Distributed Sorting

DEVAUCHELLE Alex
49004567

RENAUD Thomas
49004578

Design

Details - Communication Libraries

- Communication Method: Basic Socket and Object Stream

- Reasoning: We opted for a simple and effective communication method using basic socket and object stream for our Distributed Sorting project in Scala.

- Advantages:
    - Simplicity: Straightforward implementation and easy to understand.
    - Flexibility: Allows for custom serialization and deserialization of objects.
    - Language Agnostic: Can communicate between different programming languages if needed.

Communication Methods Comparison: gRPC vs. Socket and Stream

gRPC

Advantages:

- **Protocol Buffers**: Efficient serialization using Protocol Buffers for data interchange.
- **Bidirectional Streaming**: Supports streaming requests and responses simultaneously.
- **Interceptors**: Supports middleware for handling cross-cutting concerns.

Disadvantages:

- **Complexity**: May introduce complexity in understanding due to its feature-rich nature.
- **Learning Curve**: Developers might need time to grasp the concepts and usage.
- **Heavy Dependencies**: Requires additional libraries and tools.

Basic Socket with Object Stream

Advantages:

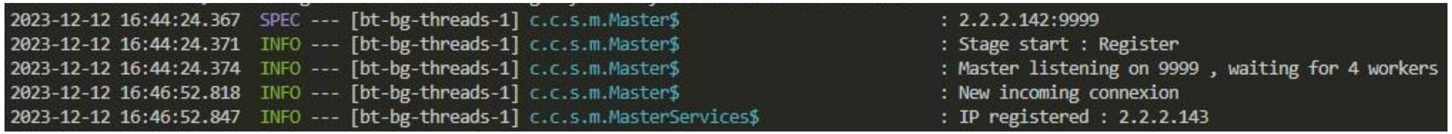
- **Simplicity**: Straightforward implementation, especially for smaller projects.
- **Custom Serialization**: Allows for custom serialization/deserialization methods.
- **Platform Independence**: Works well across different platforms.

Disadvantages:

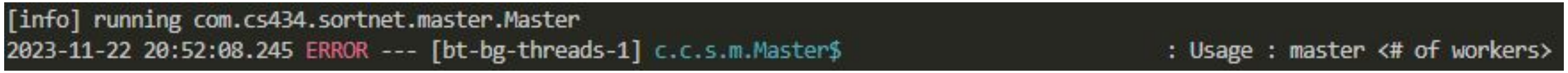
- **Limited Features**: Lacks some advanced features like built-in authentication, load balancing, and middleware.
- **Potential for Bottlenecks**: Depending on implementation, may face performance challenges with large-scale data.

Details - Logging in the Project

- Logging Library Used: Log4j
- Implementation Steps:
  - Add library dependency in build.sbt
  - Create a log4j2.xml configuration file. Define loggers, appenders, and layout patterns
  - Implementation in Code : Import Logging, Extend my class or object with Logging and use logger.{info;debug,error,spec}
- Exemple :



```
2023-12-12 16:44:24.367  SPEC --- [bt-bg-threads-1] c.c.s.m.Master$                    : 2.2.2.142:9999
2023-12-12 16:44:24.371  INFO --- [bt-bg-threads-1] c.c.s.m.Master$                    : Stage start : Register
2023-12-12 16:44:24.374  INFO --- [bt-bg-threads-1] c.c.s.m.Master$                    : Master listening on 9999 , waiting for 4 workers
2023-12-12 16:46:52.818  INFO --- [bt-bg-threads-1] c.c.s.m.Master$                    : New incoming connexion
2023-12-12 16:46:52.847  INFO --- [bt-bg-threads-1] c.c.s.m.MasterServices$            : IP registered : 2.2.2.143
```



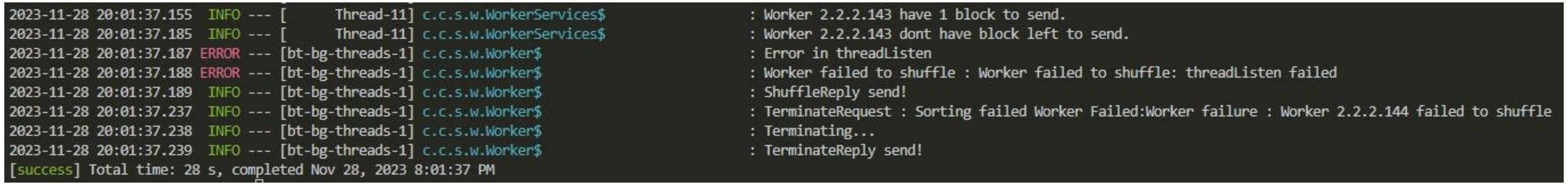
```
[info] running com.cs434.sortnet.master.Master
2023-11-22 20:52:08.245 ERROR --- [bt-bg-threads-1] c.c.s.m.Master$                : Usage : master <# of workers>
```

