

# **Progress Presentation**

**Cyan: MinWoo Kim, HyeonSeo Park, JongWon Lee**

# Environment Setting

## versions & library

- jdk 17 -> most widely used
- scala 2.13 -> stable library support
- sbt 1.11.7 -> latest
- grpc-java -> fast & easy
- scalaPB -> auto-compile protobuf to scala class
- sc-opt -> command line parsing

## local environment (cannot use server together!)

- docker & docker compose -> cross platform, imitate network environment

## server environment

- pssh: send same command or file to 20 workers
- install jdk, scala, sbt, pssh, gensort at ~/.local

## input generation

- using shell script, create multiple files in multiple directory

## current deploy process

- git pull on master -> make jar by ‘sbt assembly’  
-> pscp jar -> pssh ‘java -jar ~’

```
networks:  
  internal_net:  
    driver: bridge  
    ipam:  
      config:  
        - subnet: 2.2.2.0/24  
          gateway: 2.2.2.1
```

```
[~] -bash-4.2$ pssh -h hosts1-10.txt -i "echo hi"  
[1] 02:43:46 [SUCCESS] cyan@2.2.2.104  
hi  
[2] 02:43:46 [SUCCESS] cyan@2.2.2.102  
hi  
[3] 02:43:46 [SUCCESS] cyan@2.2.2.101  
hi  
[4] 02:43:46 [SUCCESS] cyan@2.2.2.105  
hi  
[5] 02:43:46 [SUCCESS] cyan@2.2.2.103  
hi  
[6] 02:43:46 [SUCCESS] cyan@2.2.2.109  
hi  
[7] 02:43:46 [SUCCESS] cyan@2.2.2.106  
hi  
[8] 02:43:46 [SUCCESS] cyan@2.2.2.108  
hi  
[9] 02:43:46 [SUCCESS] cyan@2.2.2.107  
hi  
[10] 02:43:46 [SUCCESS] cyan@2.2.2.110  
hi
```

```
for dir_num in $(seq 1 $NUM_DIRS); do  
  DIR="/data${dir_num}/input"  
  mkdir -p "$DIR"  
  echo "Creating files in ${DIR}..."  
  
  for file_num in $(seq 1 $FILES_PER_DIR); do  
    OUTPUT_FILE="${DIR}/partition.${file_num}"  
    HEX_STRING=$(head -c 20 /dev/urandom | od -An -tx1 |  
    RAND_INT=$(echo "ibase=16;$HEX_STRING" | bc)  
    echo " Generating ${OUTPUT_FILE} (${RECORDS} record  
    gensort -a -b${RAND_INT} ${RECORDS} "${OUTPUT_FILE}"  
done  
done
```

# Git Convention

- use prefix

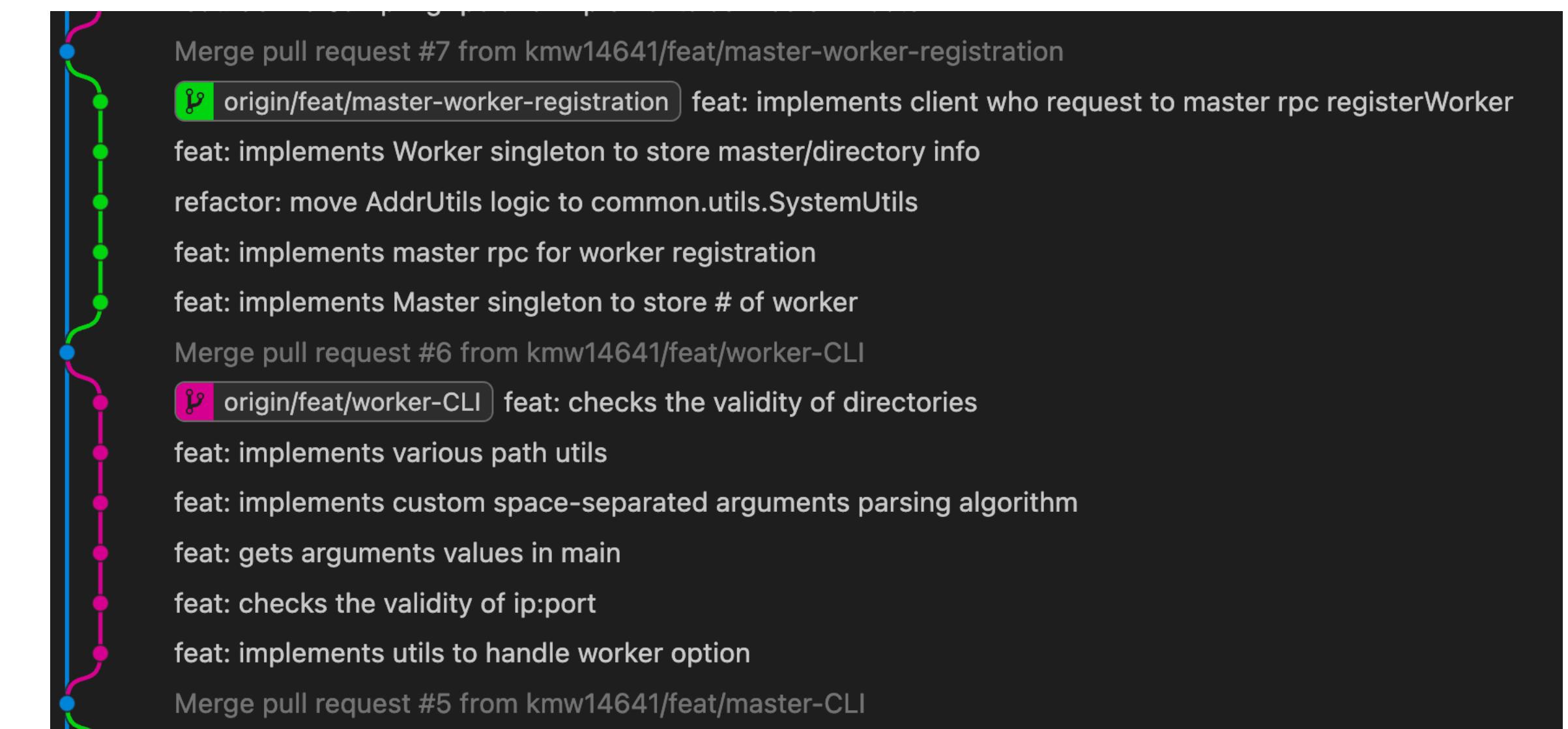
-> save reviewer's time, reduce hidden buggy code

- feat: 새 기능을 추가할 때 사용합니다.
- fix: 이전에 작성된 "잘못된 코드"를 고칠 때 사용합니다.
- update: 이미 완성된 로직에서 변경사항이 일어났을 때 사용합니다.
- refactor: 로직 변경 없이 구조적인 변경만을 포함합니다.
- rename: 이름 변경만을 포함합니다.
- chore: style, comment, gitignore 등 잡다한 커밋을 담당합니다.
- docs: 문서 관련 커밋입니다.

