

CS332

# Progress Presentation

Team Blue

# Review of weekly progress(week 1)

## What we do

- Kick-off team meeting
- Planning for the project

## Review

- First time to meet each other
- Set up the project

The screenshot displays a project management dashboard for the 'CSE-332 Team Project'. The interface is organized into several sections:

- Project Description:** Includes a link to 'sd\_project.pdf' (1208.5 KB).
- Github Repository:** Shows a repository named '332project' with a 'Project ID' field.
- Project Progress:** Lists 'Milestones' and 'Progress Presentation' with a note to 'Type "/" for commands'.
- Weekly Progress:** A list of weeks from Week1 to Week4.
- Board:** A Kanban-style board with columns for each week, showing task status (Not started, In progress, Done).
- Tasks & Planning:** A detailed view of tasks for each week, including descriptions, assignees, and status indicators.

**Task Details by Week:**

Week	Task	Status	Assignees
Week 1 (10/23)	list tasks and set milestones	Done	
Week 2 (10/24-10/30)	study GRPC library	Done	SubinKim, Mathis
	study gensort & valsart	Done	
	build scala executable file	Done	SubinKim, Mathis
Week 3 (10/31-11/6)	basic master, slave app with grpc communication	Done	
	print master's ip and port	Done	
	connect slave to master with argv master address	Done	
	send ip address from slaves to master	Done	
Week 4 (11/6-11/13)	Testing automation	Not started	
	load file from the disk	In progress	
	sample data from the file and send to master	Not started	
	master should partition the ranges and send back to slave for key ranges	In progress	
	Assignment 7	Not started	
Week 5 (11/14-11/20)	sort all files in the worker and save with key ranges partition	Not started	
	Shuffle the sorted file	Not started	
Week 6 (11/21-11/27)	Merge the sorted files and save into partitions	Not started	

# Review of weekly progress(week 2)

## What we do

- learn how to use grpc in Scala
  - grpc-java, scalapb
- build executable file with Scala application

## Review

- Warm up for the project

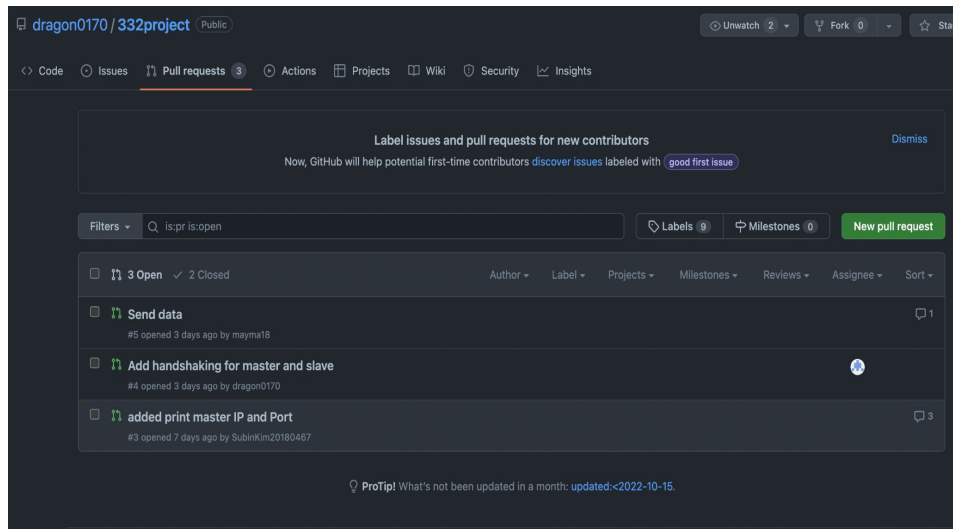
# Review of weekly progress(week 3-4)

## What we do

- Set up the basic scala project with master and slave application(sbt, grpc)
- Implement handshaking feature
  - master prints ip:port, slave connects to master and send ip to master, master waits and prints ip lists
- Open a file from the disk and send file data as string with grpc request

## Review

- Something is going wrong
- **We need a change**



# Member Role(Before)

- Self-directed, Democratic
- 최규용, 김수빈, Mathis
  - All members divide the tasks for a week with discussion.
  - Do the task during a week.
  - Review each other's task.