Error handling

Robust project :

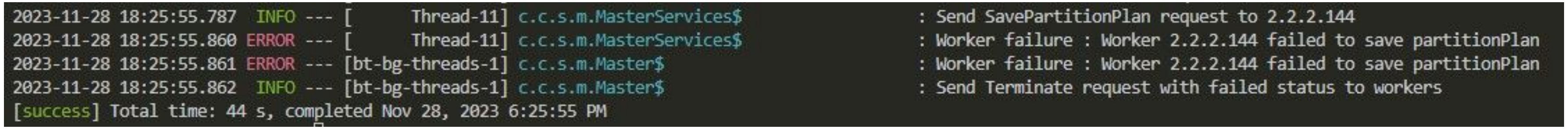
- Handle worker crash by warning the other to finish the program nicely

Worker failure:



```
2023-11-28 20:01:37.155  INFO --- [      Thread-11] c.c.s.w.WorkerServices$                  : Worker 2.2.2.143 have 1 block to send.
2023-11-28 20:01:37.185  INFO --- [      Thread-11] c.c.s.w.WorkerServices$                  : Worker 2.2.2.143 dont have block left to send.
2023-11-28 20:01:37.187 ERROR --- [bt-bg-threads-1] c.c.s.w.Worker$                          : Error in threadListen
2023-11-28 20:01:37.188 ERROR --- [bt-bg-threads-1] c.c.s.w.Worker$                          : Worker failed to shuffle : Worker failed to shuffle: threadListen failed
2023-11-28 20:01:37.189  INFO --- [bt-bg-threads-1] c.c.s.w.Worker$                          : ShuffleReply send!
2023-11-28 20:01:37.237  INFO --- [bt-bg-threads-1] c.c.s.w.Worker$                          : TerminateRequest : Sorting failed Worker Failed:Worker failure : Worker 2.2.2.144 failed to shuffle
2023-11-28 20:01:37.238  INFO --- [bt-bg-threads-1] c.c.s.w.Worker$                          : Terminating...
2023-11-28 20:01:37.239  INFO --- [bt-bg-threads-1] c.c.s.w.Worker$                          : TerminateReply send!
[success] Total time: 28 s, completed Nov 28, 2023 8:01:37 PM
```

Master handle it:



```
2023-11-28 18:25:55.787  INFO --- [      Thread-11] c.c.s.m.MasterServices$                  : Send SavePartitionPlan request to 2.2.2.144
2023-11-28 18:25:55.860 ERROR --- [      Thread-11] c.c.s.m.MasterServices$                  : Worker failure : Worker 2.2.2.144 failed to save partitionPlan
2023-11-28 18:25:55.861 ERROR --- [bt-bg-threads-1] c.c.s.m.Master$                          : Worker failure : Worker 2.2.2.144 failed to save partitionPlan
2023-11-28 18:25:55.862  INFO --- [bt-bg-threads-1] c.c.s.m.Master$                          : Send Terminate request with failed status to workers
[success] Total time: 44 s, completed Nov 28, 2023 6:25:55 PM
```

Developing environment

Make our life simple :

- Developing directly on production environment, and using Git Project
- Shell scripts to create, delete records and ensure the sorting of output files

