CS332
Progress Presentation

Team Blue

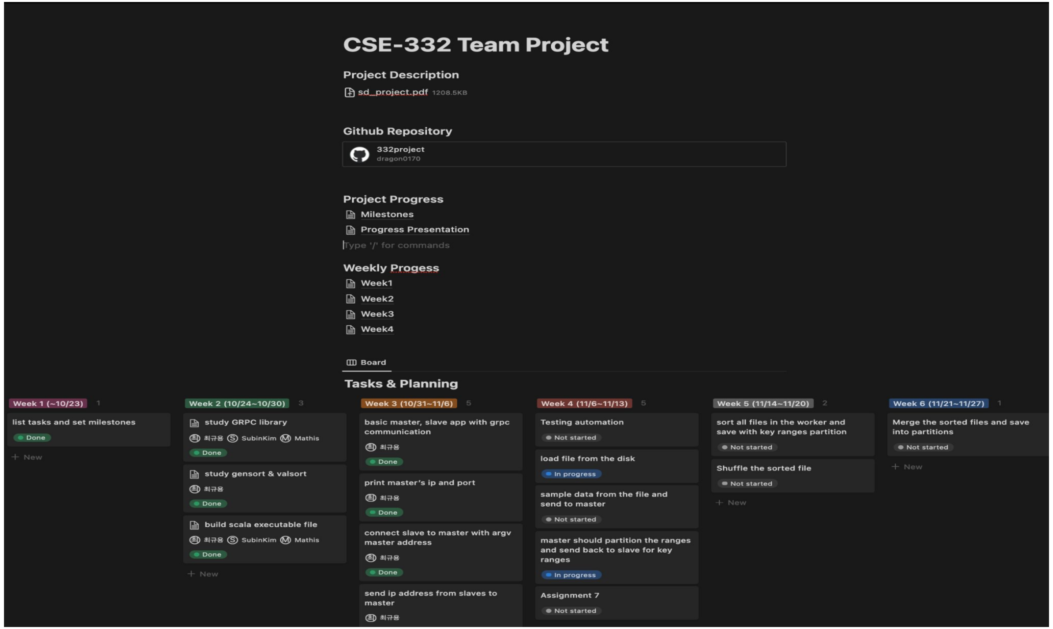
Review of weekly progress(week 1)

What we do

- Kick-off team meeting
- Planning for the project

Review

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



CSE-332 Team Project

Project Description

sd_project.pdf 1208.5KB

Github Repository

332project
dragon0170

Project Progress

Milestones
Progress Presentation
Type '/' for commands

Weekly Progess

Week1
Week2
Week3
Week4

Board

Tasks & Planning

Week 1 (~10/23) 1
- list tasks and set milestones — Done

Week 2 (10/24~10/30) 3
- study GRPC library — 최규용, SubinKim, Mathis — Done
- study gensort & valsort — 최규용 — Done
- build scala executable file — 최규용, SubinKim, Mathis — Done

Week 3 (10/31~11/6) 5
- basic master, slave app with grpc communication — 최규용 — Done
- print master's ip and port — 최규용 — Done
- connect slave to master with argv master address — 최규용 — Done
- send ip address from slaves to master — 최규용

Week 4 (11/6~11/13) 5
- 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) 2
- 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) 1
- 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

