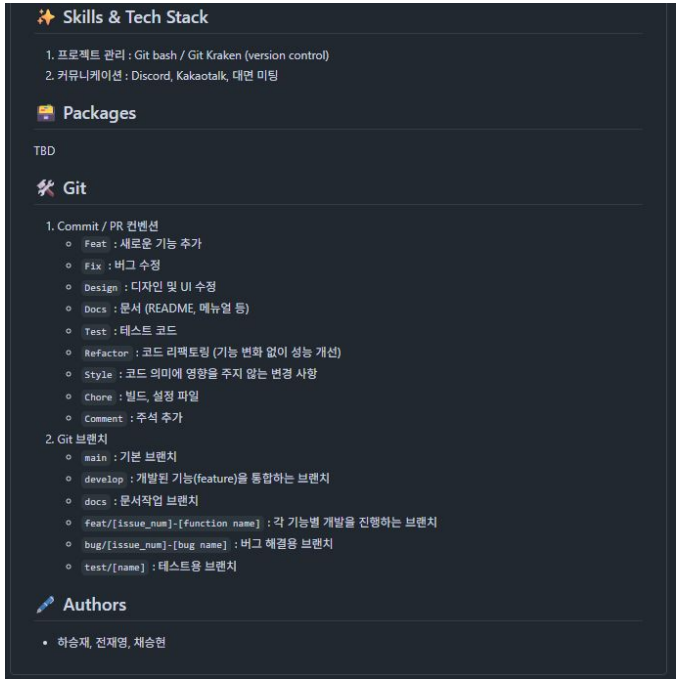
SD progress presentation

team white

Contents

1. Management



✨ Skills & Tech Stack

1. 프로젝트 관리 : Git bash / Git Kraken (version control)
2. 커뮤니케이션 : Discord, Kakaotalk, 대면 미팅

📦 Packages

TBD

🛠️ Git

1. Commit / PR 컨벤션
   - `Feat` : 새로운 기능 추가
   - `Fix` : 버그 수정
   - `Design` : 디자인 및 UI 수정
   - `Docs` : 문서 (README, 메뉴얼 등)
   - `Test` : 테스트 코드
   - `Refactor` : 코드 리팩토링 (기능 변화 없이 성능 개선)
   - `Style` : 코드 의미에 영향을 주지 않는 변경 사항
   - `Chore` : 빌드, 설정 파일
   - `Comment` : 주석 추가
2. Git 브랜치
   - `main` : 기본 브랜치
   - `develop` : 개발된 기능(feature)을 통합하는 브랜치
   - `docs` : 문서작업 브랜치
   - `feat/[issue_num]-[function name]` : 각 기능별 개발을 진행하는 브랜치
   - `bug/[issue_num]-[bug name]` : 버그 해결용 브랜치
   - `test/[name]` : 테스트용 브랜치

