Software Design Methods Final Project

Progress Presentation

Purple Team
(신지용, 안강현, 박수민)

Index

- Progress
- Logistics
- MileStones
- TODO
- Design
- Development environment
- Generating Data

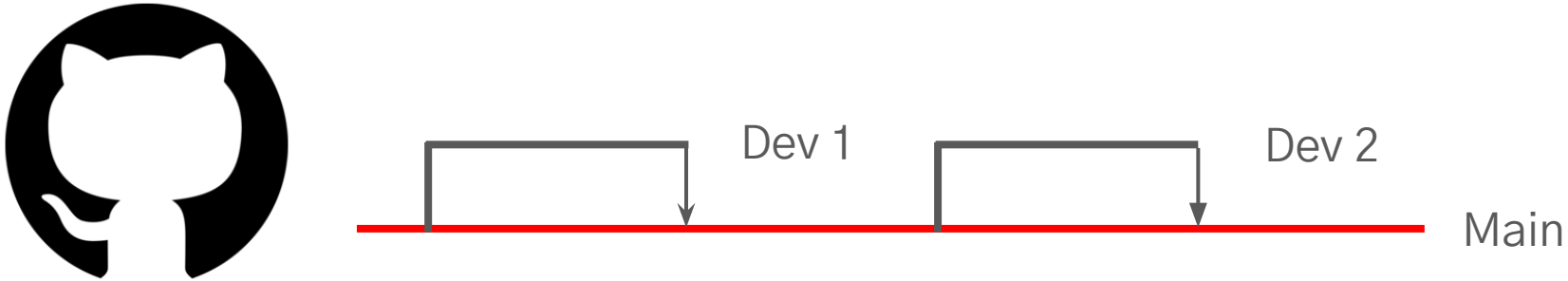
Progress

- Network Layer
  - Master–Worker TCP & Worker–Worker UDP communication
  - Reliable delivery with ACK/retry and exponential backoff
  - Worker restart recovery support

- Sorting Pipeline
  - Data distribution and statistical sampling
  - Range-based shuffle across workers
  - Local in-memory sorting (without key-value parsing)
  - K-way merge at Master

Logistics

- We held weekly online/offline meetings.
- If each member finishes one task, then picked up the next.



MileStones

MileStone #1

- Generate input data genSort
- Learn about
  - Distributed Sorting (External Sort)
  - Parallel programming
  - Network Libraries (gRPC, Netty)
- Standardize development environment (build tools, scala, etc.)
- Execute master
- Make workers connecting to master

Milestone #2 — Network Setup

- Finalize the choice of network library for Master–Worker communication
- Implement the simplest possible communication (e.g., “Hello World”) between Master server and Worker client
- Verify that multiple Workers can connect to the Master and that the Master recognizes all connected Workers

Milestone #3 — System Architecture setup

- Design the overall system structure based on the Master–Worker model
- Design the complete data processing pipeline (e.g., Sampling → Partitioning → Shuffle → Merge)
- Define a list of main classes and functions to be implemented
- Write skeleton code for the defined classes and functions

MileStones

Milestone #4 — Worker Implementation

- Local Processing Functions:
  - Implement Local Sorting: read file from local disk and sort it in memory
  - Implement Partitioning: divide sorted data according to partition keys
  - Implement Disk-based Merge: merge partition files received from other Workers
- Network Communication (Client Role):
  - Implement network code for sending sample data and status reports to the Master
  - Implement network code for Shuffle, transferring partition data between Workers

Milestone #5 — Master Implementation and System Integration

- Core Logic Functions:
  - Implement sorting of sample data collected from Workers and compute global pivot keys
  - Use the computed pivots to determine and assign key ranges for each Worker
- Network Communication (Server Role):
  - Implement the Master server code to handle Worker connection requests, data reception, and status updates
- System Integration:
  - Integrate all components so that the full sorting pipeline runs from start to finish under Master coordination