Overall Design

Data structure

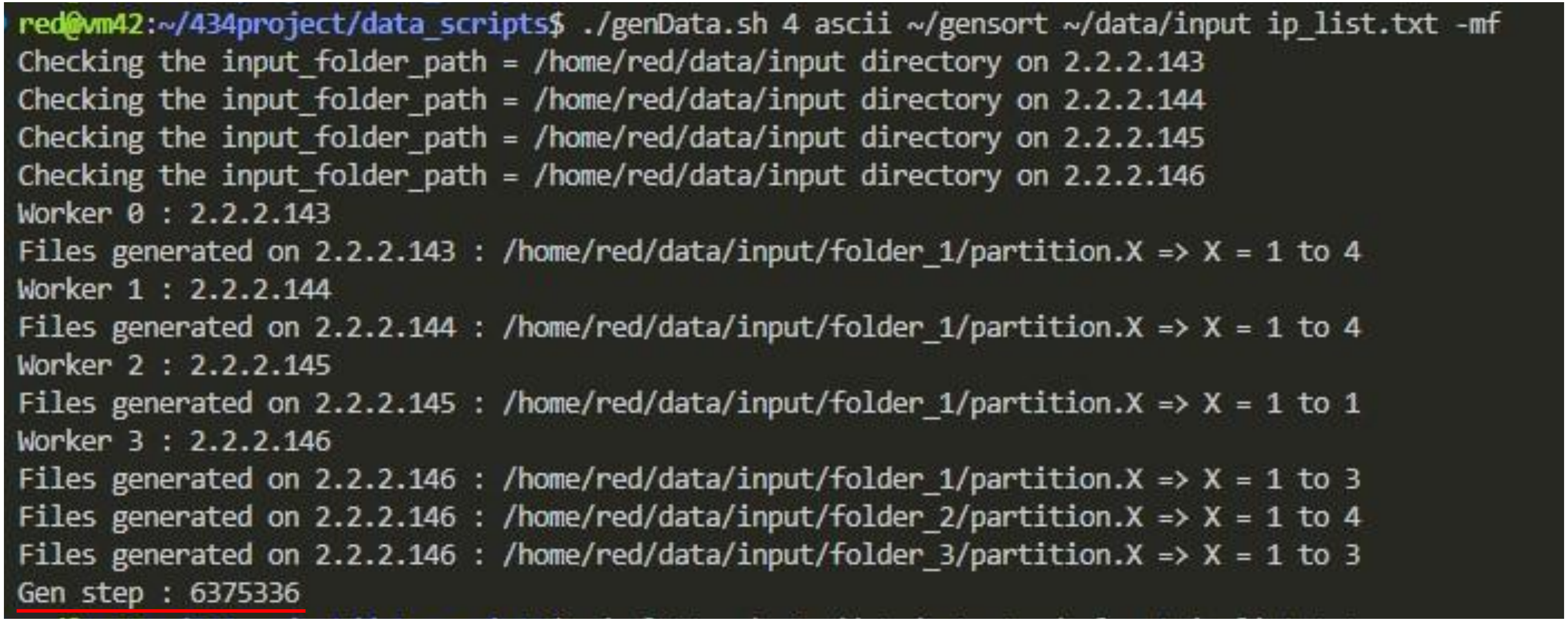
Network data structure

WorkFlow

Experiment

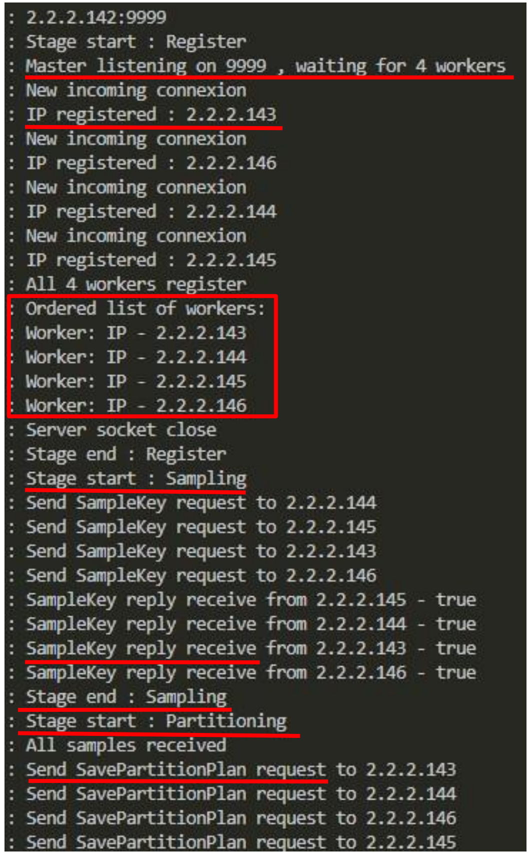
Live Demo

Data Generation



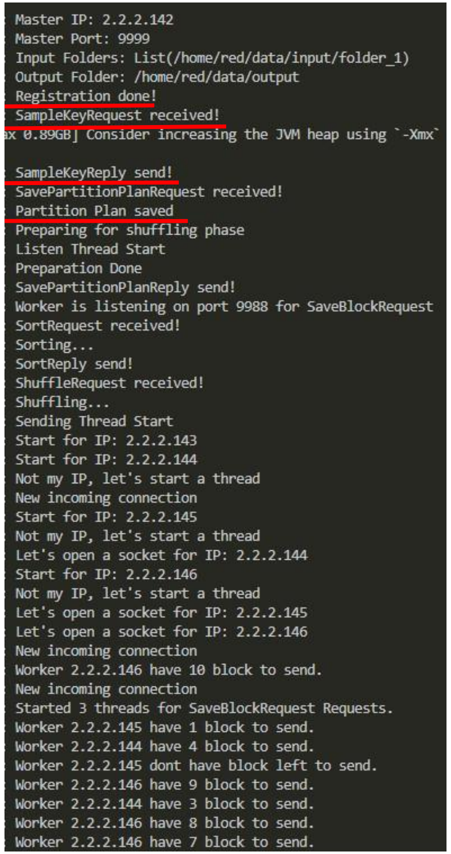
```
red@vm42:~/434project/data_scripts$ ./genData.sh 4 ascii ~/gensort ~/data/input ip_list.txt -mf
Checking the input_folder_path = /home/red/data/input directory on 2.2.2.143
Checking the input_folder_path = /home/red/data/input directory on 2.2.2.144
Checking the input_folder_path = /home/red/data/input directory on 2.2.2.145
Checking the input_folder_path = /home/red/data/input directory on 2.2.2.146
Worker 0 : 2.2.2.143
Files generated on 2.2.2.143 : /home/red/data/input/folder_1/partition.X => X = 1 to 4
Worker 1 : 2.2.2.144
Files generated on 2.2.2.144 : /home/red/data/input/folder_1/partition.X => X = 1 to 4
Worker 2 : 2.2.2.145
Files generated on 2.2.2.145 : /home/red/data/input/folder_1/partition.X => X = 1 to 1
Worker 3 : 2.2.2.146
Files generated on 2.2.2.146 : /home/red/data/input/folder_1/partition.X => X = 1 to 3
Files generated on 2.2.2.146 : /home/red/data/input/folder_2/partition.X => X = 1 to 4
Files generated on 2.2.2.146 : /home/red/data/input/folder_3/partition.X => X = 1 to 3