🖋️ Authors

- 하승재, 전재영, 채승현

git convention

github: version control system

\+ **gitkraken**, **fork**

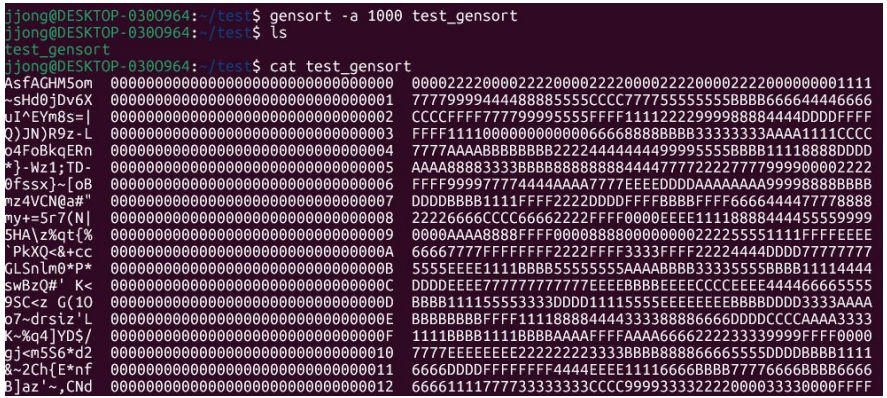
discord: save file, online meeting

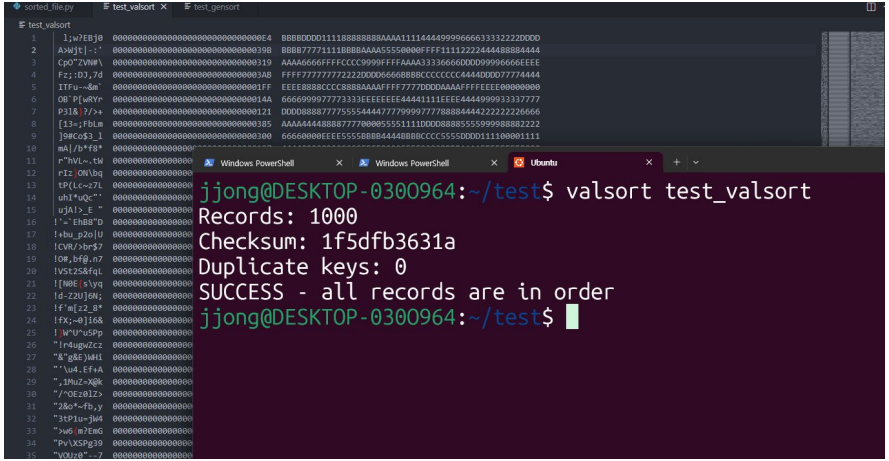
kakaotalk: simple communication

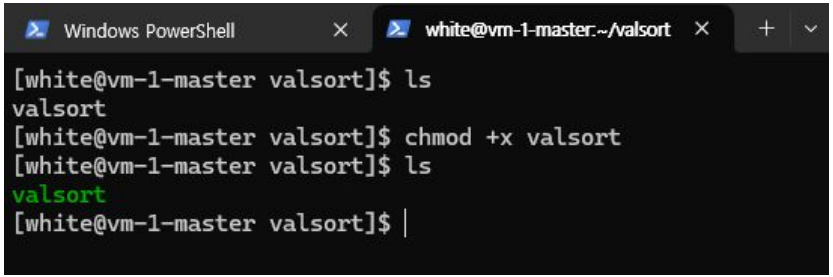
google drive: make ppt file, report

boardmix: make UML

2.a. sampling







gensort

valsort

2.b. Build Libraries Setting

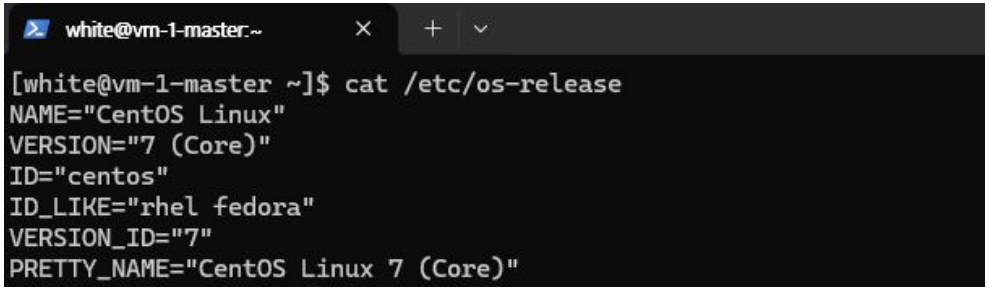
OS: window 11

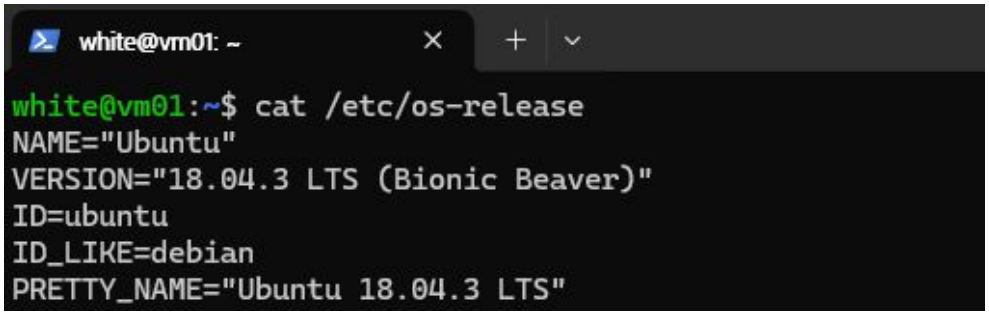
+ **WSL**(CentOS 7 or Ubuntu)