# Member Role(After)

- Surgical, Aristocracy
- 최규용
  - Main leader, surgeon
  - Overall design
  - Specify and distribute tasks
  - Do the assigned tasks
  - Review code and documentation
- 김수빈, Mathis
  - Do the assigned tasks
  - Review code and documentation

# Logistics

- **In English**
- Communication
  - main communication - Kakao Talk
  - code review and discussion - Github Pull Request
  - **(NEW)weekly meeting for progress review at every sunday afternoon - Online video call**
- Documentation
  - project & task management, general documentation - Notion
  - requirements, how to build, install - Github README.md
  - diagram, slides, spreadsheets, etc. - Google Docs, Draw.io

# Logistics

☰ README.md

## 332project

### Requirements

- JDK v1.8.0
- Scala v2.13.10
- sbt v1.7.3

### Build

This command compiles protobuf file to scala class.

```
sbt compile
```

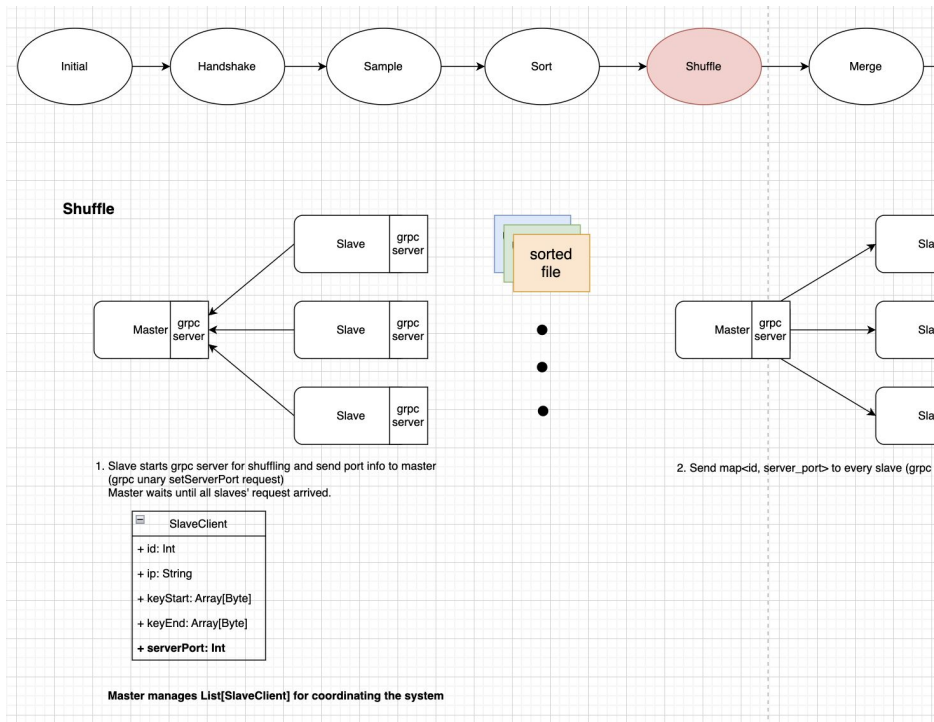
### Installation

```
sbt stage
```

### Milestones

#### #1(~11/7)

- Generate unsorted data files
- Learn grpc
- Slave connects to Master with grpc

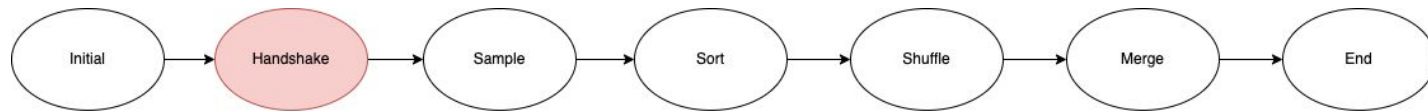




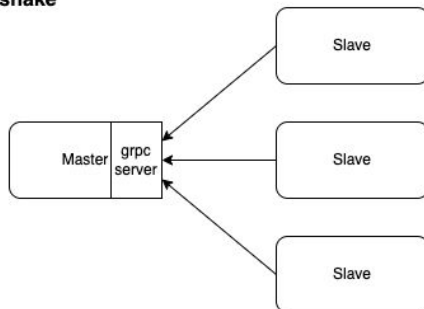
# Details

- Programming Environments
  - OS
    - Local
      - macOS 12.6
      - Ubuntu 22.04.1
    - Test(work in progress)
      - Ubuntu by Docker(TBD)
  - **JDK v1.8.0\_352**
  - sbt v1.7.3
  - **scala v2.13.10**
  - protobuf v3.19.2
  - gensort v1.5
- Libraries
  - grpc-java v1.46.0
  - scalapb v0.11.11
- Logging
  - log4j v2.19.0