- unit of commit: not code-specific. feature-focused.

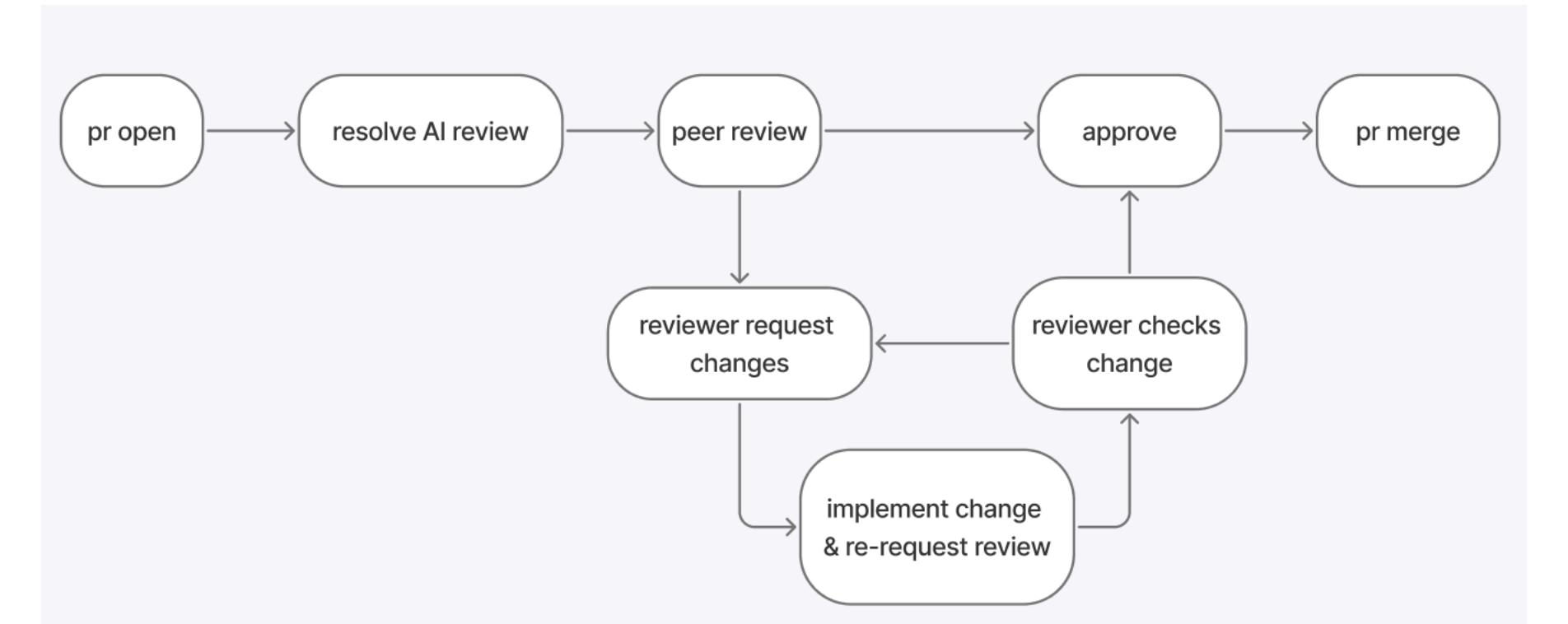
- define proto file, implement server function, register function to server builder -> X
- implement file download, implement parallel request, implement fault tolerance for request -> O

- use rebase, leave clean merge commit
  - reduce bug while resolving conflict
  - no nested merge commit



# Code Review

- use ai review
  - for each comment, reviewee should verify correctness
  - 👍 for good comment after fix, 🤢 for non-sense, otherwise leave text



- peer review lifecycle

- use issues for future improvements

- 다양한 환경에서 항상 최대의 속도를 내도록 구현  
#17 · kmw14641 opened 4 days ago
- fine grained lock 정리 [lowest]  
#16 · kmw14641 opened last week
- intermediate 파일 관리 [medium]  
#15 · kmw14641 opened last week
- 디렉토리 구조 및 객체 역할 정리 [high]  
#14 · kmw14641 opened last week
- 에러 처리 컨벤션 설정 및 적용 [high]  
#13 · kmw14641 opened last week
- 마스터 <-> 워커, 워커 <-> 워커 gRPC 채널을 초기에 열어두고 재사용하도록 수정 [high]  
#12 · kmw14641 opened last week
- 포매터 & 린터 설정 [low]  
#11 · kmw14641 opened last week

- github-discord webhook for notifications about pull request

gemini-code-assist bot last week

Medium Priority

map 함수는 컬렉션을 변환하여 새로운 컬렉션을 생성하는 데 사용됩니다. 여기서는 디버깅 목적으로 각 원소를 출력하는 부수 효과(side effect)를 위해 map 을 사용하고 있습니다. 스칼라에서는 이런 경우 foreach 를 사용하는 것이 더 관용적이고 코드의 의도를 명확하게 나타냅니다. map 을 사용하면 새로운 컬렉션이 생성되지만 사용되지 않아 혼란을 줄 수 있습니다.

