# Review

박성우

POSTECH

Fall 2023

# Test environment

- Set maximum heap size to 16GB
  _JAVA_OPTIONS="-Xmx16g"
- Input data:
  - Use ASCII format
  - 320,000 records per file
  - small dataset: 2 files per node (64MB per node)
  - big dataset: 10 files per node (320MB per node)
  - large dataset: 100 files per node (3.2GB per node)
- # nodes (with 1 master node)
  - small dataset: 4 worker nodes ⍰ total 256MB input data
  - big dataset: 9 worker nodes ⍰ total 2.88GB input data
  - large dataset: 9 worker nodes ⍰ total 28.8GB input data
- Execution time measurement
  - start: all worker JVM processes started
  - end: master JVM process finished
- Result verification
  - intra-file sort: using valsort
  - inter-file, inter-machine sort: manually comparing head/tail of each file

# Team Red

- Command:
  - bin/master 4
  - bin/worker 2.2.2.142:9999 -I ~/dataset/small -O ~/out/small/red -ascii
- Small dataset
  - execution time: 16s
- Big dataset
  - execution time: 123s
- Large dataset
  - execution time: 7334s
- Correctness verification

| | |
|---|---|
| 6. Does the master print a sequence of workers? | Yes |
| 7. Is the output sorted in each worker? | Yes |
| 8. # of records in the input == # of records in the output? | Yes |

# Team Green

- Command:
  - build/master 4
  - build/worker 2.2.2.142:50051 -I ~/dataset/small -O ~/out/small/green
- Small dataset
  - execution time: 169s
- Big dataset
  - execution time: 2743s
- Correctness verification

| | |
|---|---|
| 6. Does the master print a sequence of workers? | Yes |
| 7. Is the output sorted in each worker? | Yes |
| 8. # of records in the input == # of records in the output? | Yes |

# Team Blue

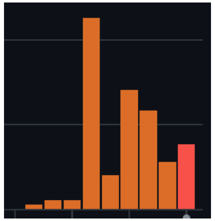
- Command:
  - java -jar master.jar 4
  - java -jar worker.jar 2.2.2.142:30962 -I ~/bin.dataset/small -O ~/out/small/blue
- Small dataset
  - execution time: 14s
- Big dataset
  - execution time: 87s
- Large dataset
  - execution time: 1084s
- Correctness verification

| | |
|---|---|
| 6. Does the master print a sequence of workers? | Yes |
| 7. Is the output sorted in each worker? | Yes |
| 8. # of records in the input == # of records in the output? | Yes |

# Comments from tests

- Red
  - needs an option for ASCII input (???)
- Green
  - works only on ASCII input (???)
  - deletes input files and creates new files in the input directory (???)
    - ⍰ The TA had a hard time to deal with this problem.
- Blue
  - works okay on both ASCII and binary input (without requiring an extra option)
  - Output file names are wrong (e.g., partition??????????).
  - creates tmp files in the output directories
  - Both master and workers use the same port number which is hard-coded.

# Red (37 files, 393 / 1513 lines, 142 commits)



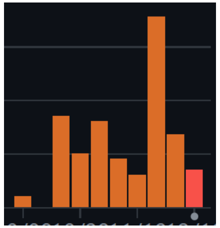
- Excellent documentation (design, API, usage)
- **Uses java.net.***
- Uses logging (org.apache.logging.log4j.scala.Logging)
- Includes test code and shell scripts
- **Uses assert{}/require{}**

```
assert(partitionSortedFiles.nonEmpty, "No partitions found after sorting")
```

- Frequent code refactoring, **high-quality code**

```
/**
   * Sends sampled keys from the specified folder.
   *
   * @param folderPath      Path to the folder containing data files.
   * @param input_data_type Input data type ("byte" or "ascii").
   * @return                List of sampled keys.
   */
 def sendSamples(folderPath: String, input_data_type: String): List[Key] = {
```

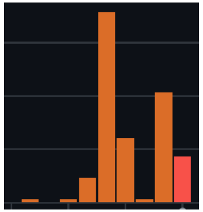
# Green (26 files, 1 / 1440 lines, 294 commit)



- Includes test code
- **Automated testing using Docker and GitHub Action (???)**
- Uses logging (org.apache.logging.log4j.scala.Logging)
- Uses Future and Promise, but also uses Await.result
- Uses assert{} mostly in test code
- Uses implicit class (???)

```
val promise = Promise[SampleReply]
Future {
  try { ...
    promise.success(SampleReply(sampledKeys))
  } catch {
    case e: Exception =>
      promise.failure(e)
  }
}(executionContext)
promise.future
```

# Blue (11 files, 34 / 695 lines, 190 commits)



- Uses Protobuf
  - **simple interface to Master and Worker**
- Uses logging (com.typesafe.scalalogging.Logger)
- Extensive use of Future and Promise (???)
- Includes test code
- **Uses (a variant of) assert{}**
- **Simple design (???)**