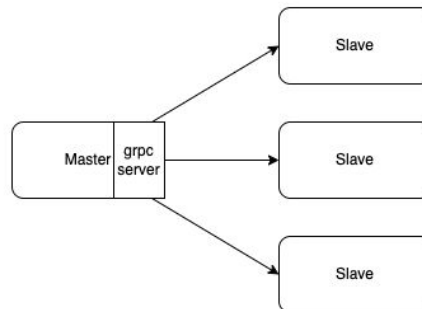
# Design



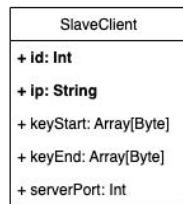
## Handshake



1. Send slave's IP to master (grpc unary handshake request)  
Master waits until all slaves' request arrived.

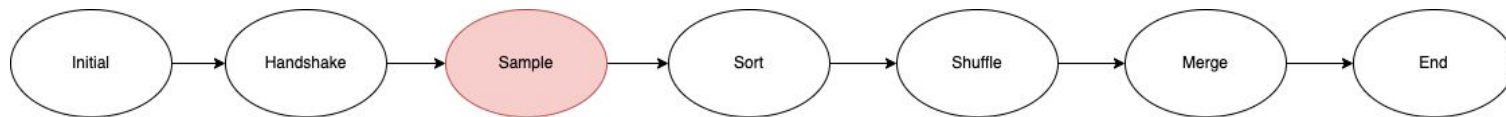


2. Send slave id to each slave (grpc unary handshake response)  
From now, slave will send all grpc requests with its own id



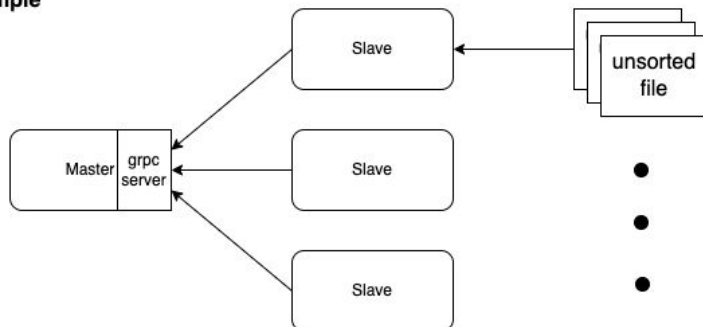
Master manages List[SlaveClient] for coordinating the system

# Design

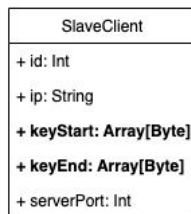


## Sample

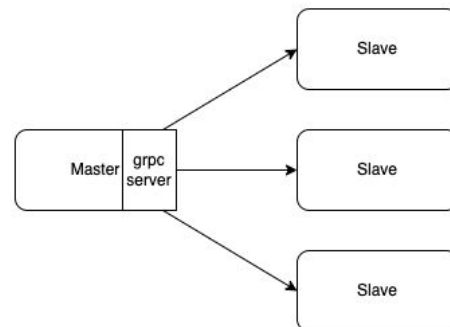
1. Open and sample 1MB data from each file and concatenate them



2. Send sampled data to master (grpc unary sendSampleData request)  
Master waits until all slaves' request arrived.  
Master analyze collected sample data and save key ranges(start, end) for each slave

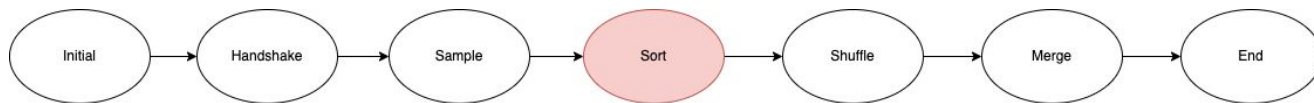


Master manages List[SlaveClient] for coordinating the system



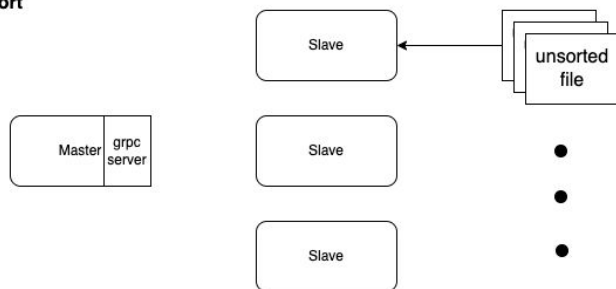
3. Send map<id, key\_ranges> to every slave (grpc unary sendSampleData response)  
Slave saves the mapping data in the application.