Suggested change

```
31 - workersRangeAssignment.map {  
32 -   // Print assigned ranges for debugging  
33 -   case ((ip, port), (start, end)) =>  
34 -     val startInt = new BigInteger(1, start.toByteArray())  
35 -     val endInt = new BigInteger(1, end.toByteArray())  
36 -     println(s"Assigned range to worker $ip:$port =>  
37 -       ${startInt.toString(16)}, ${endInt.toString(16)}")  
  
31 + workersRangeAssignment.foreach {  
32 +   // Print assigned ranges for debugging  
33 +   case ((ip, port), (start, end)) =>  
34 +     val startInt = new BigInteger(1, start.toByteArray())  
35 +     val endInt = new BigInteger(1, end.toByteArray())  
36 +     println(s"Assigned range to worker $ip:$port =>  
37 +       ${startInt.toString(16)}, ${endInt.toString(16)})")  
}
```

Commit suggestion ▾

# github

Feat/sync phase

kmw14641 [kmw14641/332project] New review comment on pull request #21: Feat/sync phase 이거 Worker에 상수로 넣으시죠 저도 거기다 모아놨어요

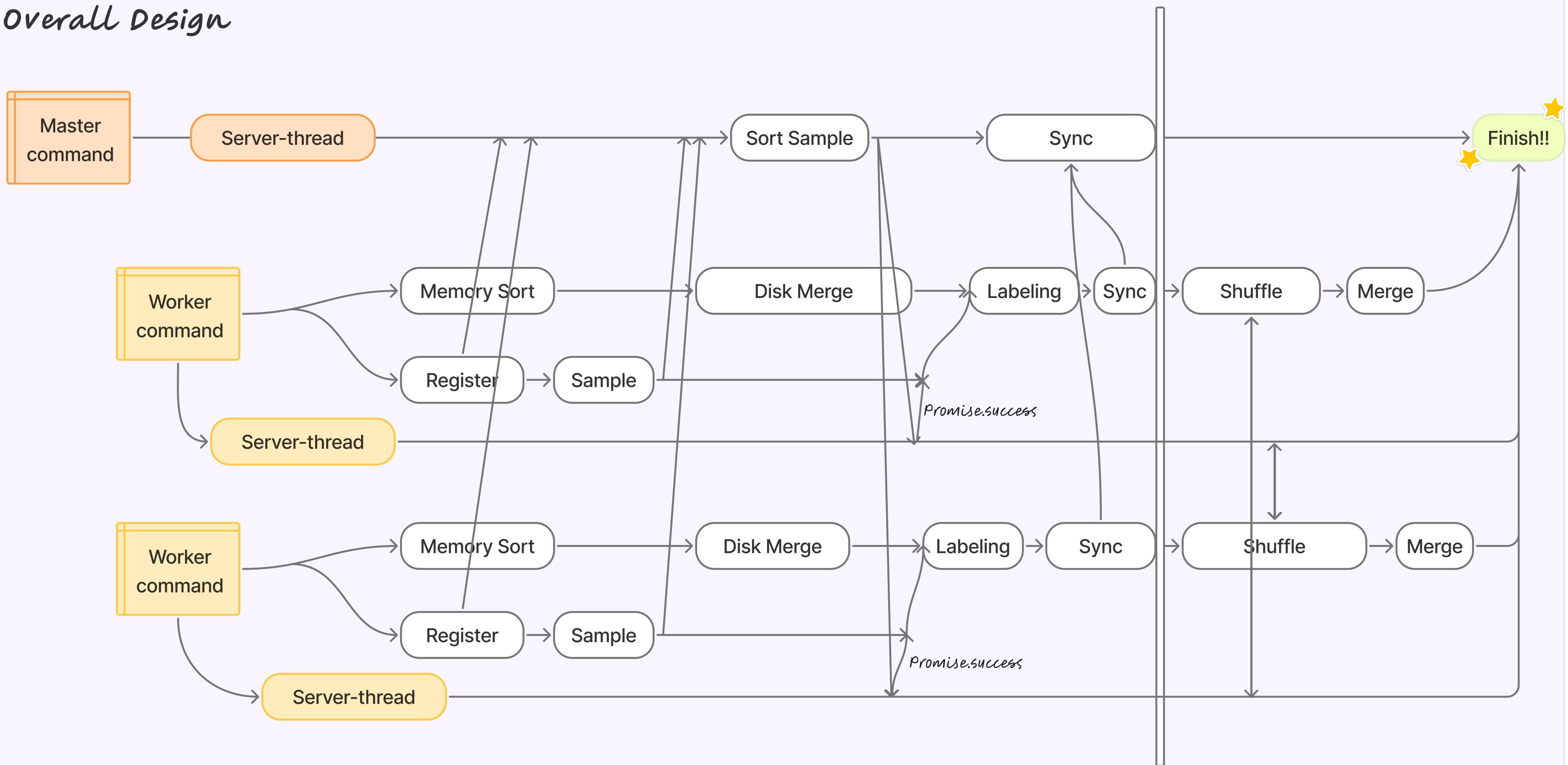
kmw14641 [kmw14641/332project] Pull request review submitted: #21 Feat/sync phase 완

GitHub 2025. 11. 15. 오후 7:26

kmw14641 [kmw14641/332project] Pull request opened: #22 mac에서 만들아가는 문제 수정

gemini-code-assist[bot]

