|  |  |
| --- | --- |
| **Faculty of Computers and Information Technology**  **Master of Cyber-Physical Systems (CPS)** |  |

**Optimizing Task Scheduling in Heterogeneous MapReduce Environments Using Intelligent Algorithms**

* **Paper (Appendix)**
* **Course: Cloud Computing**
* **Supervisor**: **Dr. Hazem A. Elbaz**
* **Prepared by**: **Fadi Younis Baker**
* **Date**: **23/08/2025**

**Appendix A – Datasets**

This appendix provides the datasets used in the experimental evaluation, along with download links and sample commands.

**A.1 Enron Email Dataset**

Size: ~500,000 emails from ~150 users

Use Case: WordCount & text analysis

Download: Enron Dataset – CMU

<https://www.cs.cmu.edu/~enron/>

**Command**:

Wget

https://www.cs.cmu.edu/~enron/enron\_mail\_20110402.tar.gz

tar -xzf enron\_mail\_20110402.tar.gz

**A.2 Google Web Graph (SNAP)**

Size: 916,428 nodes, 5,105,039 edges

Use Case: PageRank & graph analysis

Download: SNAP Web-Google

<https://snap.stanford.edu/data/web-Google.html>

**Command**:

Wget

https://snap.stanford.edu/data/web-Google.txt.gz

gunzip web-Google.txt.gz

**A.3 Synthetic Data (Hadoop RandomTextWriter)**

Description: Generates large-scale random text for benchmarking (10–200 GB).

**Command**:

Hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar randomtextwriter /bench/randomtext-10g

**Appendix B - Code Listings**

**B.1 WordCount with Combiner (Java)**

The following Java code implements the WordCount program with a Combiner function to reduce shuffle traffic.

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

// Combiner to reduce intermediate data

public static class IntSumCombiner extends IntSumReducer {}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumCombiner.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**B.2 LATE Scheduling Simulator (Python – Pseudocode)**

# Estimate finish time based on progress rate

def estimated\_time\_to\_end(task):

progress\_rate = task.progress / max(task.runtime, 1e-9)

if progress\_rate <= 0:

return float("inf")

return (1.0 - task.progress) / progress\_rate

# LATE Scheduler

for task in running\_tasks:

ete = estimated\_time\_to\_end(task)

if ete > average\_ete \* 1.5: # Straggler threshold

launch\_speculative\_copy(task)

**B.3 AI-based Scheduler (Python – Machine Learning)**

import pandas as pd

from sklearn.ensemble import GradientBoostingRegressor

# Training data: logs with task features and runtime

data = pd.read\_csv("cluster\_logs.csv")

X = data[["cpu\_usage", "io\_speed", "net\_latency", "mem\_usage"]]

y = data["task\_runtime"]

# Train ML model

model = GradientBoostingRegressor(random\_state=42)

model.fit(X, y)

# Predict runtime for a new task

def schedule\_task(node\_features):

predicted\_runtime = model.predict([node\_features])[0]

if predicted\_runtime > 1.3 \* cluster\_mean\_runtime:

return "speculate"

return "normal"

**Appendix C - Execution Instructions**

**C.1 Running WordCount on Enron Dataset**

hdfs dfs -mkdir -p /data/enron

hdfs dfs -put data/enron/maildir /data/enron/

hadoop jar target/your-jar-name.jar org.example.WordCount /data/enron /out/wordcount

**C.2 Generating Synthetic Data and Sort**

hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar randomtextwriter /bench/randomtext-10g

hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar sort /bench/randomtext-10g /bench/sorted-10g

**C.3 Enabling Speculative Execution (mapred-site.xml)**

<property>

<name>mapreduce.map.speculative</name>

<value>true</value>

</property>

<property>

<name>mapreduce.reduce.speculative</name>

<value>true</value>

</property>