Java: jdk 8 (JDK 8u202)

Scala: scala 3.3.7

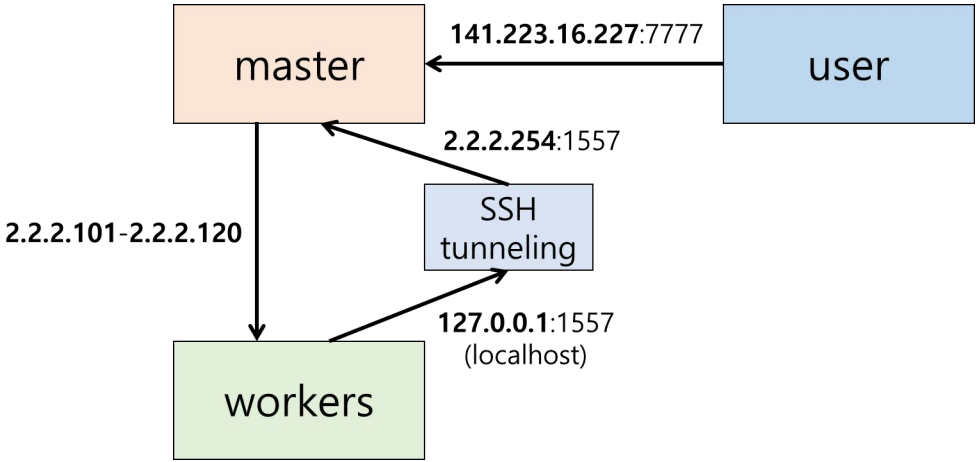
Libraries: Scalapb, netty, protobuf, gRPC



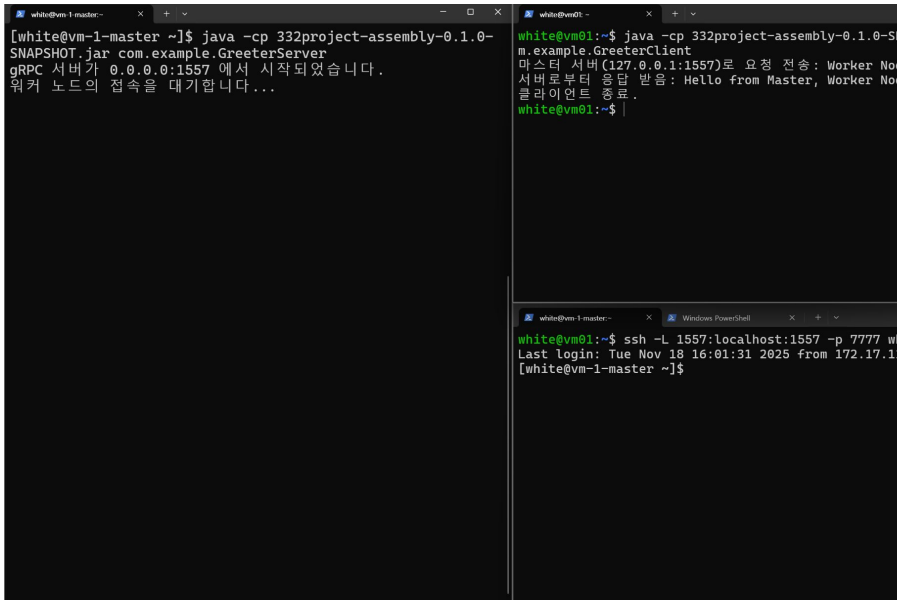


master uses CentOS, **worker** uses Ubuntu

2.c. server connection test



could be automated with JSch



connection succeeded

3.a. sample sort

step	to do
sampling	for each worker, sample random K data (ex. first 1000 of data).
partitioning	master collects and sorts all samples, then finds the pivots (N-1) to divide them into ranges corresponding to the number of workers (N). → informs this pivot information back to all workers.
shuffle	each worker sorts its data and compares this against the pivots provided. then send them to each worker.
merge	each worker bother the sended data and combine them by k-way merge.

in short, quick sort with multiple pivots!

