

SD final presentation

Team White

demo

Let's try simple fault tolerance live demo!

Master

```
$ java -cp 332project-assembly-0.1.0-SNAPSHOT.jar master.MasterApp  
<num of workers> [-d | --debug]
```

Worker

```
$ java -cp 332project-assembly-0.1.0-SNAPSHOT.jar worker.WorkerApp <master IP:port> -l  
<input dir1> [<input dir2> ...] -O <output dir> [-d | --debug]
```

result - correctness

```
white@vm01:~$ ls  
332project-assembly  
white@vm01:~$ cd ou  
white@vm01:~/output$  
6321900 partition.0  
6433600 partition.1  
6584700 partition.2  
5639900 partition.3  
6689000 partition.4  
6246400 partition.5  
7470000 partition.6  
5814200 partition.7  
6313800 partition.8  
6265300 partition.9  
white@vm01:~/output$  
  
white@vm03:~$ ls  
332project-assembly  
white@vm03:~$ cd output  
white@vm03:~/output$  
6397000 partition.20  
6129900 partition.21  
6588200 partition.22  
6634400 partition.23  
6632500 partition.24  
6509500 partition.25  
6448200 partition.26  
6513600 partition.27  
7314900 partition.28  
6089700 partition.29  
white@vm04:~/output$  
  
PS C:\Users\G-POS\Desktop\jong22\sd> python ./test.py  
1920000000  
  
white@vm01:~$ ./valsort output/partition.0  
Records: 63111  
Checksum: 7b0d2fd419ff  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.1  
Records: 62553  
Checksum: 7a5af00f07f4  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.2  
Records: 67601  
Checksum: 83ebac81f807  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.3  
Records: 58366  
Checksum: 721ecf1dbc1a  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.4  
Records: 68137  
Checksum: 84e2abc575d6  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.5  
Records: 62577  
Checksum: 7a62fc8df54d  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.6  
Records: 67590  
Checksum: 8484473fba61  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.7  
Records: 61781  
Checksum: 78489c3f433e  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.8  
Records: 66438  
Checksum: 82287bea59a6  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ ./valsort output/partition.9  
Records: 59786  
Checksum: 74c3bbd115a7  
Duplicate keys: 0  
SUCCESS - all records are in order  
white@vm01:~$ |
```

total output size is same to input

all output is sorted!

result - fault tolerance

	dead worker restarts	others
register	register again	keep working
sampling	sampling again	
partitioning	partitioning again	wait until the partitioning of the dead worker is complete, → restart from shuffling.
shuffling		
merging		

result - fault tolerance

```
INFO: Worker 0 finished Shuffling and is now WAITING. (2/3)
Dec 09, 2025 11:40:02 AM utils.Logging$ logInfo
INFO: Worker 2 finished Shuffling and is now WAITING. (3/3)
Dec 09, 2025 11:40:02 AM utils.Logging$ logInfo
INFO: All workers finished Shuffling. Moving to MERGING.
Dec 09, 2025 11:40:17 AM utils.Logging$ logInfo
INFO: Death Worker recovered: ID=2, IP=2.2.2.105
Dec 09, 2025 11:40:17 AM utils.Logging$ logInfo
INFO: Worker 2 port updated: 36969 -> 41191
Dec 09, 2025 11:40:17 AM utils.Logging$ logWarning
WARNING: Worker 2 recovered during Merging. Assigning PARTITIONING to recover lost data.
Dec 09, 2025 11:40:17 AM utils.Logging$ logInfo
INFO: Worker 2 recovered. Current Global Phase: Merging. Assigned State: Partitioning
Dec 09, 2025 11:40:21 AM utils.Logging$ logInfo
INFO: Worker 0 finished Merging and is now WAITING. (1/3)
Dec 09, 2025 11:40:21 AM utils.Logging$ logInfo
INFO: Worker 0 completed recovery PARTITIONING.
Dec 09, 2025 11:40:21 AM utils.Logging$ logInfo
INFO: Worker 1 finished Merging and is now WAITING. (2/3)
Dec 09, 2025 11:40:21 AM utils.Logging$ logInfo
INFO: Worker 1 completed recovery PARTITIONING.
Dec 09, 2025 11:40:33 AM utils.Logging$ logInfo
INFO: Worker 2 finished Partitioning and is now WAITING. (3/3)
Dec 09, 2025 11:40:33 AM utils.Logging$ logInfo
INFO: Worker 2 completed recovery PARTITIONING.
Dec 09, 2025 11:40:33 AM utils.Logging$ logInfo
INFO: Signalling FAILURE to all workers to restart SHUFFLING.
Dec 09, 2025 11:40:33 AM utils.Logging$ logInfo
INFO: All workers finished Partitioning. Moving to SHUFFLING.
Dec 09, 2025 11:40:43 AM utils.Logging$ logInfo
INFO: Worker 1 finished Shuffling and is now WAITING. (1/3)
Dec 09, 2025 11:40:45 AM utils.Logging$ logInfo
INFO: Worker 0 finished Shuffling and is now WAITING. (2/3)
Dec 09, 2025 11:40:47 AM utils.Logging$ logInfo
INFO: Worker 2 finished Shuffling and is now WAITING. (3/3)
Dec 09, 2025 11:40:47 AM utils.Logging$ logInfo
INFO: All workers finished Shuffling. Moving to MERGING.
```