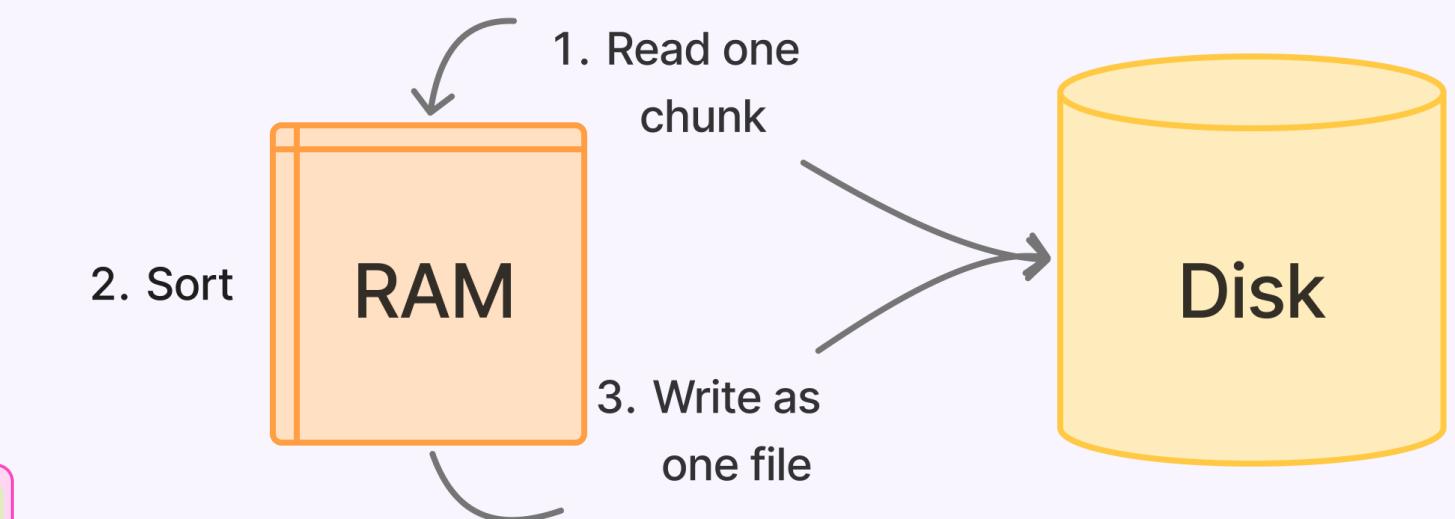
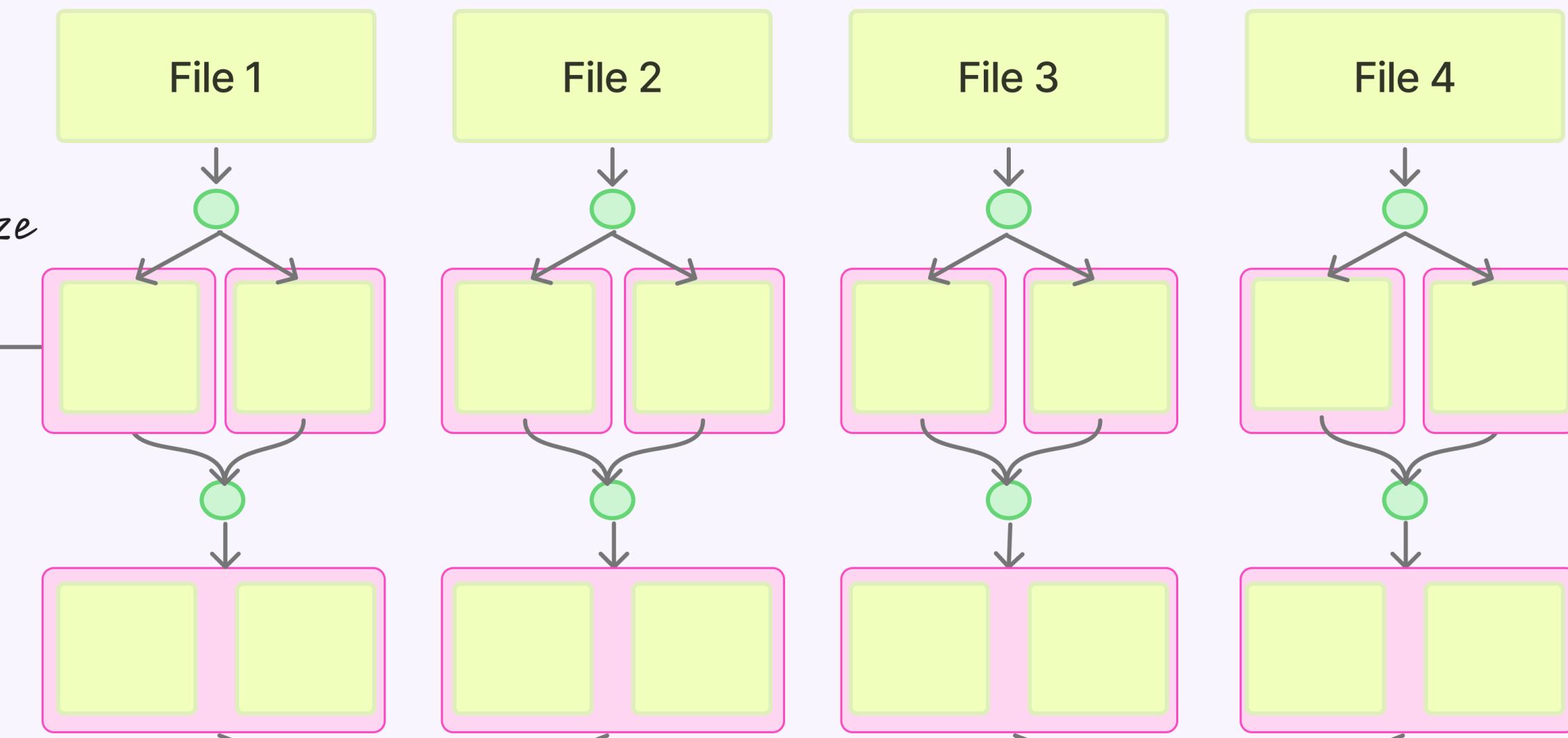
## Overall Design



