Project progress

Team RED

김대희
박정훈
홍재영

Weekly Progress

Week 1

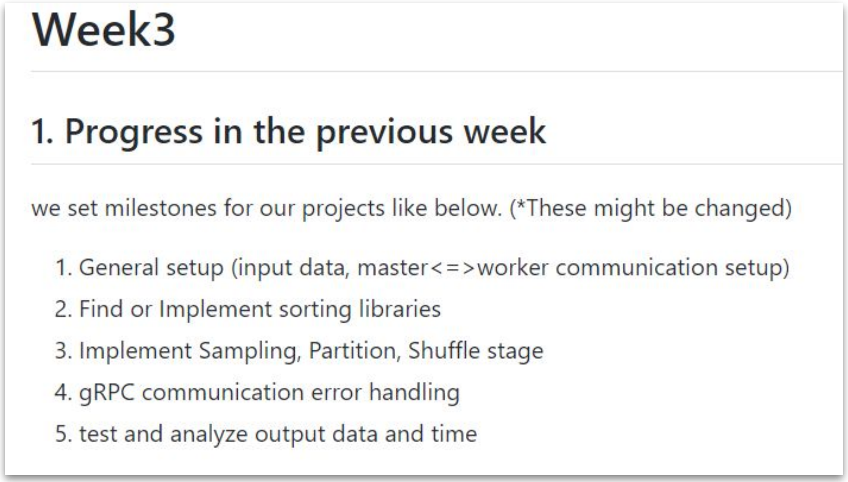
- Create project repo(https://github.com/jyhong1/332project)
- Git 을 통한 communication 방식 정리

Week 2 (Mid-term week)

- Fix environment setting - Ubuntu 18.04 + Intellij(Editor)
- Brief understanding & discussion about project
- Simple test of generating data by gensort

Week3

- Advance understanding of overall process
  - Construct server(Master/Worker) & Generate input data
  - Sampling, Sorting, Partitioning, Shuffle, Merge
- Manufactured process diagram
- Grpc/protobuf seminar
- Set milestones ⟶



Week3

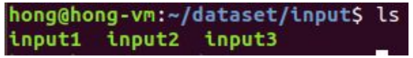
1. Progress in the previous week

we set milestones for our projects like below. (*These might be changed)

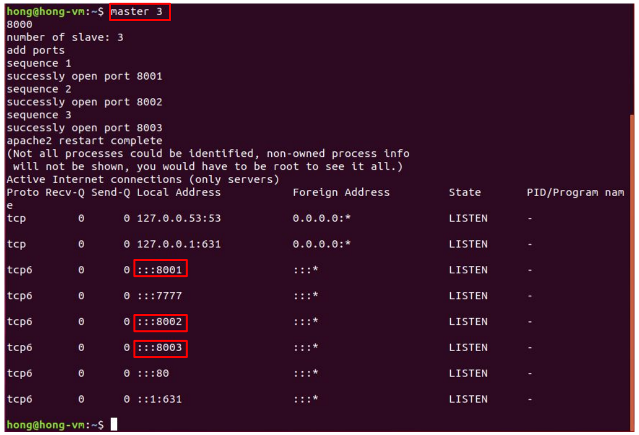
1. General setup (input data, master<=>worker communication setup)
2. Find or Implement sorting libraries
3. Implement Sampling, Partition, Shuffle stage
4. gRPC communication error handling
5. test and analyze output data and time