3.a. sample sort - **if worker is dead**

if worker is dead, worker is going to resurrect soon.

but **input data will be still remained**.

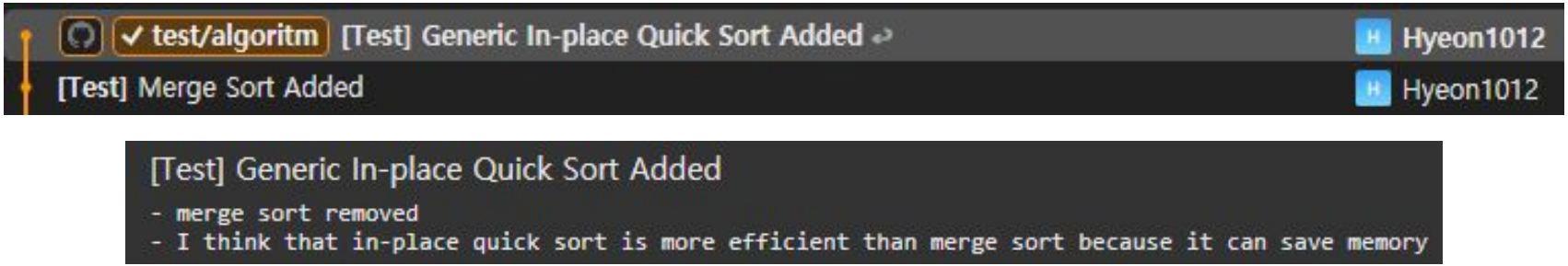
a resurrected worker gets the overall progress of sample sort at the master.

→ **do something** (for step 1 ~~~, for step 2 ~~~ , for step 3/4 ~~~)

key idea : worker creates and saves temporary files before send data to other worker