## Disk based merge sort

<memory sort>

output file size = safe chunk size

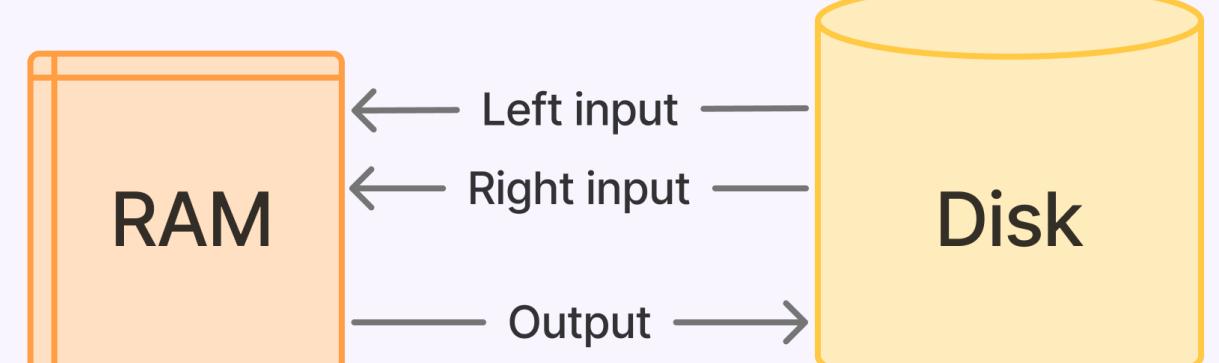
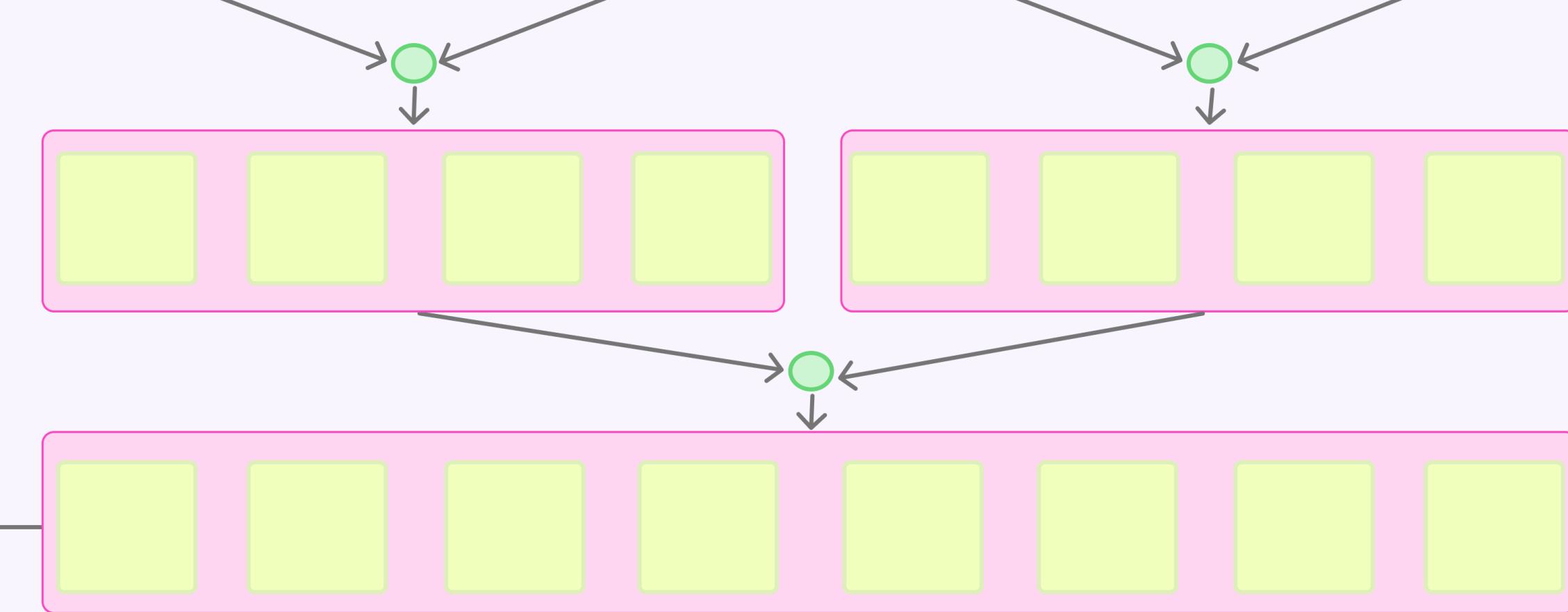


3 \* file size on memory

2 for read, 1 for write

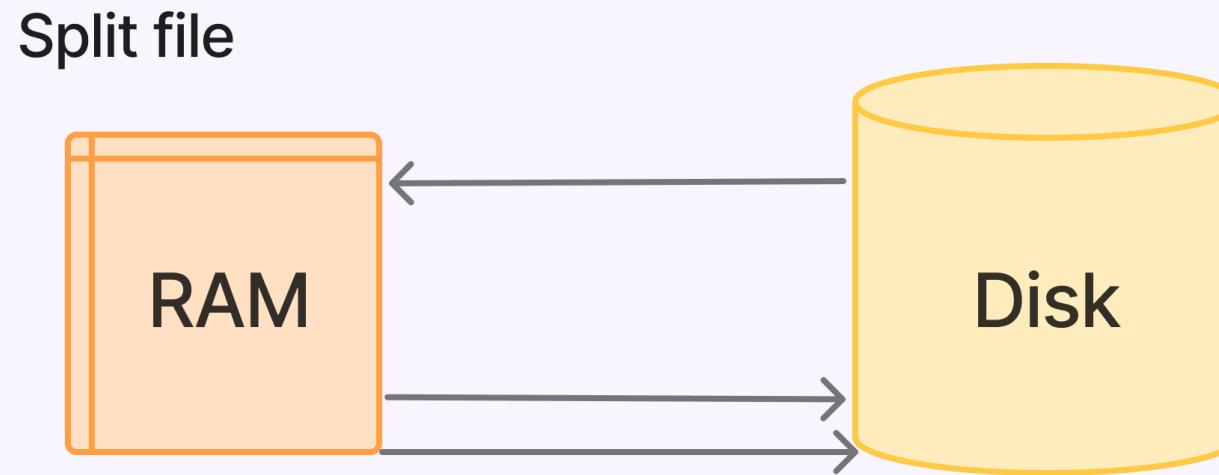
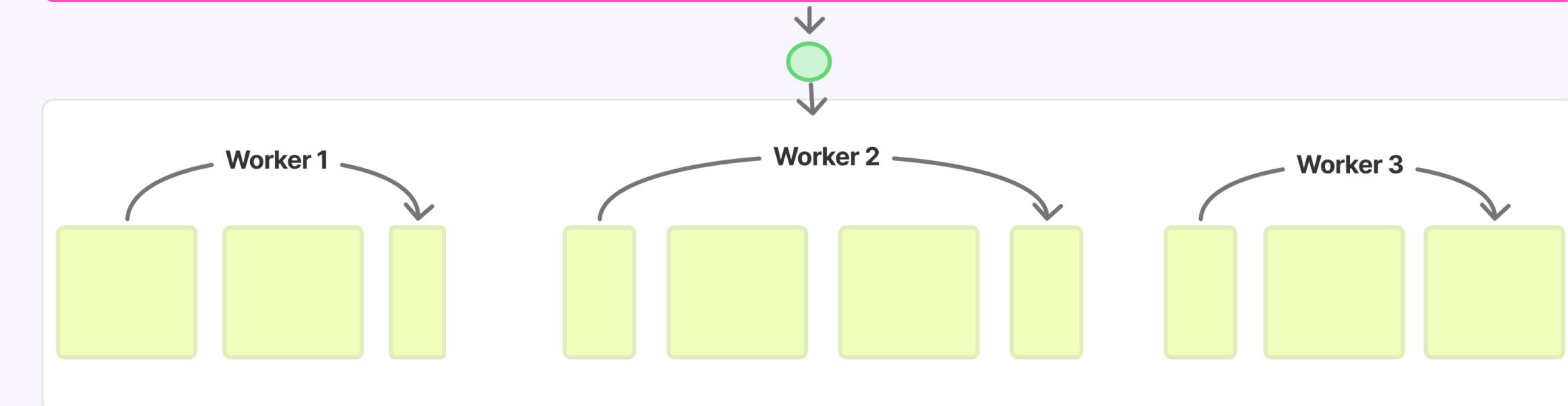
<Disk merge sort>