Week4

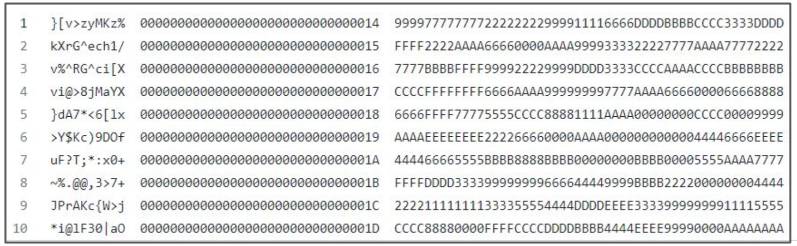
- Implement basic code example of ScalaPB and grpc
- Make shell script for test
  - port_test.sh - Open additional port with master command
  - generate_input.sh - Generate input and save into several partitions



```
hong@hong-vm:~/dataset/input$ ls
input1  input2  input3
```



```
hong@hong-vm:~$ master 3
8000
number of slave: 3
add ports
sequence 1
successly open port 8001
sequence 2
successly open port 8002
sequence 3
successly open port 8003
apache2 restart complete
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program nam
e
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -

tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -

tcp6       0      0 :::8001                 :::*                    LISTEN      -

tcp6       0      0 :::7777                 :::*                    LISTEN      -

tcp6       0      0 :::8002                 :::*                    LISTEN      -

tcp6       0      0 :::8003                 :::*                    LISTEN      -

tcp6       0      0 :::80                   :::*                    LISTEN      -

tcp6       0      0 ::1:631                 :::*                    LISTEN      -

hong@hong-vm:~$
```

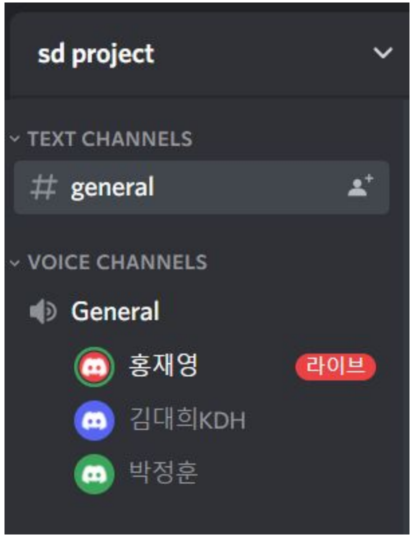


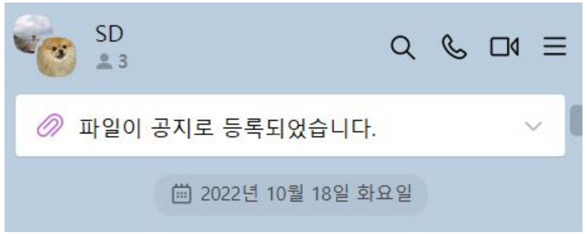
```
}[v>zyMKz%  00000000000000000000000000000014  99997777777722222222999911116666DDDDBBBBCCCC3333DDDD
kXrG^ech1/  00000000000000000000000000000015  FFFF2222AAAA66660000AAAA99993333222277777AAAA77772222
v%^RG^ci[X  00000000000000000000000000000016  7777BBBBFFFF99992222999DDDD3333CCCCAAAACCCCBBBBBBBB
vi@>8jMaYX  00000000000000000000000000000017  CCCCFFFFFFFF6666AAAA99999999777 7AAAA666600006666888
}dA7*<6[lx  00000000000000000000000000000018  6666FFFF77775555CCCC88881111AAAA00000000CCCC00009999
>Y$Kc)9DOf  00000000000000000000000000000019  AAAAEEEEEEEE22226666000 0AAAA000000000000 44446666EEEE
uF?T;*:x0+  0000000000000000000000000000001A  444466665555BBBB8888BBBB00000000BBBB00005555AAAA7777
~%.@@,3>7+  0000000000000000000000000000001B  FFFFDDDD3333999999996666444499999BBBB222200000004444
JPrAKc{W>j  0000000000000000000000000000001C  22221111111133335555444 4DDDDEEEE3333999999991111555 5
*i@lF30|aO  0000000000000000000000000000001D  CCCC88880000FFFFCCCCDDDDBBBB4444EEEE99990000AAAAAAAA
```

Logistics

- **Communication method**

1. Regular - Video meeting every Tuesday 10 pm
   - Summary of individual progress.
   - Set milestone & goal of the following week.
   - Synchronize between team members.
2. Irregular - Ask and share materials anytime (Discord, Kakaotalk)
3. Dev.
   - Divide parts in detail on code implement & research / Q&A session on regular meeting
   - Use Github's Issue-PR system on further progress.





