.myorg.FullInvertedIndex /input /full_output

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/full_inverted_index$ hadoop jar fullinvertedindex.jar org.myorg.FullInvertedIndex /input /full_output
2024-06-04 18:59:59,889 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-06-04 19:00:00,074 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-06-04 19:00:00,074 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-06-04 19:00:00,101 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-06-04 19:00:00,381 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolR
unner to remedy this.
2024-06-04 19:00:00,635 INFO mapred.FileInputFormat: Total input files to process : 0
2024-06-04 19:00:00,657 INFO mapreduce.JobSubmitter: number of splits:0
2024-06-04 19:00:00,988 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local773474632_0001
2024-06-04 19:00:00,988 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-06-04 19:00:01,317 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-06-04 19:00:01,321 INFO mapreduce.Job: Running job: job_local773474632_0001
2024-06-04 19:00:01,333 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-06-04 19:00:01,341 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-06-04 19:00:01,354 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
```

- Here I viewed the final full inverted index output
  - hdfs dfs -cat /full_output/part-00000

```
szerea56456@cs-570-instance:~/hadoop-3.3.5/full inverted index$ hdfs dfs -cat /full output/part-00000
a:      {(2, 2)}
banana: {(2, 3)}
is:     {(0, 1), (0, 4), (1, 1), (2, 1)}
it:     {(0, 0), (0, 3), (1, 2), (2, 0)}
what:   {(0, 2), (1, 0)}
```

Enhancement Ideas

- **Optimized Algorithms:**
  - Implement more efficient mapping and reducing functions.
- **Real-Time Updates:**
  - Integrate real-time indexing capabilities.
- **Fault Tolerance:**
  - Enhance error handling and recovery mechanisms.
- **Scalability:**
  - Optimize resource allocation for handling larger datasets.

Conclusion

- Successfully implemented MapReduce programs for both partial and full inverted indexes.
- Demonstrated effective use of Hadoop for large-scale text processing.
- Established a foundation for further exploration of advanced indexing techniques.
- Future Work will be to Implement and test proposed enhancements to improve performance and scalability.

References

- https://hc.labnet.sfbu.edu/~henry/npu/classes/javascript/node_js/course/nodeschool/learnyounode/http_json_api_server.html
- https://www.youtube.com/watch?v=cju2NqPEcp4&t=39s
- https://cloud.google.com/docs/overview
- https://joecreager.com/learnyounode-lesson-10-time-server/

Link to Github

https://github.com/snit-daniel/Big-Data-Processing-Analytics/blob/main/MapReduce/Full%20Inverted%20Index