→ **even if one worker died, we can easily resume the shuffle / merge.**

3.b. k-way merge sort



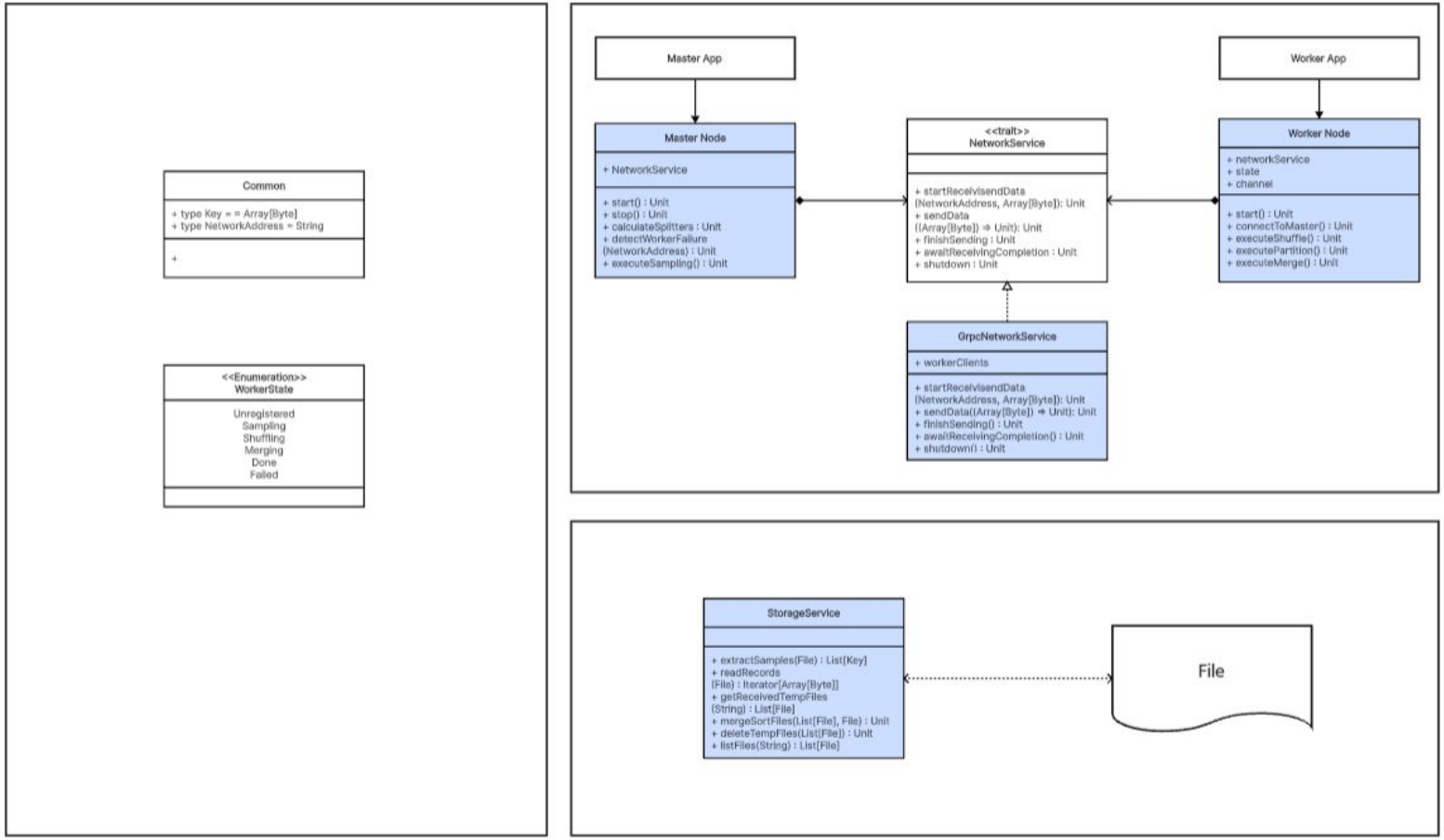
in fact, **quick sort is more efficient/faster than merge sort for smaller datasets**

so, I implemented **generic in-place quick sort**

but I noticed that I can use **java.util.Arrays.sort()**

I have to implement k-way merge by using this

Class Diagram



Review of Weekly Progress

Week 1 Setup & Planning:Project initialization, GitHub repository setup, and establishing code conventions and meeting schedules.