Gen step : 6375336
```

Master



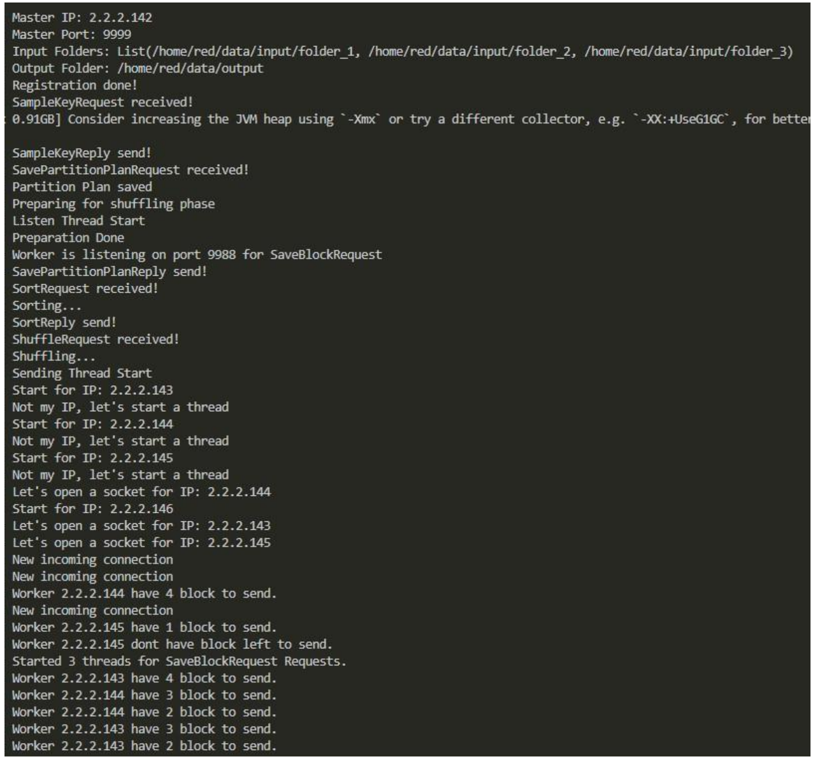
```
: 2.2.2.142:9999
: Stage start : Register
: Master listening on 9999 , waiting for 4 workers
: New incoming connexion
: IP registered : 2.2.2.143
: New incoming connexion
: IP registered : 2.2.2.146
: New incoming connexion
: IP registered : 2.2.2.144
: New incoming connexion
: IP registered : 2.2.2.145
: All 4 workers register
: Ordered list of workers:
: Worker: IP - 2.2.2.143
: Worker: IP - 2.2.2.144
: Worker: IP - 2.2.2.145
: Worker: IP - 2.2.2.146
: Server socket close
: Stage end : Register
: Stage start : Sampling
: Send SampleKey request to 2.2.2.144
: Send SampleKey request to 2.2.2.145
: Send SampleKey request to 2.2.2.143
: Send SampleKey request to 2.2.2.146
: SampleKey reply receive from 2.2.2.145 - true
: SampleKey reply receive from 2.2.2.144 - true
: SampleKey reply receive from 2.2.2.143 - true
: SampleKey reply receive from 2.2.2.146 - true
: Stage end : Sampling
: Stage start : Partitioning
: All samples received
: Send SavePartitionPlan request to 2.2.2.143
: Send SavePartitionPlan request to 2.2.2.144
: Send SavePartitionPlan request to 2.2.2.146
: Send SavePartitionPlan request to 2.2.2.145
```