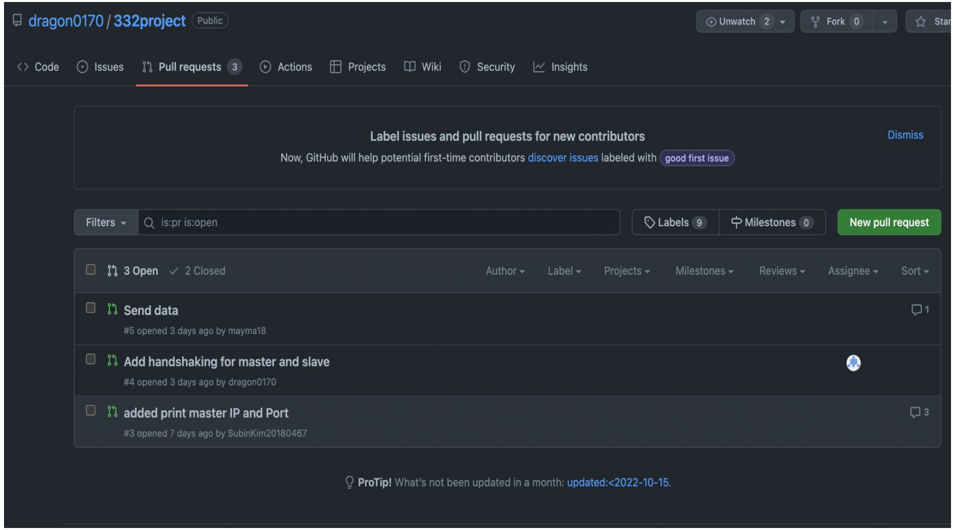
Review of weekly progress(week 3-4)

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**



Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

□ Send data 1
#5 opened 3 days ago by mayma18

□ Add handshaking for master and slave
#4 opened 3 days ago by dragon0170

□ added print master IP and Port 3
#3 opened 7 days ago by SubinKim20180467

ProTip! What's not been updated in a month: updated:<2022-10-15.

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.

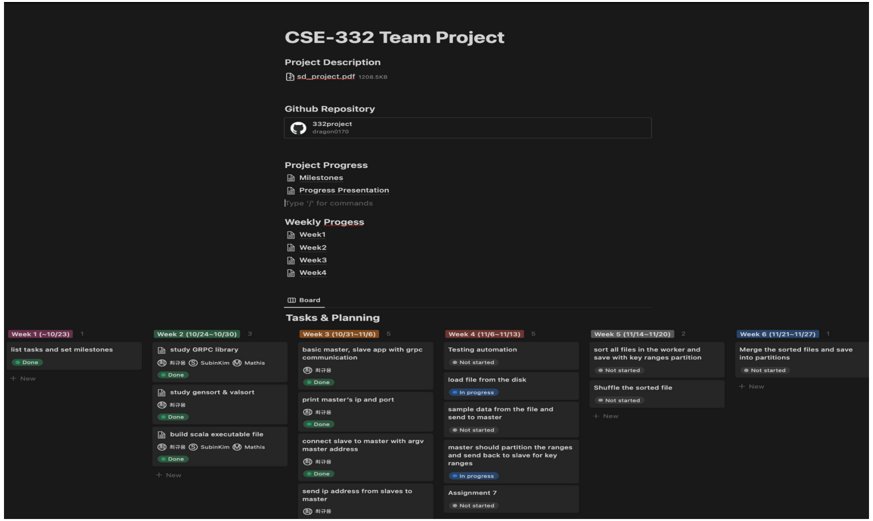
Member Role(After)

- 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



CSE-332 Team Project

Project Description

sd_project.pdf 1208.5KB

Github Repository

332project
dragon0170

Project Progress

- Milestones
- Progress Presentation

Type '/' for commands

Weekly Progess

- Week1
- Week2
- Week3
- Week4

Board

Tasks & Planning

Week 1 (~10/23) 1
- list tasks and set milestones — Done

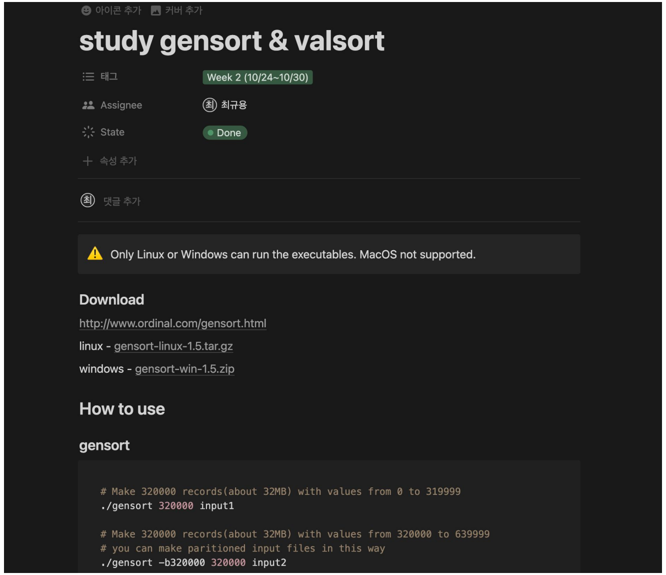
Week 2 (10/24~10/30) 3
- study GRPC library — 최규용, SubinKim, Mathis — Done
- study gensort & valsort — 최규용 — Done
- build scala executable file — 최규용, SubinKim, Mathis — Done