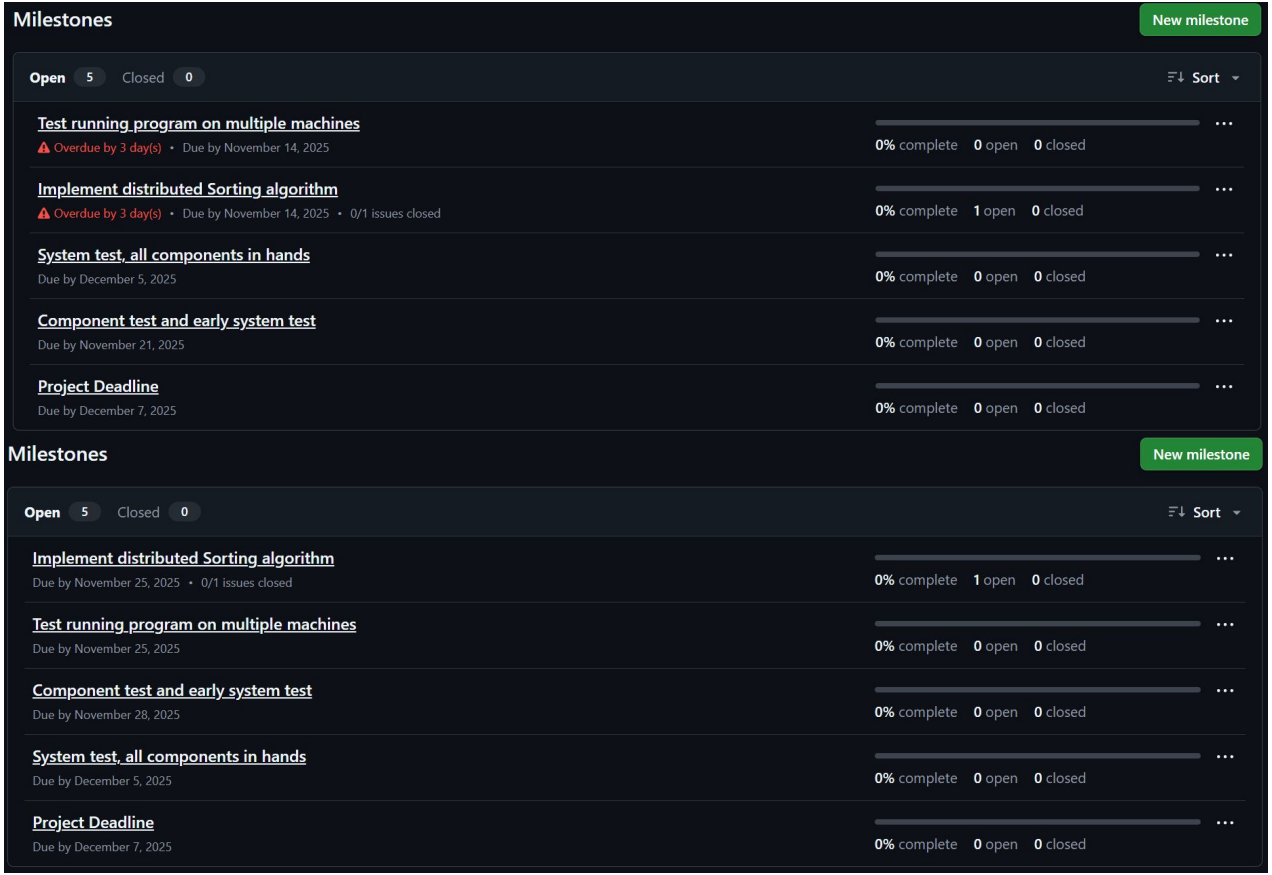
Week 2 System Architecture: Designed the high-level system architecture and discussed core concepts for distributed sorting and tech stack.

Week 3 Environment & Tooling: Configured the development environment, secured VPN/IP access, and set up essential tooling (UML, IntelliJ).

Week 4 Core Prototyping: Began implementation with gRPC/protobuf (using ScalaPB), created Master/Worker prototypes, and generated sample data.

Week 5 Sorting Logic & Integration: Drafted class diagrams, implemented core sorting logic (QuickSort), and prepared for code integration and testing.

MileStone


Milestones
New milestone
Open 5 Closed 0
Sort
Test running program on multiple machines
Overdue by 3 day(s) • Due by November 14, 2025
0% complete 0 open 0 closed
Implement distributed Sorting algorithm
Overdue by 3 day(s) • Due by November 14, 2025 • 0/1 issues closed
0% complete 1 open 0 closed
System test, all components in hands
Due by December 5, 2025
0% complete 0 open 0 closed
Component test and early system test
Due by November 21, 2025
0% complete 0 open 0 closed
Project Deadline
Due by December 7, 2025
0% complete 0 open 0 closed
Milestones
New milestone
Open 5 Closed 0
Sort
Implement distributed Sorting algorithm
Due by November 25, 2025 • 0/1 issues closed
0% complete 1 open 0 closed
Test running program on multiple machines
Due by November 25, 2025
0% complete 0 open 0 closed
Component test and early system test
Due by November 28, 2025
0% complete 0 open 0 closed
System test, all components in hands
Due by December 5, 2025
0% complete 0 open 0 closed
Project Deadline
Due by December 7, 2025
0% complete 0 open 0 closed