Worker 43 …



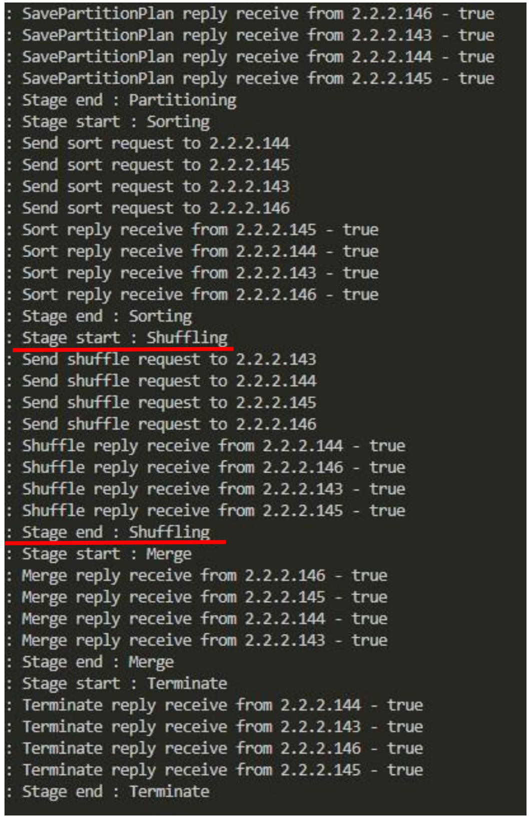
```
Master IP: 2.2.2.142
Master Port: 9999
Input Folders: List(/home/red/data/input/folder_1)
Output Folder: /home/red/data/output
Registration done!
SampleKeyRequest received!
x 0.89GB] Consider increasing the JVM heap using `-Xmx`

SampleKeyReply send!
SavePartitionPlanRequest received!
Partition Plan saved
Preparing for shuffling phase
Listen Thread Start
Preparation Done
SavePartitionPlanReply send!
Worker is listening on port 9988 for SaveBlockRequest
SortRequest received!
Sorting...
SortReply send!
ShuffleRequest received!
Shuffling...
Sending Thread Start
Start for IP: 2.2.2.143
Start for IP: 2.2.2.144
Not my IP, let's start a thread
New incoming connection
Start for IP: 2.2.2.145
Not my IP, let's start a thread
Let's open a socket for IP: 2.2.2.144
Start for IP: 2.2.2.146
Not my IP, let's start a thread
Let's open a socket for IP: 2.2.2.145
Let's open a socket for IP: 2.2.2.146
New incoming connection
Worker 2.2.2.146 have 10 block to send.
New incoming connection
Started 3 threads for SaveBlockRequest Requests.
Worker 2.2.2.145 have 1 block to send.
Worker 2.2.2.144 have 4 block to send.
Worker 2.2.2.145 dont have block left to send.
Worker 2.2.2.146 have 9 block to send.
Worker 2.2.2.144 have 3 block to send.
Worker 2.2.2.146 have 8 block to send.
Worker 2.2.2.146 have 7 block to send.
```