Week 3 (10/31~11/6) 5
- basic master, slave app with grpc communication — 최규용 — Done
- print master's ip and port — 최규용 — Done
- connect slave to master with argv master address — 최규용 — Done
- send ip address from slaves to master — 최규용

Week 4 (11/6~11/13) 5
- 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) 2
- 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) 1
- Merge the sorted files and save into partitions — Not started



아이콘 추가 커버 추가

study gensort & valsort

태그: Week 2 (10/24~10/30)

Assignee: 최규용

State: Done

속성 추가

댓글 추가

⚠️ Only Linux or Windows can run the executables. MacOS not supported.

Download

http://www.ordinal.com/gensort.html

linux - gensort-linux-1.5.tar.gz

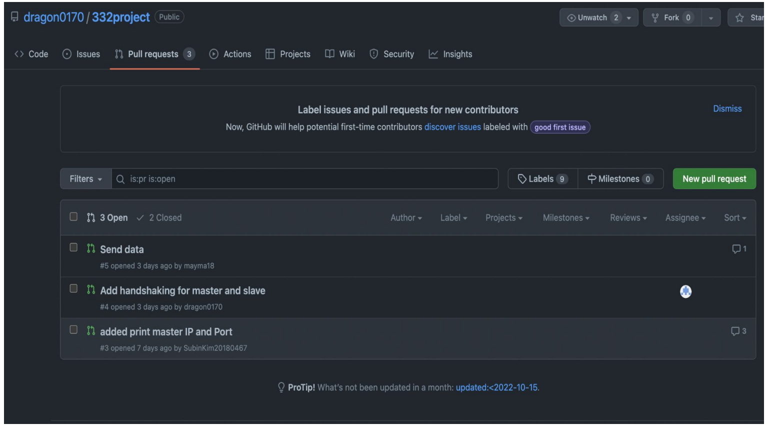
windows - gensort-win-1.5.zip

How to use

gensort

```
# Make 320000 records(about 32MB) with values from 0 to 319999
./gensort 320000 input1

# Make 320000 records(about 32MB) with values from 320000 to 639999
# you can make paritioned input files in this way
./gensort -b320000 320000 input2
```



Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

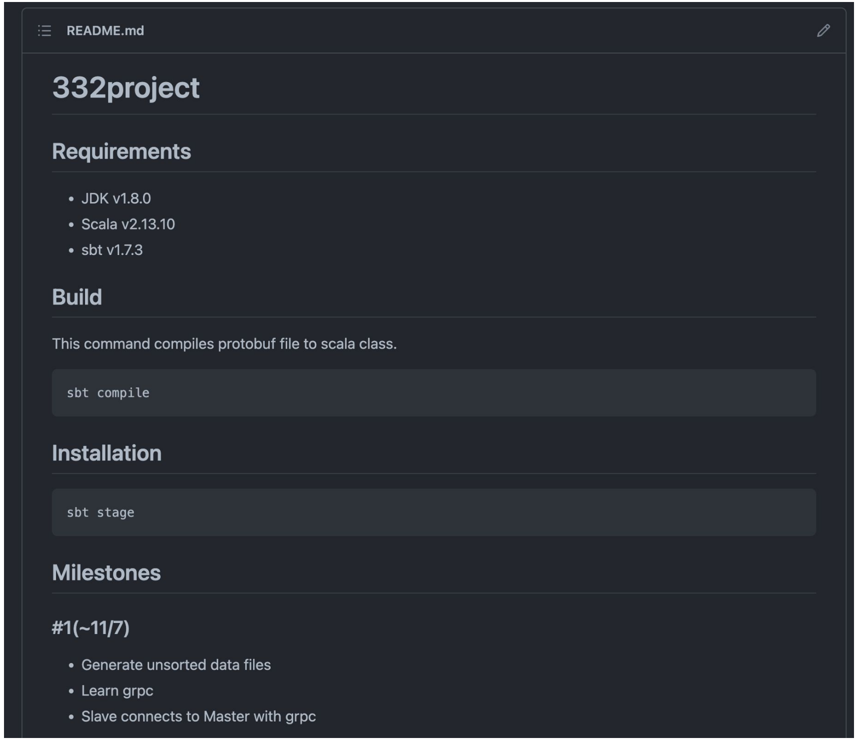
is:pr is:open

3 Open ✓ 2 Closed

- Send data — #5 opened 3 days ago by mayma18 — 1
- Add handshaking for master and slave — #4 opened 3 days ago by dragon0170
- added print master IP and Port — #3 opened 7 days ago by SubinKim20180467 — 3

ProTip! What's not been updated in a month: updated:<2022-10-15.

Logistics



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
```

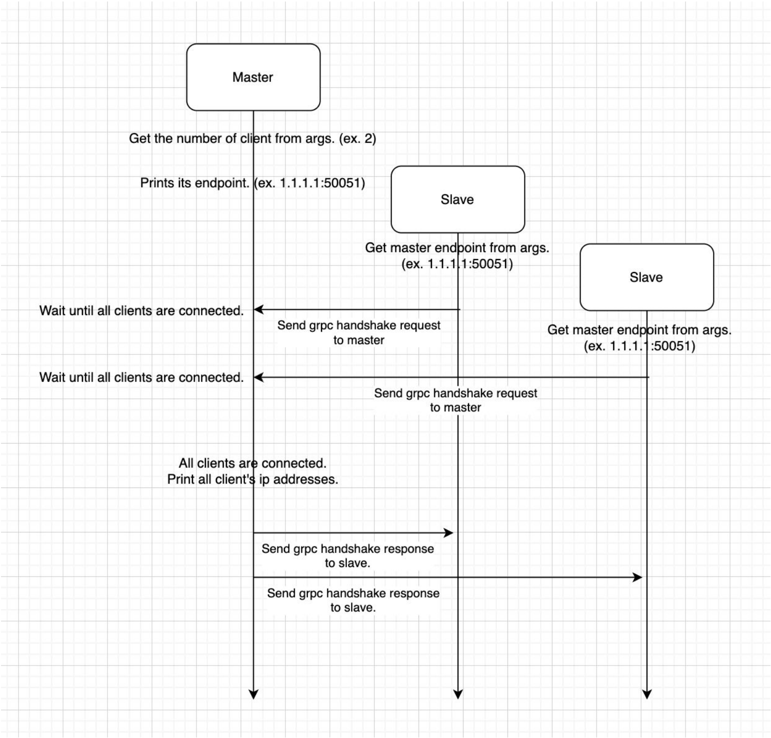
Installation