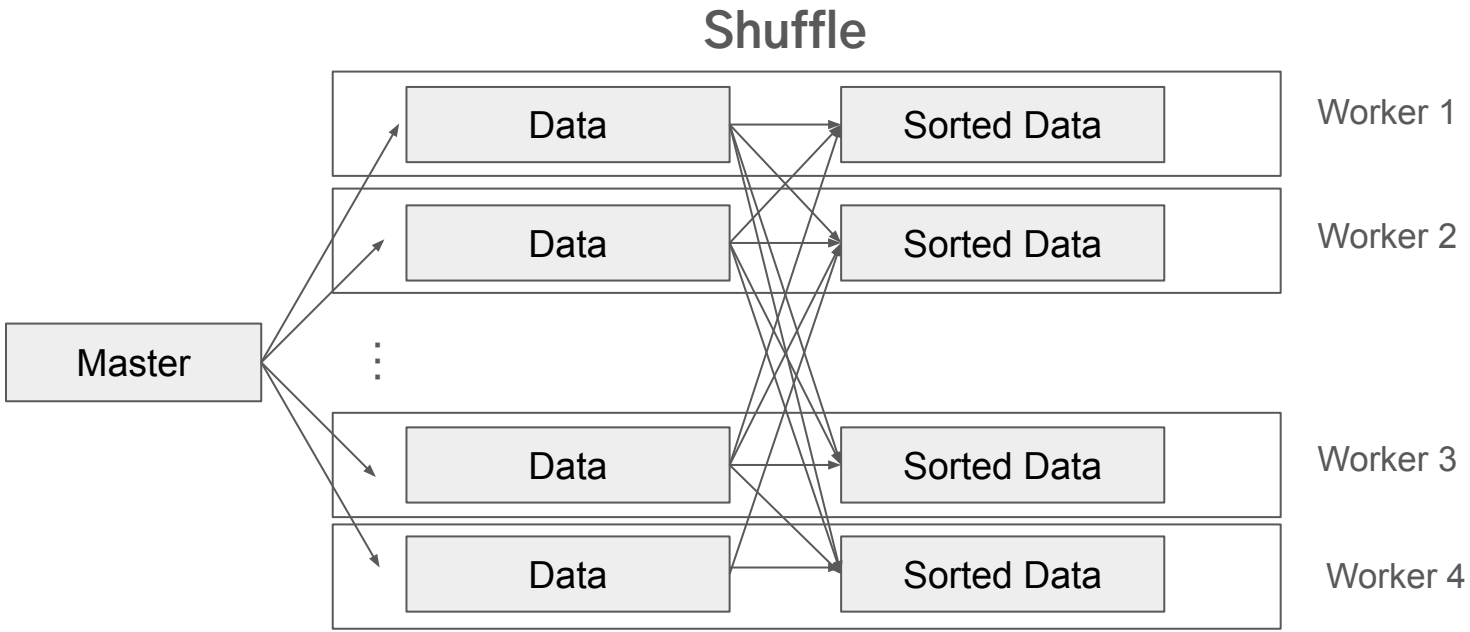
Milestone #6: Testing, Debugging, and Fault Tolerance Verification

- Run the full system and verify that the final output file is correctly sorted
- Fault-Tolerance Test: forcibly terminate one Worker during execution and confirm that the system successfully completes the job
- Fix bugs found during testing and stabilize the codebase