Worker 46



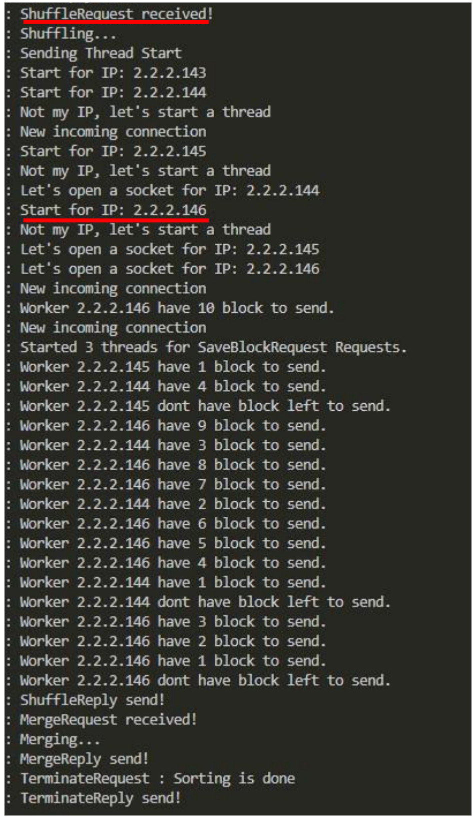
```
Master IP: 2.2.2.142
Master Port: 9999
Input Folders: List(/home/red/data/input/folder_1, /home/red/data/input/folder_2, /home/red/data/input/folder_3)
Output Folder: /home/red/data/output
Registration done!
SampleKeyRequest received!
0.91GB] Consider increasing the JVM heap using `-Xmx` or try a different collector, e.g. `-XX:+UseG1GC`, for better

SampleKeyReply send!
SavePartitionPlanRequest received!
Partition Plan saved
Preparing for shuffling phase
Listen Thread Start
Preparation Done
Worker is listening on port 9988 for SaveBlockRequest
SavePartitionPlanReply send!
SortRequest received!
Sorting...
SortReply send!
ShuffleRequest received!
Shuffling...
Sending Thread Start
Start for IP: 2.2.2.143
Not my IP, let's start a thread
Start for IP: 2.2.2.144
Not my IP, let's start a thread
Start for IP: 2.2.2.145
Not my IP, let's start a thread
Let's open a socket for IP: 2.2.2.144
Start for IP: 2.2.2.146
Let's open a socket for IP: 2.2.2.143
Let's open a socket for IP: 2.2.2.145
New incoming connection
New incoming connection
Worker 2.2.2.144 have 4 block to send.
New incoming connection
Worker 2.2.2.145 have 1 block to send.
Worker 2.2.2.145 dont have block left to send.
Started 3 threads for SaveBlockRequest Requests.
Worker 2.2.2.143 have 4 block to send.
Worker 2.2.2.144 have 3 block to send.
Worker 2.2.2.144 have 2 block to send.
Worker 2.2.2.143 have 3 block to send.
Worker 2.2.2.143 have 2 block to send.
```

Master



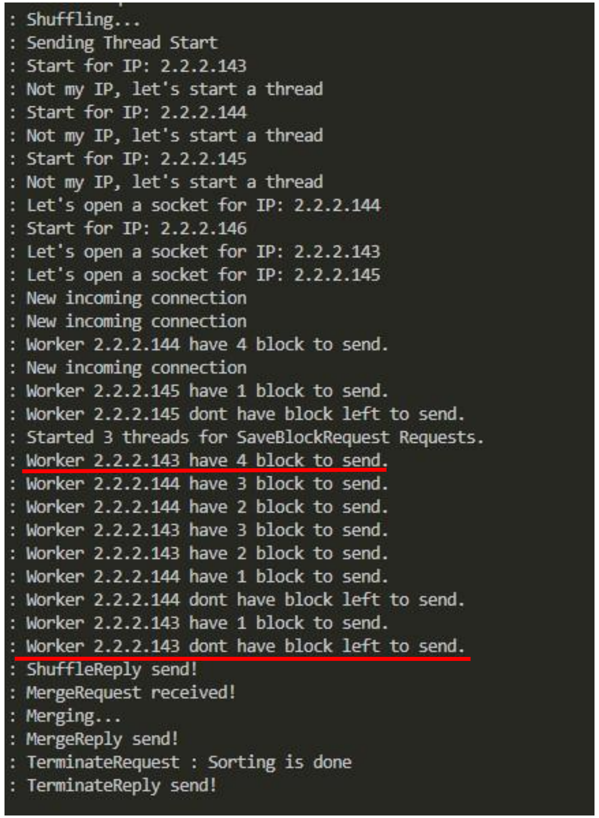
```
: SavePartitionPlan reply receive from 2.2.2.146 - true
: SavePartitionPlan reply receive from 2.2.2.143 - true
: SavePartitionPlan reply receive from 2.2.2.144 - true
: SavePartitionPlan reply receive from 2.2.2.145 - true
: Stage end : Partitioning
: Stage start : Sorting
: Send sort request to 2.2.2.144
: Send sort request to 2.2.2.145
: Send sort request to 2.2.2.143
: Send sort request to 2.2.2.146
: Sort reply receive from 2.2.2.145 - true
: Sort reply receive from 2.2.2.144 - true
: Sort reply receive from 2.2.2.143 - true
: Sort reply receive from 2.2.2.146 - true
: Stage end : Sorting
: Stage start : Shuffling
: Send shuffle request to 2.2.2.143
: Send shuffle request to 2.2.2.144
: Send shuffle request to 2.2.2.145
: Send shuffle request to 2.2.2.146
: Shuffle reply receive from 2.2.2.144 - true
: Shuffle reply receive from 2.2.2.146 - true
: Shuffle reply receive from 2.2.2.143 - true
: Shuffle reply receive from 2.2.2.145 - true
: Stage end : Shuffling
: Stage start : Merge
: Merge reply receive from 2.2.2.146 - true
: Merge reply receive from 2.2.2.145 - true
: Merge reply receive from 2.2.2.144 - true
: Merge reply receive from 2.2.2.143 - true
: Stage end : Merge
: Stage start : Terminate
: Terminate reply receive from 2.2.2.144 - true
: Terminate reply receive from 2.2.2.143 - true
: Terminate reply receive from 2.2.2.146 - true
: Terminate reply receive from 2.2.2.145 - true
: Stage end : Terminate
```