```
sbt stage
```

Milestones

#1(~11/7)

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



Master

Get the number of client from args. (ex. 2)

Prints its endpoint. (ex. 1.1.1.1:50051)

Slave

Get master endpoint from args.
(ex. 1.1.1.1:50051)

Slave

Wait until all clients are connected.

Send grpc handshake request
to master

Get master endpoint from args.
(ex. 1.1.1.1:50051)

Wait until all clients are connected.

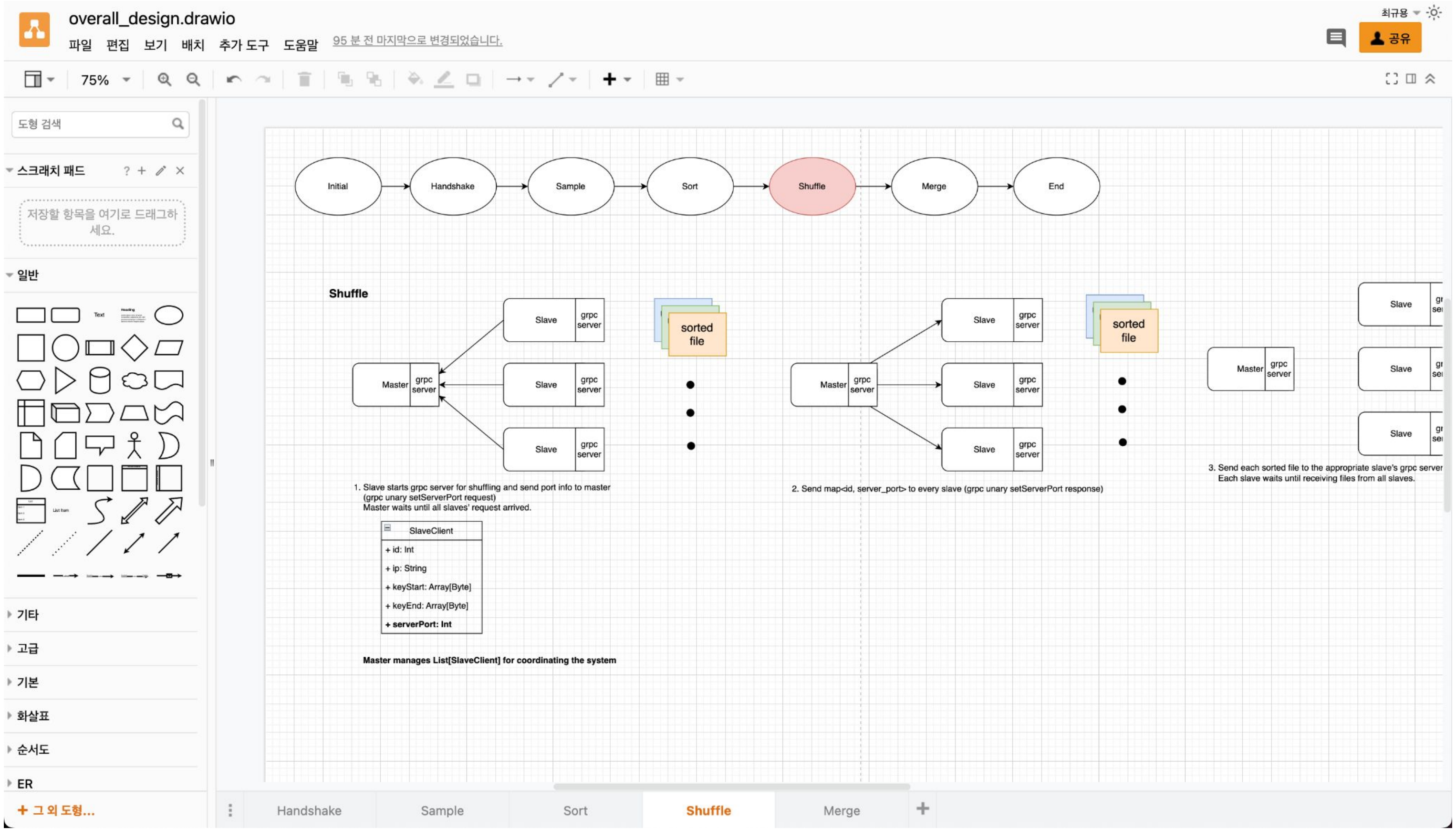
Send grpc handshake request
to master

All clients are connected.
Print all client's ip addresses.

Send grpc handshake response
to slave.

Send grpc handshake response
to slave.

Logistics



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

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
   Each slave waits until receiving files from all slaves.

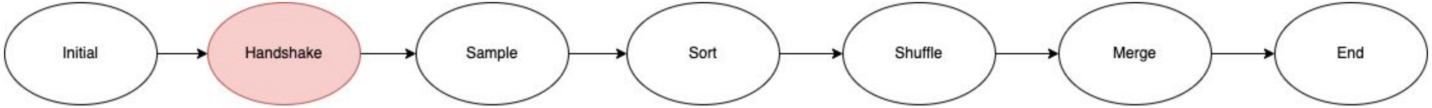
SlaveClient
+ id: Int
+ ip: String
+ keyStart: Array[Byte]