# Design



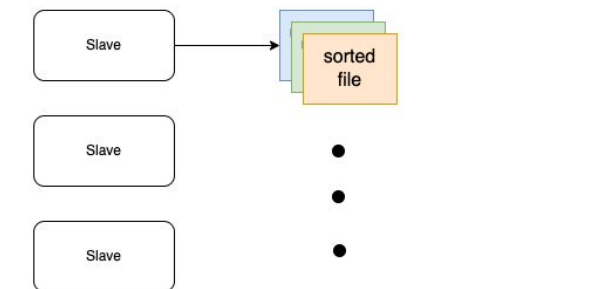
## Sort

1. Open file and sort data in the file one by one



2. Partition sorted data using the mapping data of `map<id, key_ranges>`

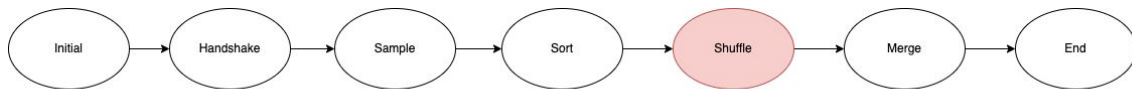
SlaveClient
+ id: Int
+ ip: String
+ keyStart: Array[Byte]
+ keyEnd: Array[Byte]
+ serverPort: Int



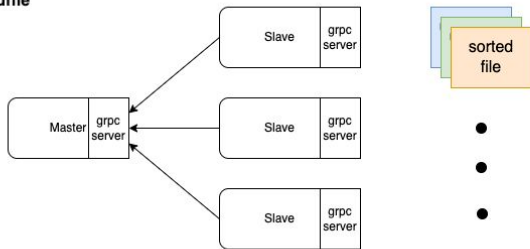
3. Save partitioned data into files with name of slave id labeling. (ex. sorted\_num\_targetSlaveId)  
Iterate all files in the directory.

Master manages `List[SlaveClient]` for coordinating the system

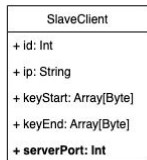
# Design



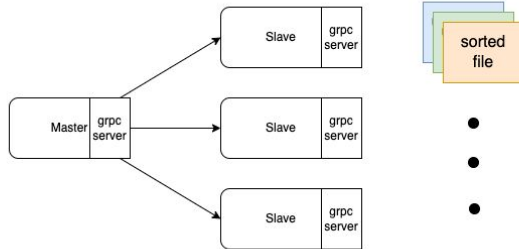
## Shuffle



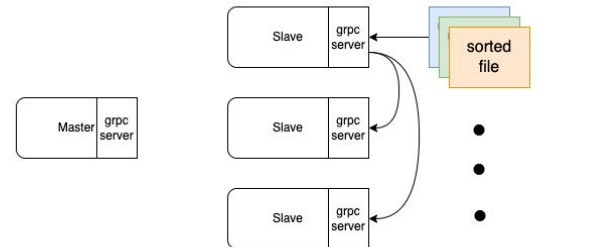
1. Slave starts grpc server for shuffling and send port info to master (grpc unary setServerPort request)  
Master waits until all slaves' request arrived.



Master manages List[SlaveClient] for coordinating the system

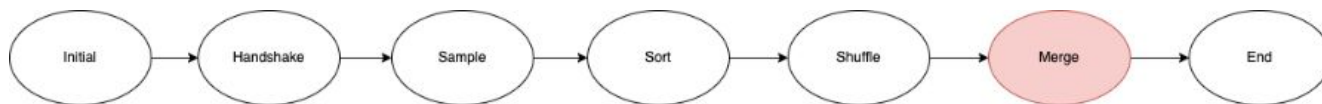


2. Send map<id, server\_port> to every slave (grpc unary setServerPort response)

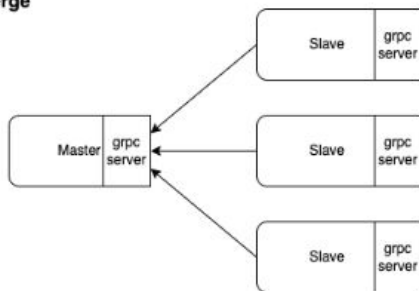


3. Send each sorted file to the appropriate slave's grpc server (grpc client-streaming sendSortedFiles request)  
Each slave waits until receiving files from all slaves.