Worker 43 …



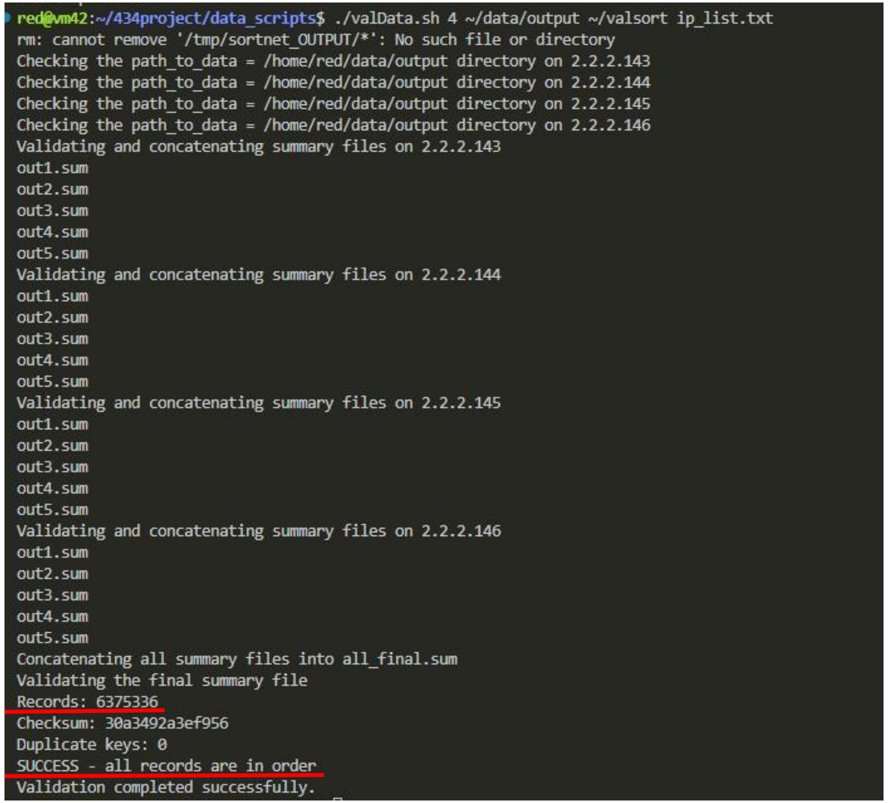
```
: ShuffleRequest received!
: Shuffling...
: Sending Thread Start
: Start for IP: 2.2.2.143
: Start for IP: 2.2.2.144
: Not my IP, let's start a thread
: New incoming connection
: Start for IP: 2.2.2.145
: Not my IP, let's start a thread
: Let's open a socket for IP: 2.2.2.144
: Start for IP: 2.2.2.146
: Not my IP, let's start a thread
: Let's open a socket for IP: 2.2.2.145
: Let's open a socket for IP: 2.2.2.146
: New incoming connection
: Worker 2.2.2.146 have 10 block to send.
: New incoming connection
: Started 3 threads for SaveBlockRequest Requests.
: Worker 2.2.2.145 have 1 block to send.
: Worker 2.2.2.144 have 4 block to send.
: Worker 2.2.2.145 dont have block left to send.
: Worker 2.2.2.146 have 9 block to send.
: Worker 2.2.2.144 have 3 block to send.
: Worker 2.2.2.146 have 8 block to send.
: Worker 2.2.2.146 have 7 block to send.
: Worker 2.2.2.144 have 2 block to send.
: Worker 2.2.2.146 have 6 block to send.
: Worker 2.2.2.146 have 5 block to send.
: Worker 2.2.2.146 have 4 block to send.
: Worker 2.2.2.144 have 1 block to send.
: Worker 2.2.2.144 dont have block left to send.
: Worker 2.2.2.146 have 3 block to send.
: Worker 2.2.2.146 have 2 block to send.
: Worker 2.2.2.146 have 1 block to send.
: Worker 2.2.2.146 dont have block left to send.
: ShuffleReply send!
: MergeRequest received!
: Merging...
: MergeReply send!
: TerminateRequest : Sorting is done
: TerminateReply send!
```