+ keyEnd: Array[Byte]
+ serverPort: Int

Master manages List[SlaveClient] for coordinating the system

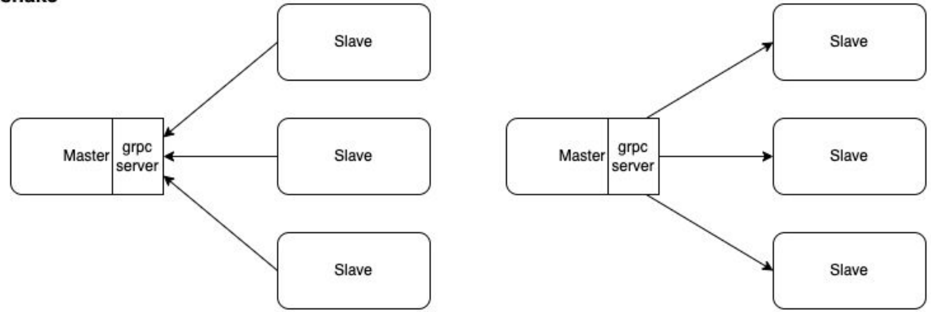
Details

- 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

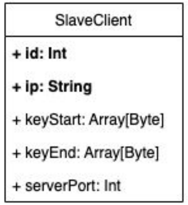


Handshake



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



SlaveClient

+ id: Int

+ ip: String

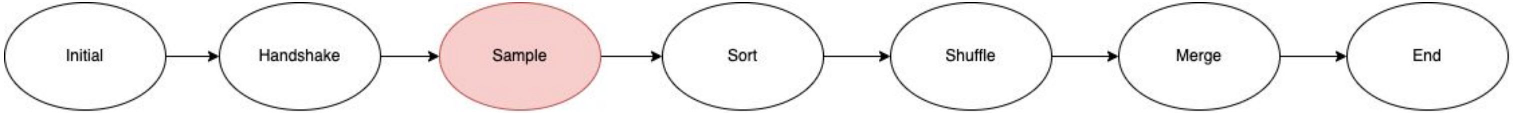
+ keyStart: Array[Byte]

+ keyEnd: Array[Byte]

+ serverPort: Int

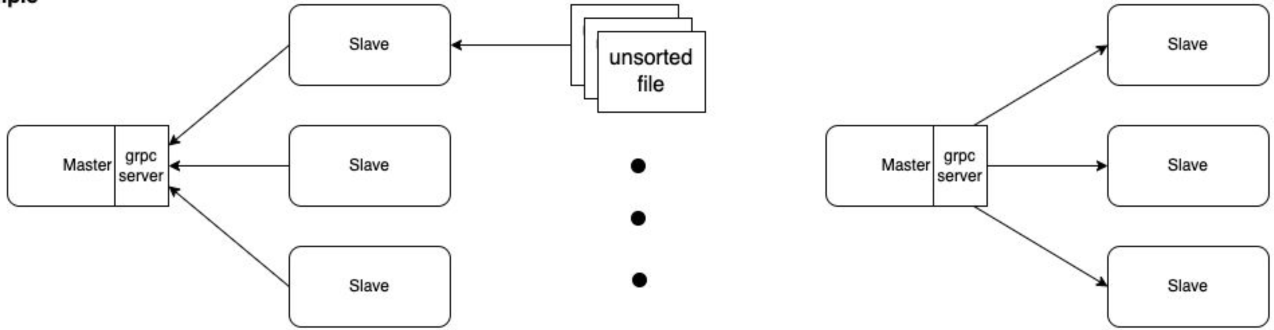
Master manages List[SlaveClient] for coordinating the system

Design



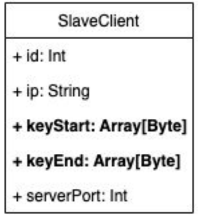
Sample

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



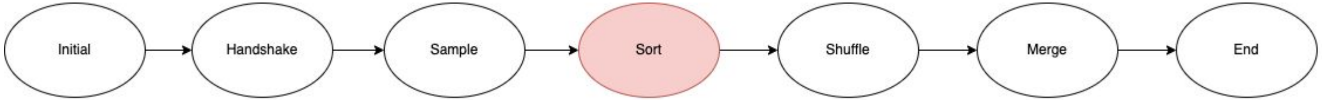
2. Send sampled data to master (grpc unary sendSampleData request)
Master waits until all slaves' request arrived.
Master analyze collected sample data and save key ranges(start, end) for each slave

