Hadoop Cluster Usage Guide

Table of Contents

Basic HDFS Operations

Checking Cluster Status

```
# View HDFS report
hdfs dfsadmin -report

# Check cluster health
hdfs fsck /

# List DataNodes
hdfs dfsadmin -listDatanodeReport
```

File System Operations

```
# Create directory
hdfs dfs -mkdir /user/example

# Upload file
hdfs dfs -put localfile.txt /user/example/

# Download file
hdfs dfs -get /user/example/file.txt

# List files
hdfs dfs -ls /user/example

# Delete file/directory
hdfs dfs -rm -r /user/example/file.txt

# Check file content
hdfs dfs -cat /user/example/file.txt

# Copy files between HDFS locations
hdfs dfs -cp /user/source/file.txt /user/dest/
```

Running MapReduce Jobs

Python MapReduce Structure

1. Mapper Template:

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
 # Process input and emit key-value pairs
 key, value = process_line(line)
 print(f'{key}\t{value}')
```

2. Reducer Template:

```
#!/usr/bin/env python3
import sys

current_key = None
values = []

for line in sys.stdin:
    key, value = line.strip().split('\t')
    # Process values for each key
    process_values(key, value)
```

Running MapReduce Jobs

```
hadoop jar $HADOOP_STREAMING_JAR \
    -files mapper.py,reducer.py \
    -mapper "python3 mapper.py" \
    -reducer "python3 reducer.py" \
    -input /user/input/* \
    -output /user/output
```

Stock Market Analysis Example

Setup

1. Create project directory:

   ```
   mkdir stock_analysis
   cd stock_analysis
   ```

2. Create required files:

generate_stock_data.py:

```python
#!/usr/bin/env python3
import csv
import random
from datetime import datetime, timedelta

def generate_stock_data():
    stocks = ['AAPL', 'GOOGL', 'MSFT', 'AMZN', 'META', 'TSLA', 'NVDA', 'AMD']

    with open('stock_data.csv', 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['date', 'symbol', 'open', 'high', 'low', 'close',
'volume'])

        base_date = datetime(2023, 1, 1)
        for _ in range(1000):
            stock = random.choice(stocks)
            date = base_date + timedelta(days=random.randint(0, 365))
            base_price = random.uniform(100, 1000)
            writer.writerow([
                date.strftime('%Y-%m-%d'),
                stock,
                round(base_price, 2),
                round(base_price * random.uniform(1, 1.1), 2),
                round(base_price * random.uniform(0.9, 1), 2),
                round(base_price * random.uniform(0.9, 1.1), 2),
                random.randint(100000, 10000000)
            ])

if __name__ == "__main__":
    generate_stock_data()
```

mapper.py:

```python
#!/usr/bin/env python3
import sys
import csv
from io import StringIO

for line in sys.stdin:
    try:
        if line.startswith('date'):
            continue

        csv_reader = csv.reader(StringIO(line.strip()))
        row = next(csv_reader)
        date, symbol, open_price, high, low, close, volume = row
        print(f'{symbol}\t{close},{volume}')
    except Exception as e:
        sys.stderr.write(f"Error: {str(e)}\n")
```

reducer.py:

```python
#!/usr/bin/env python3
import sys

current_symbol = None
total_volume = 0
price_sum = 0
count = 0

for line in sys.stdin:
    try:
        symbol, values = line.strip().split('\t')
        close, volume = values.split(',')

        if current_symbol != symbol:
            if current_symbol:
                avg_price = price_sum / count
                print(f'{current_symbol}\t{avg_price:.2f}\t{total_volume}')
            current_symbol = symbol
            total_volume = 0
            price_sum = 0
            count = 0

        price_sum += float(close)
        total_volume += int(volume)
        count += 1

    except Exception as e:
        sys.stderr.write(f"Error: {str(e)}\n")

if current_symbol and count > 0:
    avg_price = price_sum / count
    print(f'{current_symbol}\t{avg_price:.2f}\t{total_volume}')
```

run_analysis.sh:

```bash
#!/bin/bash
set -e

HADOOP_STREAMING_JAR=$(find $HADOOP_HOME -name "hadoop-streaming*.jar" | head
-1)
INPUT_DIR="/user/stock/input"
OUTPUT_DIR="/user/stock/output"

echo "Generating stock data..."
python3 generate_stock_data.py

echo "Making scripts executable..."
chmod +x mapper.py reducer.py

echo "Creating HDFS directories..."
hdfs dfs -mkdir -p $INPUT_DIR

echo "Uploading data to HDFS..."
hdfs dfs -put -f stock_data.csv $INPUT_DIR/

echo "Removing old output directory if exists..."
hdfs dfs -rm -r -f $OUTPUT_DIR

echo "Running MapReduce job..."
hadoop jar $HADOOP_STREAMING_JAR \
    -files mapper.py,reducer.py \
    -mapper "python3 mapper.py" \
    -reducer "python3 reducer.py" \
    -input $INPUT_DIR/* \
    -output $OUTPUT_DIR

echo "Analysis Results:"
hdfs dfs -cat $OUTPUT_DIR/part-*
```

Running the Analysis

1. Make files executable:

```
chmod +x generate_stock_data.py mapper.py reducer.py run_analysis.sh
```

2. Run the analysis:

```
./run_analysis.sh
```

Monitoring and Management

Web Interfaces

- NameNode: http://hadoop-namenode:9870
- YARN Resource Manager: http://hadoop-namenode:8088
- MapReduce Job History: http://hadoop-namenode:19888

Common Commands

```
# Check HDFS space usage
hdfs dfs -du -h /

# Check running applications
yarn application -list

# Kill a running application
yarn application -kill application_id

# Check logs
yarn logs -applicationId application_id
```

Troubleshooting

Common Issues and Solutions

1. Permission Issues

   ```
   # Fix HDFS permissions
   hdfs dfs -chmod -R 755 /user/directory
   hdfs dfs -chown -R user:group /user/directory
   ```

2. MapReduce Job Failures

   ```
   # Check job logs
   yarn logs -applicationId application_id > job_logs.txt
   ```

Check specific container logs

yarn logs -applicationId application_id -containerId container_id

```
3. HDFS Issues
```bash
# Run filesystem check
hdfs fsck /

# Recovery mode if needed
hdfs dfsadmin -safemode enter
hdfs dfsadmin -safemode leave
```

Best Practices

1. Data Management

- Use appropriate compression for data files
- Organize data in hierarchical directories
- Regular cleanup of temporary files

2. Job Optimization

- Use appropriate input split sizes
- Monitor resource usage
- Implement proper error handling in mapper/reducer

3. Security

- Set appropriate file permissions
- Use secure authentication when needed
- Regular security audits

4. Maintenance

- Regular backup of NameNode metadata
- Monitor disk space usage
- Keep system and Hadoop versions updated

Additional Resources

1. Official Documentation:
```

- Apache Hadoop Documentation
- HDFS Commands Guide

2. Monitoring Tools:

- Ganglia
- Grafana
- Prometheus

3. Development Tools:

- Eclipse Plugin for Hadoop
- PyCharm with Python Hadoop support