Worker 46



```
: Shuffling...
: Sending Thread Start
: Start for IP: 2.2.2.143
: Not my IP, let's start a thread
: Start for IP: 2.2.2.144
: Not my IP, let's start a thread
: Start for IP: 2.2.2.145
: Not my IP, let's start a thread
: Let's open a socket for IP: 2.2.2.144
: Start for IP: 2.2.2.146
: Let's open a socket for IP: 2.2.2.143
: Let's open a socket for IP: 2.2.2.145
: New incoming connection
: New incoming connection
: Worker 2.2.2.144 have 4 block to send.
: New incoming connection
: Worker 2.2.2.145 have 1 block to send.
: Worker 2.2.2.145 dont have block left to send.
: Started 3 threads for SaveBlockRequest Requests.
: Worker 2.2.2.143 have 4 block to send.
: Worker 2.2.2.144 have 3 block to send.
: Worker 2.2.2.144 have 2 block to send.
: Worker 2.2.2.143 have 3 block to send.
: Worker 2.2.2.143 have 2 block to send.
: Worker 2.2.2.144 have 1 block to send.
: Worker 2.2.2.144 dont have block left to send.
: Worker 2.2.2.143 have 1 block to send.
: Worker 2.2.2.143 dont have block left to send.
: ShuffleReply send!
: MergeRequest received!
: Merging...
: MergeReply send!
: TerminateRequest : Sorting is done
: TerminateReply send!
```

Data Validation



```
red@vm42:~/434project/data_scripts$ ./valData.sh 4 ~/data/output ~/valsort ip_list.txt
rm: cannot remove '/tmp/sortnet_OUTPUT/*': No such file or directory
Checking the path_to_data = /home/red/data/output directory on 2.2.2.143
Checking the path_to_data = /home/red/data/output directory on 2.2.2.144
Checking the path_to_data = /home/red/data/output directory on 2.2.2.145
Checking the path_to_data = /home/red/data/output directory on 2.2.2.146
Validating and concatenating summary files on 2.2.2.143
out1.sum
out2.sum
out3.sum
out4.sum
out5.sum
Validating and concatenating summary files on 2.2.2.144
out1.sum
out2.sum
out3.sum
out4.sum
out5.sum
Validating and concatenating summary files on 2.2.2.145
out1.sum
out2.sum
out3.sum
out4.sum
out5.sum
Validating and concatenating summary files on 2.2.2.146
out1.sum
out2.sum
out3.sum
out4.sum
out5.sum
Concatenating all summary files into all_final.sum
Validating the final summary file
Records: 6375336
Checksum: 30a3492a3ef956
Duplicate keys: 0
SUCCESS - all records are in order
Validation completed successfully.
```

Project management

Milestones

Design **Deadline : 10/16 - 11/12**

- ✓ Generate input data
- ✓ Overall flow chart design
- ✓ Choosing communication library, programming environment, logging system
- ✓ Choosing data structure for sampling, sorting, partitioning, shuffling and merging
- ✓ Documentation report : Data structure design
- ✓ Documentation report : Network data structure
- ✓ Sequence diagram on communication protocol

Milestones

Implementation 1/2 **Deadline : 11/13 - 11/26**

✓ Connection Master-Worker
✓ Setup logger (log4j)
✓ Communication sampling task
✓ Sample input, send and received from worker to master
✓ Key range attribution and broadcasted, partition plan
✓ Communication sorting task
✓ Sorting file
✓ Partition each file given partition plan
✓ Testing partial solution

Milestones

Implementation 2/2 **Deadline : 11/27 - 12/03**

- ✓ Communication shuffling task
- ✓ File transfer between worker
- ✓ Communication merging task
- ✓ Merging method
- ✓ Validating output result (valsort)
- ✓ Unit testing
- ✓ Error handling
- ✓ Testing overall solution

Responsibilities

Alex (surgeon) :

- Environment setup
- Design creation
- Documentation
- Master/Worker logic and implementation
- Shell Scripts
- Unit tests

Thomas :

- Weekly report
- Documentation
- Services implementation
- Unit tests

What do we learned from the project

Topics

- Concurrent programming

- Data Formats: Dealing with the serialization and deserialization of data efficiently in a distributed environment.

- Code Documentation: Writing clear and comprehensive documentation for the codebase.

- Team Collaboration: Working effectively in a team on a distributed project, coordinating efforts, and resolving issues collaboratively.

If we were to redo…

We would follow the same path, but :

- Deal with Unit Test and Error Handling earlier
- Advanced Unit Test
- More professional use of GitHub actions and branches to manage milestones

Thank you for listening