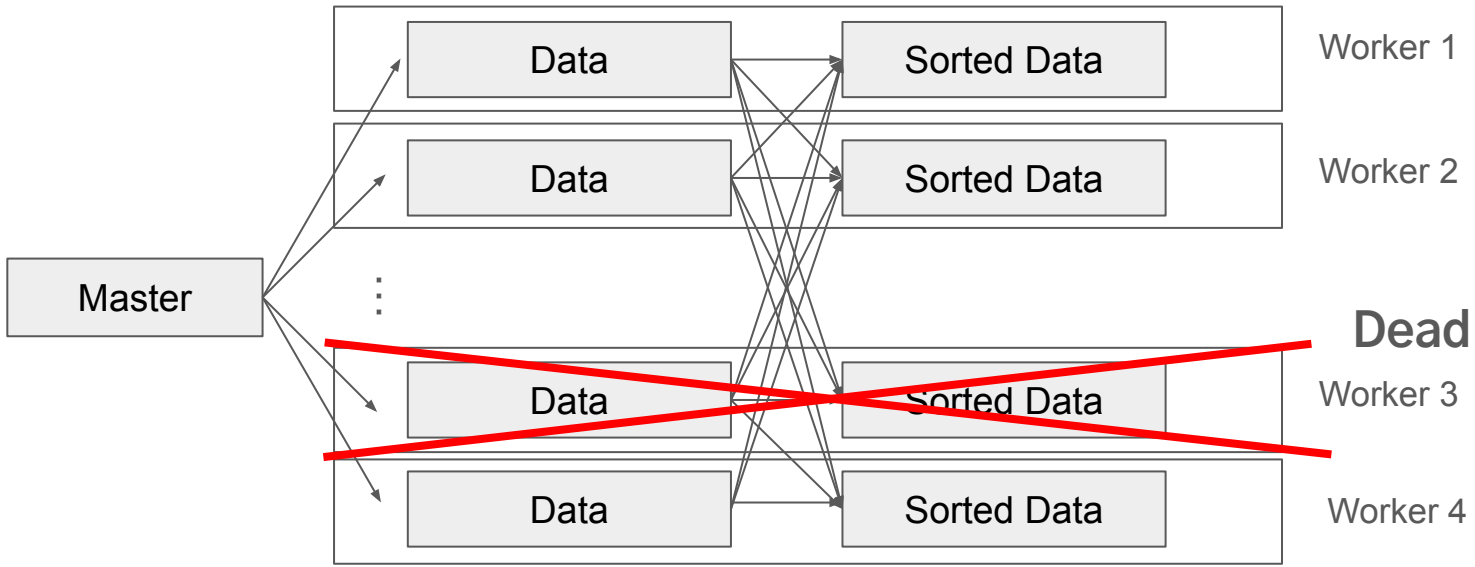
TODO

- Set logging library (Maybe log4j)
- Implement key-value parsing (ASCII & binary) and switchable mode flag
- Implement External Sort (handle limited memory)
- Support genSort format (process real datasets)
- Strengthen fault tolerance (checkpoints, restart recovery)
- Improve error handling (timeouts, exception recovery)
- Add test cases (integration tests, failure scenarios)

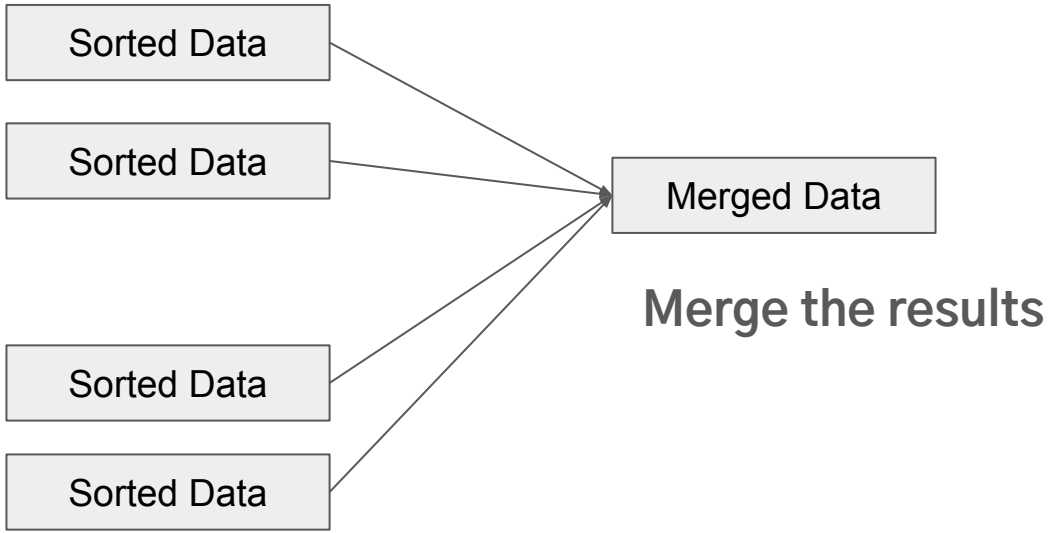
Design


Shuffle
Data
Sorted Data
Worker 1
Data
Sorted Data
Worker 2
Master
Data
Sorted Data
Worker 3
Data
Sorted Data
Worker 4

Design


Data
Sorted Data
Worker 1
Data
Sorted Data
Worker 2
Master
Dead
Data
Sorted Data
Worker 3
Data
Sorted Data
Worker 4

Design


Sorted Data
Sorted Data
Sorted Data
Sorted Data
Merged Data
Merge the results

Development Environment

- Scala: 2.13.0
- Sbt: 1.11.7
- Java 8
- Netty: 4.1.100

Dataset generation by gensort

- Create Data (ascii)

```
./gensort -a -b0    500 ascii_part0
./gensort -a -b500  500 ascii_part1
./gensort -a -b1000 500 ascii_part2
./gensort -a -b1500 500 ascii_part3
```

```
AsfAGHM5om  0000000000000000000000000000000000  00002222000022220000222200002222000022220000000001111
~sHd0jDv6X  0000000000000000000000000000000001  77779999444488885555CCCC777755555555BBBB666644446666
uI^EYm8s=|  0000000000000000000000000000000002  CCCCFFFF777799995555FFFF1111222299998888444 4DDDDFFFF
Q)JN)R9z-L  0000000000000000000000000000000003  FFFF11110000000000006666888 8BBBB33333333AAAA1111CCCC
o4FoBkqERn  0000000000000000000000000000000004  7777AAAABBBBBBBB22224444444499995555BBBB11118888DDDD
*}-Wz1;TD-  0000000000000000000000000000000005  AAAA88883333BBBB888888884444777722227777999900002222
0fssx}~[oB  0000000000000000000000000000000006  FFFF999977774444AAAA7777EEEEDDDDAAAAAAAA99998888BBBB
mz4VCN@a#"  0000000000000000000000000000000007  DDDDBBBB1111FFFF2222DDDDFFFFBBBBFFFF6666444477778888
my+=5r7(N|  0000000000000000000000000000000008  22226666CCCC66662222FFFF0000EEEE11118888444455559999
```

Q&A

Thank you for listening