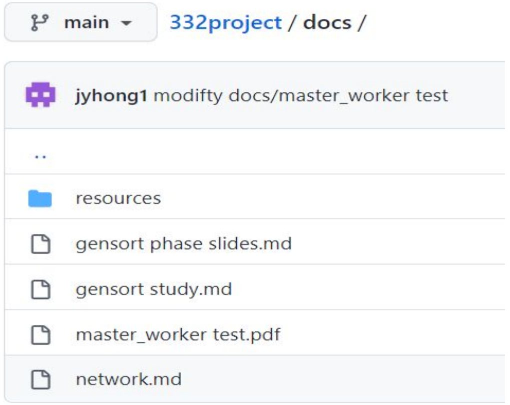
파일이 공지로 등록되었습니다.

2022년 10월 18일 화요일

Logistics

- **Documentation produced so far**

- 332project/docs/*.(pdf,md…etc)
- 332project/Weekly Progress report/
- Own design, research, and useful materials
- Private documentations which are not organized yet



main 332project / docs /

jyhong1 modifty docs/master_worker test

..

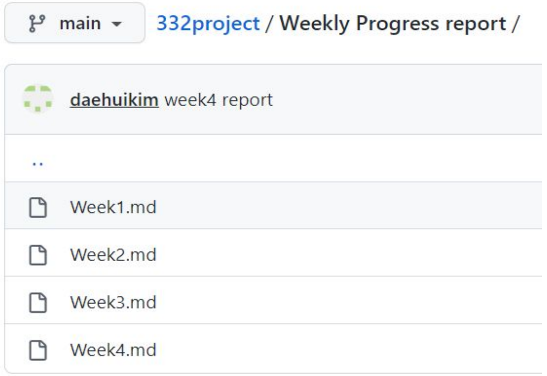
resources

gensort phase slides.md

gensort study.md

master_worker test.pdf

network.md



main 332project / Weekly Progress report /

daehuikim week4 report

..

Week1.md

Week2.md

Week3.md

Week4.md

Milestone

Completed

- Setup
- High-level Project Design
- gRPC communication example test

TO-DO (can be updated)

- Low-level Project Design ex) Class-method design, Msg system..
- Design specific methods & handlers in each phases.
- Implement each Phases (Sampling, Sorting, Partitioning, Shuffle, Merge)
- Set testing environment & testing

Challenges

1. How to divide the number of items equally to every workers? Which way of sampling can guarantee same divisions?
2. How to shuffle items parallel without interruption while using file system?
3. Trivial curiosities..

- The reason of disk sizes
- How to shuffle safely without overflow or losing data
- How to sort disks completely parallel and how to check that is correct
- error handlings on every phases…

=> We need to overcome such challenges for specific implementations!

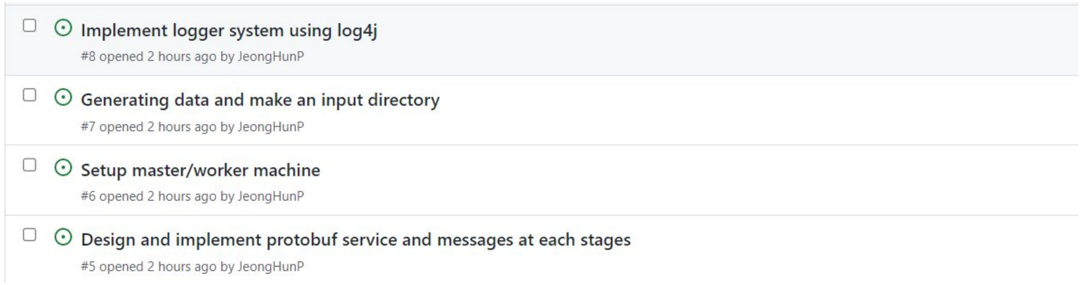
Milestone Changes

Week 1 - Week 4:

- Just set brief and big goals of next week
- Changed in detail every week but…

Week 5 ~

- Start to set big milestones with detailed issues
- Each issue will be resolved by the member assigned with PR



Implement logger system using log4j
#8 opened 2 hours ago by JeongHunP

Generating data and make an input directory
#7 opened 2 hours ago by JeongHunP

Setup master/worker machine
#6 opened 2 hours ago by JeongHunP

Design and implement protobuf service and messages at each stages
#5 opened 2 hours ago by JeongHunP