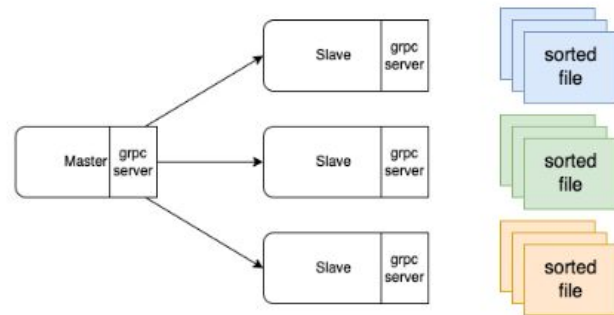
# Design



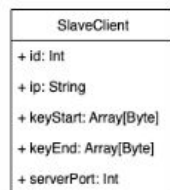
## Merge



1. Slave send the number of sorted files to master (grpc unary sendNumFile request)  
Master waits until all slaves' request arrived.

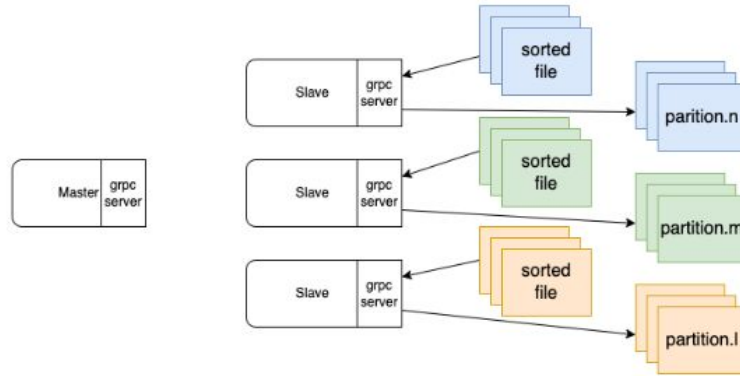


2. Send start index of partitioned file to every slave (grpc unary sendNumFile response)

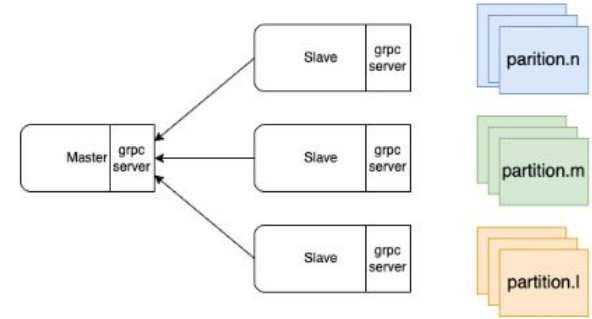


Master manages List[SlaveClient] for coordinating the system

# Design



3. Each slave runs k-way merge algorithm for sorted files and save the final partition files.



4. Slave send message for completion of merging to master (grpc unary notifyMergingCompleted request)  
Now, slave can shutdown the application.  
Master waits until all slaves' request arrived and shutdown the application after that.

# Progress

- We set 4 big milestones at the beginning.
  - a. Be familiar with **gensort/valsort, grpc, protobuf**. Master can connect to Slave with grpc and **sends ip address and prints the ip address list**. (~11/7)
  - b. **Sample data** from the file in each worker. Master **determines and broadcasts sorting key ranges for each slave**. (~11/13)
  - c. **Sort input files** in each slave and **save the sorted results into partitioned files** with appropriate key ranges. (~11/20)
  - d. **Shuffle the sorted files** with each other between slaves. **Merge all sorted files** in each slave and **save into partitioned files** with appropriate size. (~12/4)
- Completed milestones
  - First milestone



# Progress

- Modules implemented so far
  - Handshaking
    - connects and send ip address from slave to master
    - master waits until all slaves are connected and prints ip list
- Modules that are not working yet
  - Sampling(In progress)
  - Sorting
  - Shuffling
  - Merging