Additional handling of  
file size, thread pool size  
for low memory case



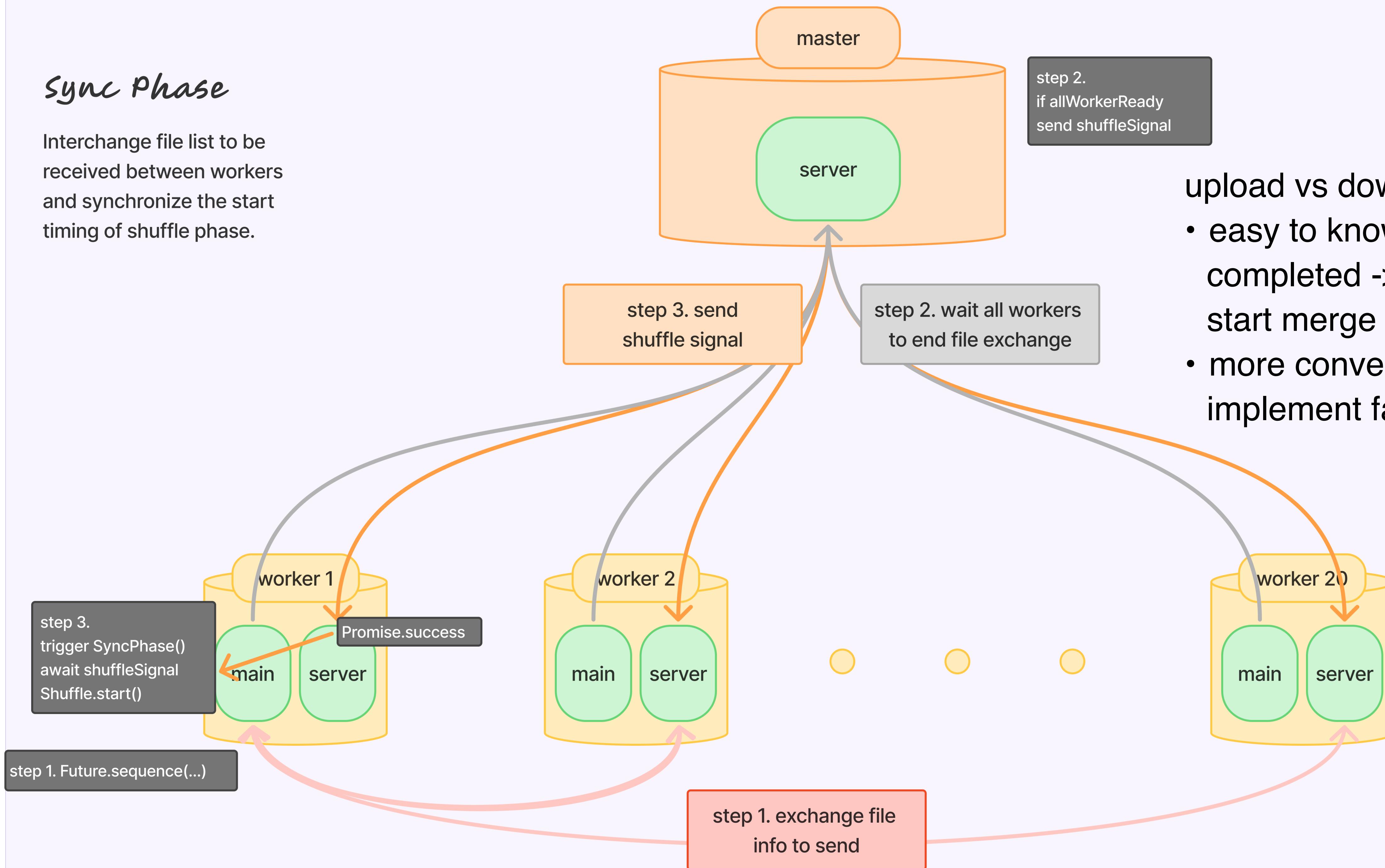
<Labeling>

use binary search to find pivot point



## Sync Phase

Interchange file list to be received between workers and synchronize the start timing of shuffle phase.