Environment

Programming Environment

Scala version: 2.12.17 (Latest 2.12.x maintenance)

Sbt version: 1.8.0 (Latest stable)

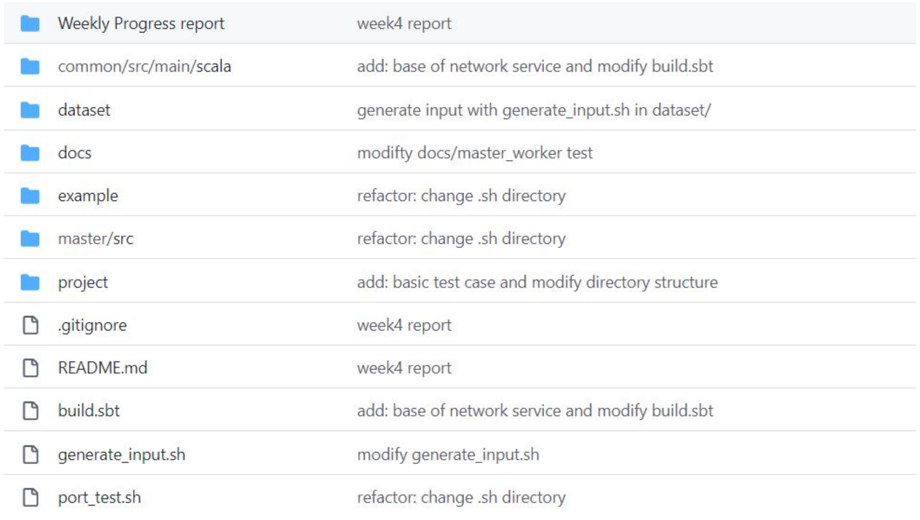
Ubuntu 18.04 (all of team members)

Libraries

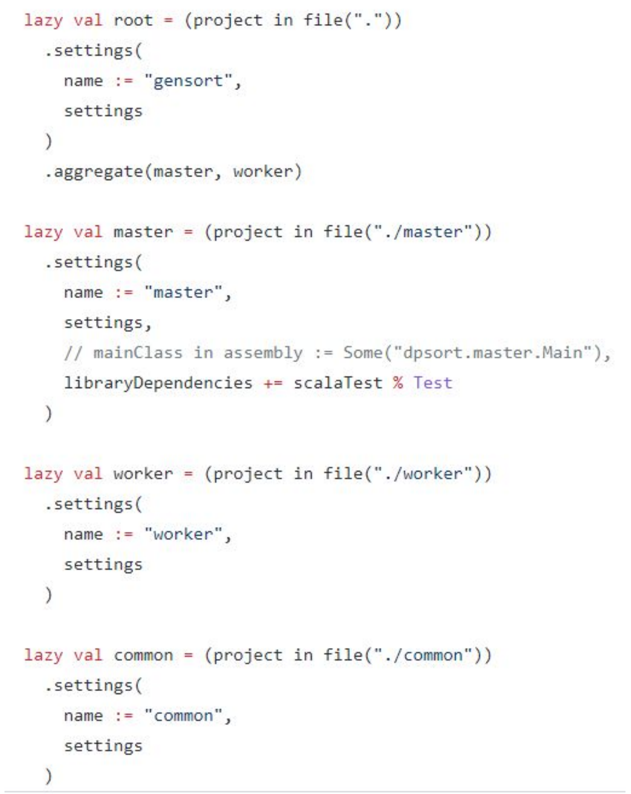
grpc - ScalaPB: automatically compiles and creates Scala classes for grpc.