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



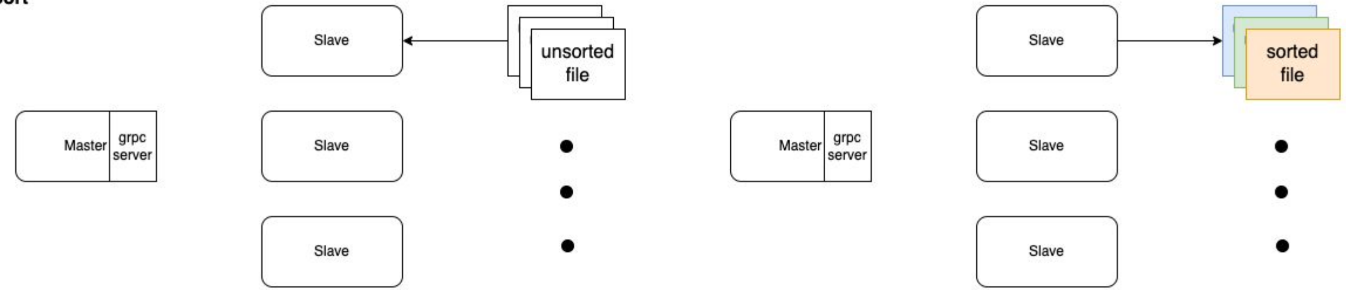
Master manages List[SlaveClient] for coordinating the system

Design



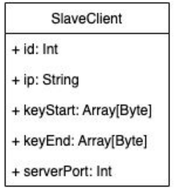
Sort

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>

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

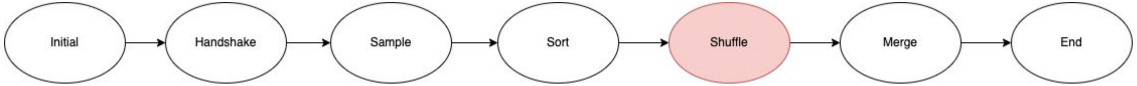


SlaveClient

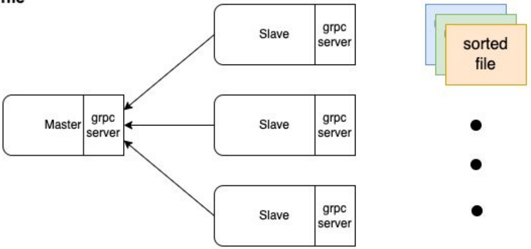
+ id: Int
+ ip: String
+ keyStart: Array[Byte]
+ keyEnd: Array[Byte]
+ serverPort: Int

Master manages List[SlaveClient] for coordinating the system

Design



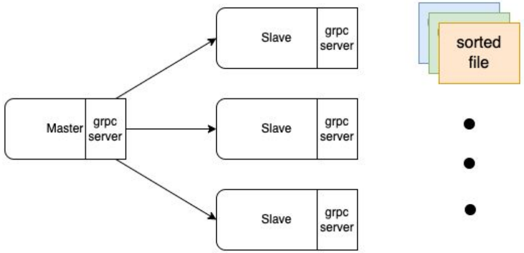
Shuffle



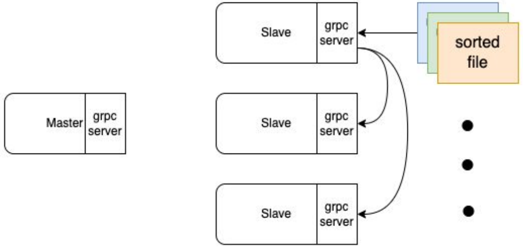
1. Slave starts grpc server for shuffling and send port info to master (grpc unary setServerPort request)
   Master waits until all slaves' request arrived.

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

Master manages List[SlaveClient] for coordinating the system