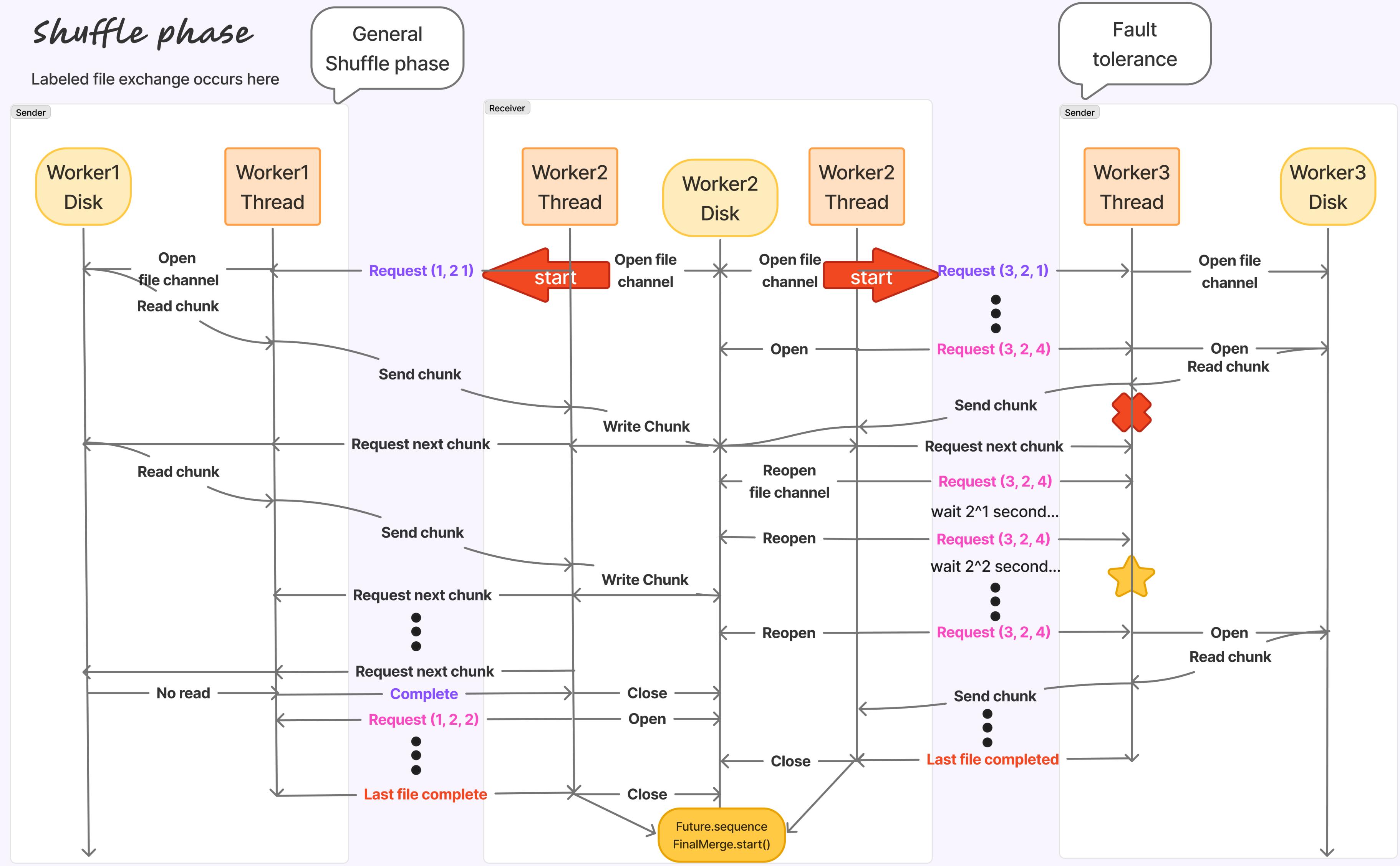


## upload vs download

- easy to know shuffle completed -> start merge
- more convenient to implement fault tolerance

## shuffle phase

Labeled file exchange occurs here



## protobuf:

```
returns (stream DownloadResponse);
```

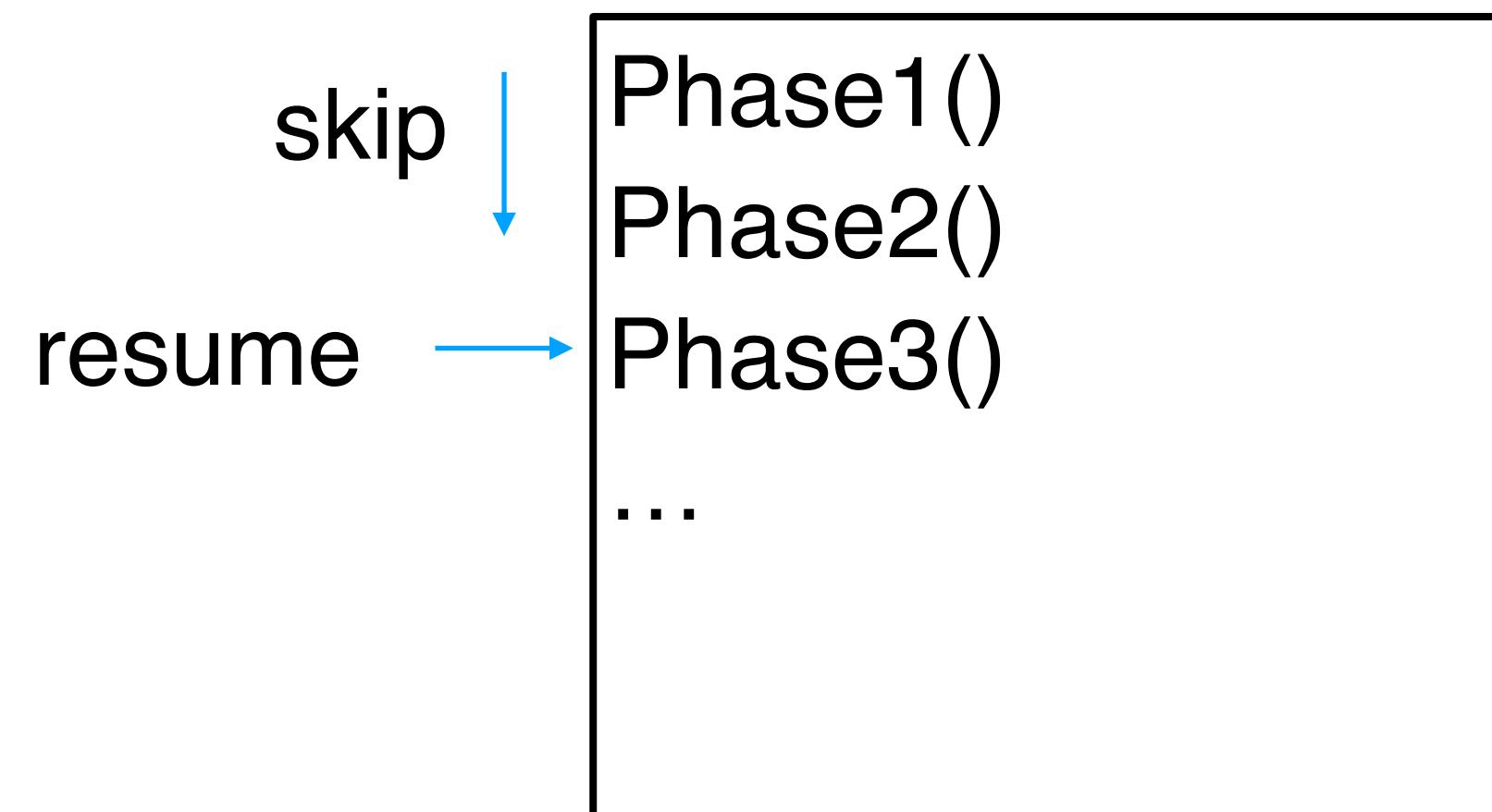
- on scale 0.1, 10 minute
- increase file size -> 9 minute (?)
  - global disk lock -> 7 minute
  - increase chunk size -> 6 minute