Logger - log4j: info, debug, warn, error, fatal …

Structure of project



Weekly Progress report	week4 report
common/src/main/scala	add: base of network service and modify build.sbt
dataset	generate input with generate_input.sh in dataset/
docs	modifty docs/master_worker test
example	refactor: change .sh directory
master/src	refactor: change .sh directory
project	add: basic test case and modify directory structure
.gitignore	week4 report
README.md	week4 report
build.sbt	add: base of network service and modify build.sbt
generate_input.sh	modify generate_input.sh
port_test.sh	refactor: change .sh directory

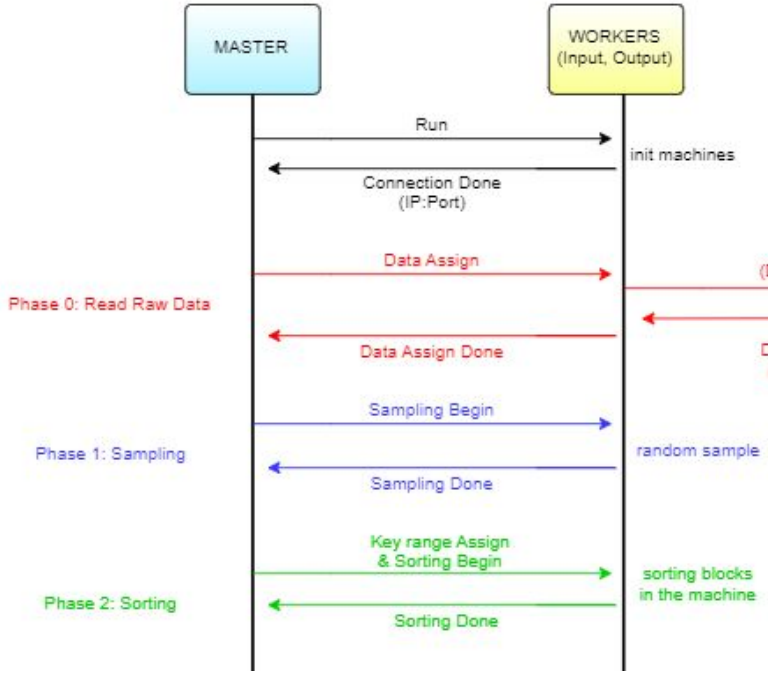


```
lazy val root = (project in file("."))
  .settings(
    name := "gensort",
    settings
  )
  .aggregate(master, worker)

lazy val master = (project in file("./master"))
  .settings(
    name := "master",
    settings,
    // mainClass in assembly := Some("dpsort.master.Main"),
    libraryDependencies += scalaTest % Test
  )

lazy val worker = (project in file("./worker"))
  .settings(
    name := "worker",
    settings
  )

lazy val common = (project in file("./common"))
  .settings(
    name := "common",
    settings
  )
```

Project (can be modified)

- master
- worker
- common(network, log …)
- examples

Design Overview (Connection to Phase 2)