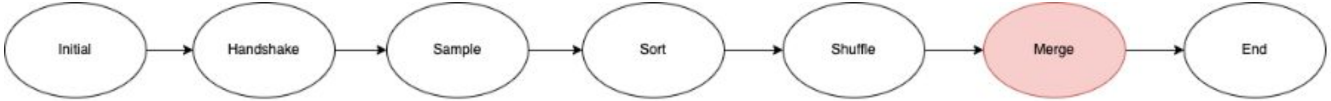


2. Send map<id, server_port> to every slave (grpc unary setServerPort response)

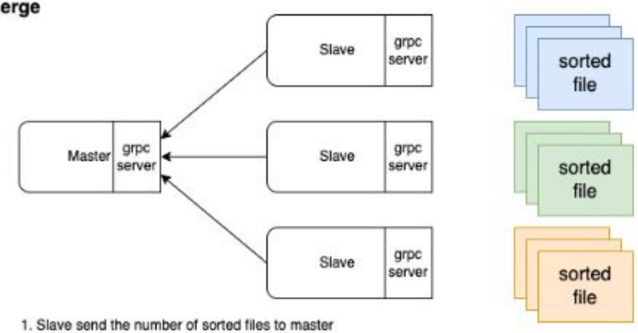


3. Send each sorted file to the appropriate slave's grpc server (grpc client-streaming sendSortedFiles request)
   Each slave waits until receiving files from all slaves.

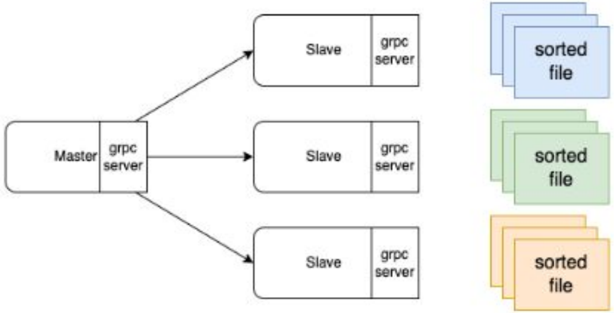
Design



Merge



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



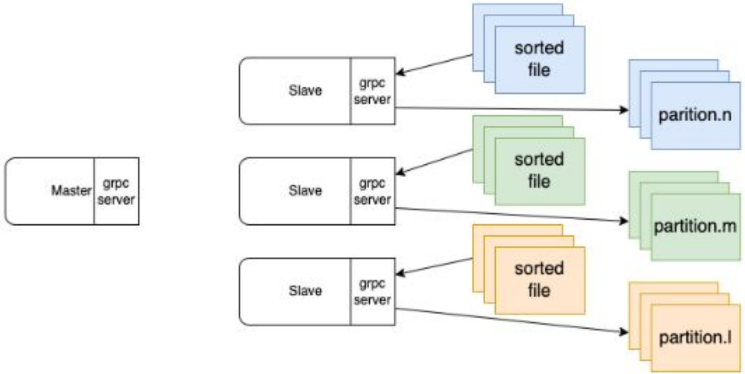
2. Send start index of partitioned file to every slave (grpc unary sendNumFile response)

SlaveClient

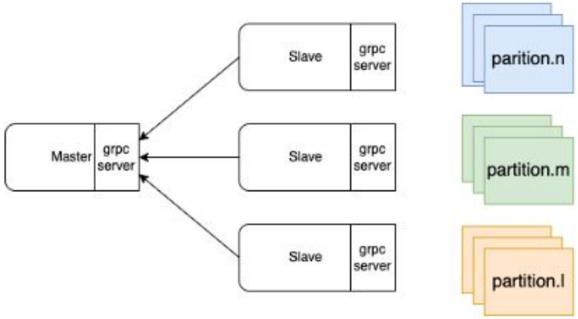
+ id: Int
+ ip: String
+ keyStart: Array[Byte]
+ keyEnd: Array[Byte]
+ serverPort: Int

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 connects 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 slaves**. (~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