# State restoration (Design)

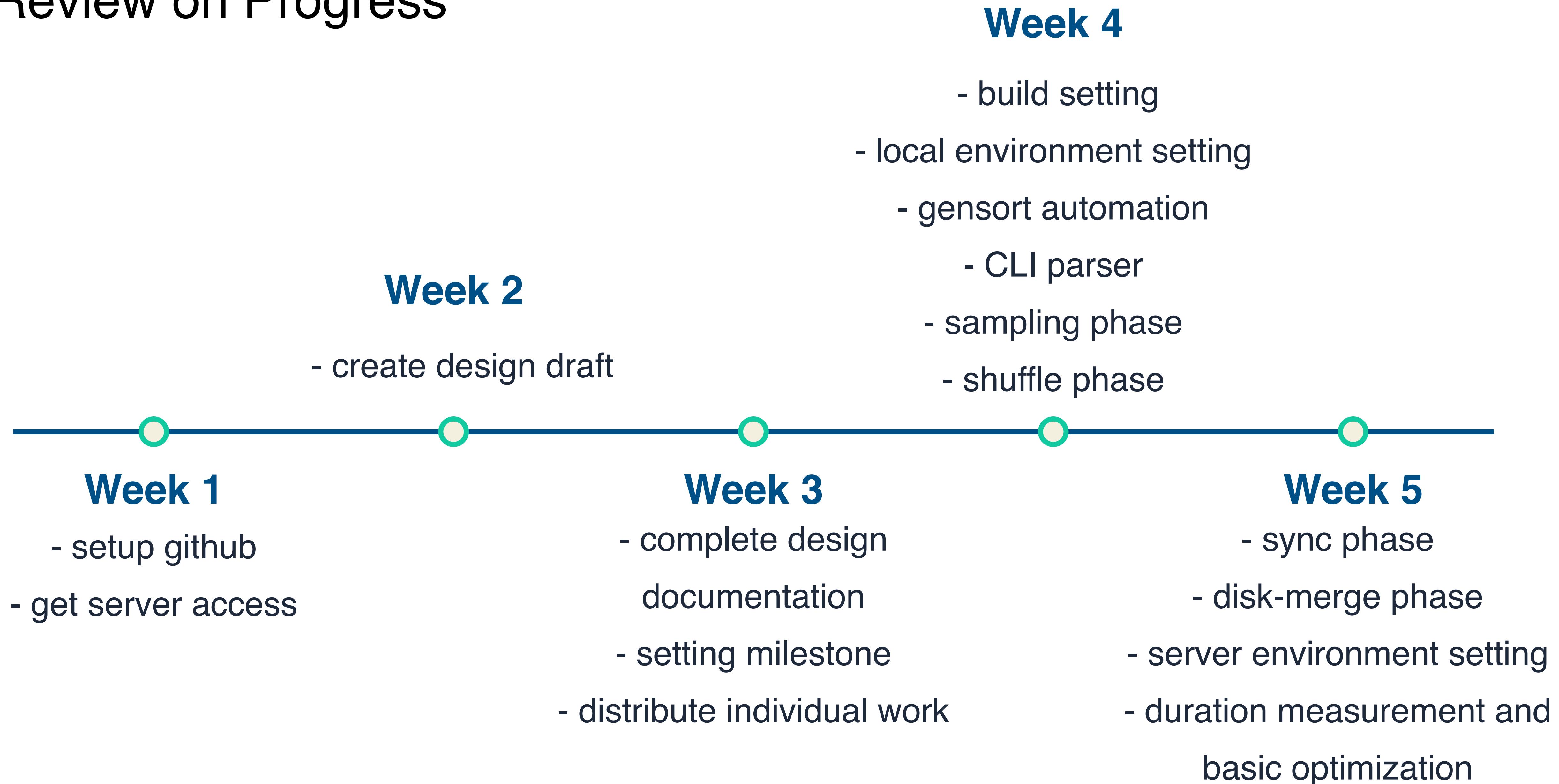
- Worker has global state
- Each phase records current progress after some work
  - ex) remaining file list for shuffle phase
- State is also stored in disk file
- When machine restarts
  - Broadcast new port
  - Execute program from start
  - For each phase, skip if already processed
  - In phase, skip already processed work, and resume

current phase: shuffle  
remaining file list: [~]  
...

file: worker\_state



# Review on Progress



# Project Milestones, Future Works

## Milestone 1: Setting development environment

- 1-1: Setting server environment
  - Execution environment update (completed)
  - Install code or binary (almost)
- 1-2: Setting local environment (completed)

## Milestone 2: Complete Implementation

- 2-1: Entrypoint of program (completed)
- 2-2: Sampling phase (completed)
- 2-3: Sort on local machine (reviewing)
- 2-4: Synchronization phase (considering fault tolerance) (reviewing)
- 2-5: Shuffle phase (considering fault tolerance) (reviewing)
- 2-6: State restoration
- 2-7: Final merge phase

## Milestone 3: Debugging

Check correctness of program

## Milestone 4: Test

Unit test & Integration test

Test fault tolerance scenario

## Milestone 5: Optimization

Parallel operation, etc

## Additional Objectives

Refactoring

- more OOP style, error handling convention, ...

Logging