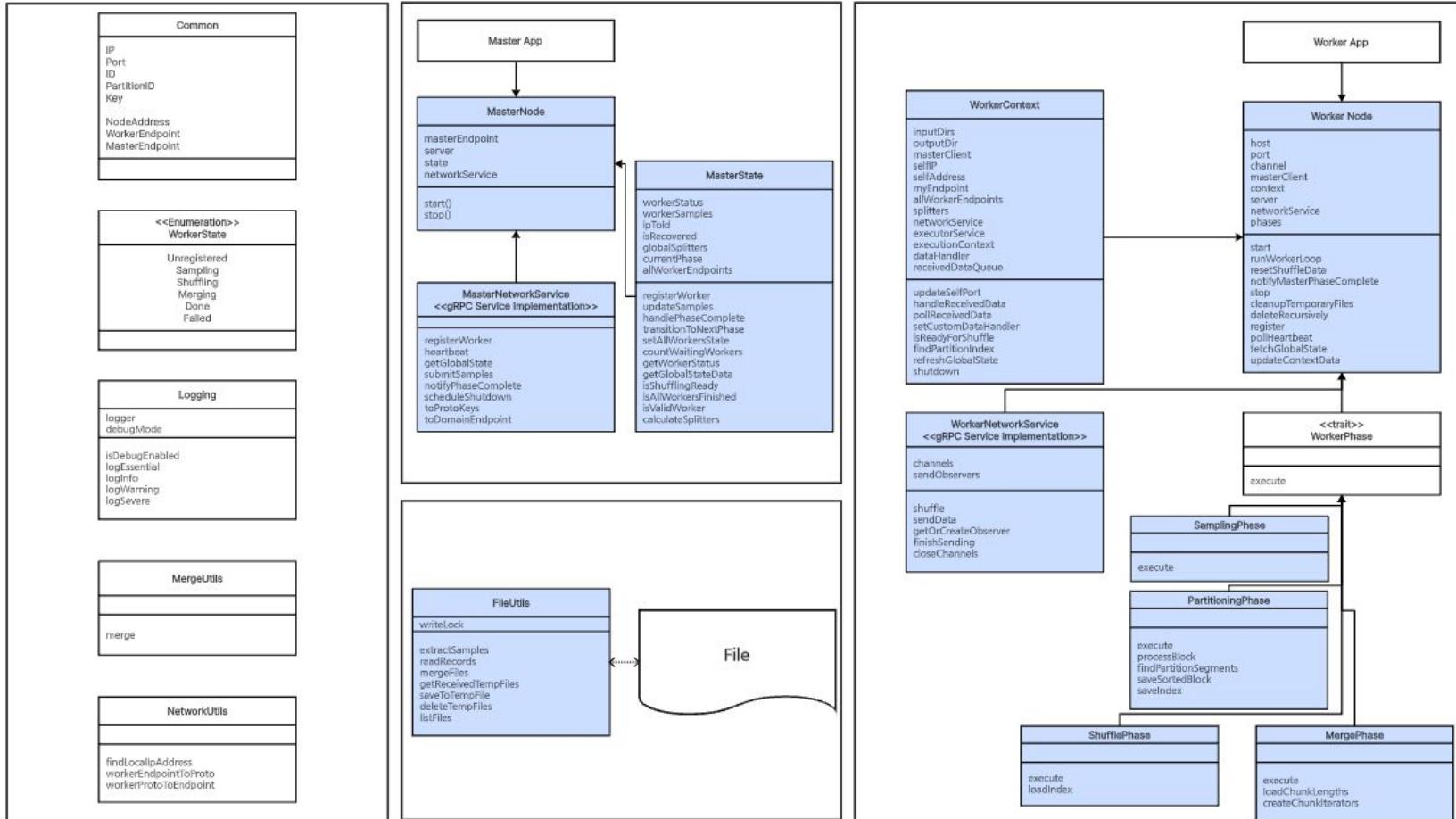
<- worker 2 Dead and Recover

<- all workers restart Shuffling again

Architecture

Master	Worker
Managing worker states transitioning between task phases calculating data distribution criteria (Splitter)	Reading local files sorting network transmission merging
Receives Heartbeat Register workers Issues commands	status reporting (Heartbeat) data transmission (Shuffling)

Class Diagram



Overall progress

Master

Implement worker register, state management, splitter calculation and transmission to all workers

→ small issue? - deleting temp file phase

Worker

Implement reading files, divide them by blocks, sorting blocks, network transmission (shuffling), merging.

→ implement issue?

Implement Issue?

when the merge phase begins, **all workers pause for 3 seconds** to wait for **late-arriving data**.

this is because the shuffling phase ends when all workers have **sent all their data. (not received all data)**

although we felt a bit uneasy about this method, **it worked well when we tested it.**

→ we attempted to terminate shuffling when each worker received EOS from all other workers, but failed.



Lessons learned from the project

1. **Proactive Time Management**
 - a. Start tasks early to buffer against unexpected bugs and issues.
2. **Clear Naming Conventions**
 - a. Use intuitive and descriptive variable names to enhance code readability.
3. **Decoupled Architecture for Collaboration**
 - a. Design a modular structure to allow team members to work independently and minimize conflicts.
4. **Importance of Logging**
 - a. Implement comprehensive logging to track program flow and facilitate easier debugging.