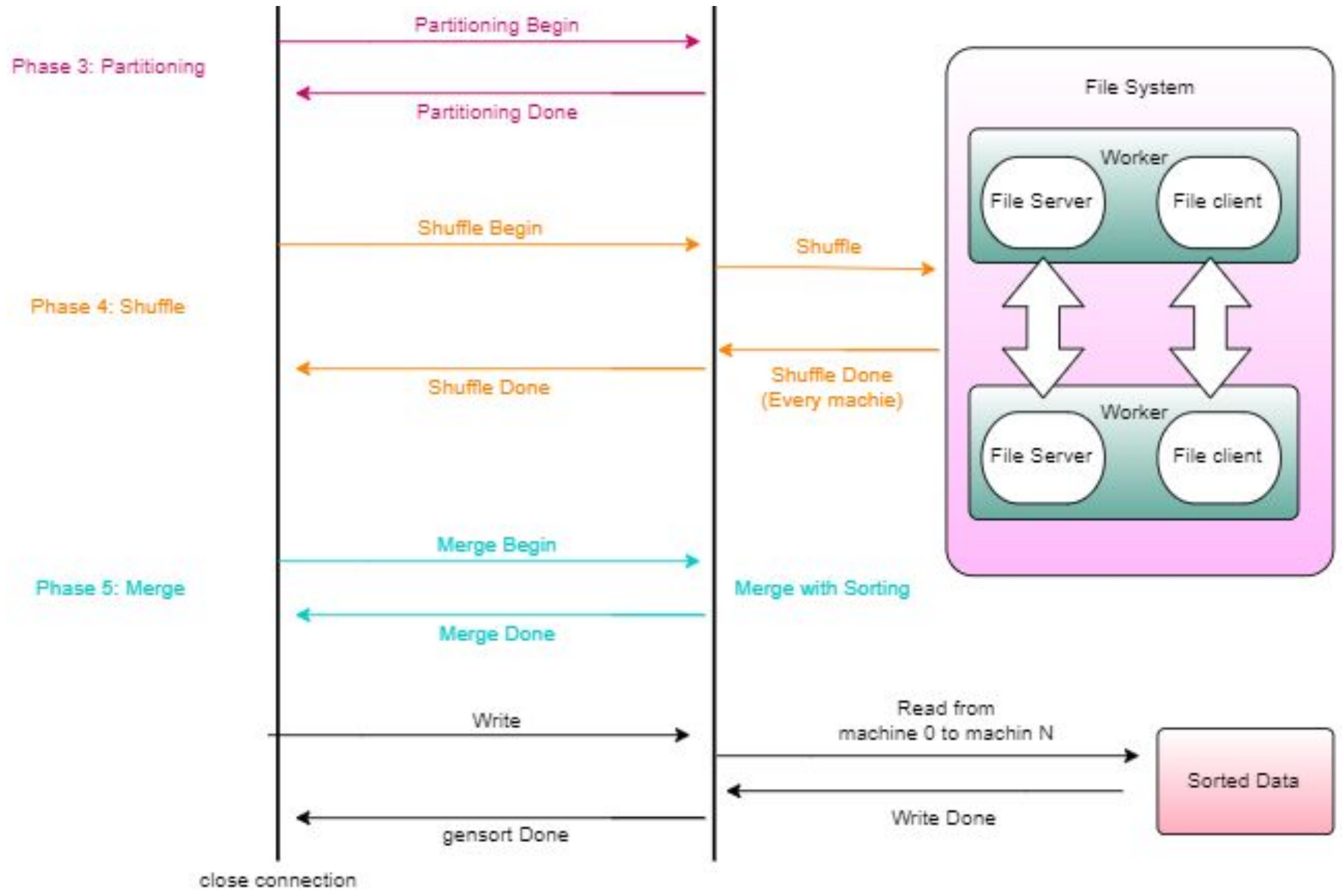
Wait until every workers connected to master

Divide and assign input data to every Workers

How to determine key ranges of each machine?
by median, by mean …

Which Library or sorting will be used to sort disks?
How to Sort every disk palallel? …

Design Overview (Phase 2 to gensort complete)



File system for processing shuffle parallel

Once every machine got proper partitions, close the File servers

Write data from machine 0 to N

Class design overview

Master

- Set key ranges to each partitions
- Synchronize workers by regulating each phase request and response
- In charge of giving right commands to workers and shuffling

Worker

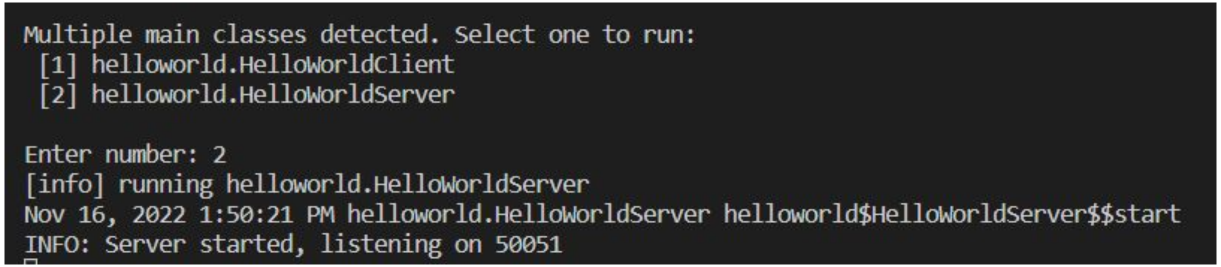
- Carry out each phase task by just Master’s command
- Functions: Sampling, Sorting, Partitioning, Merging …
- Dealing with data transfer between workers

Network

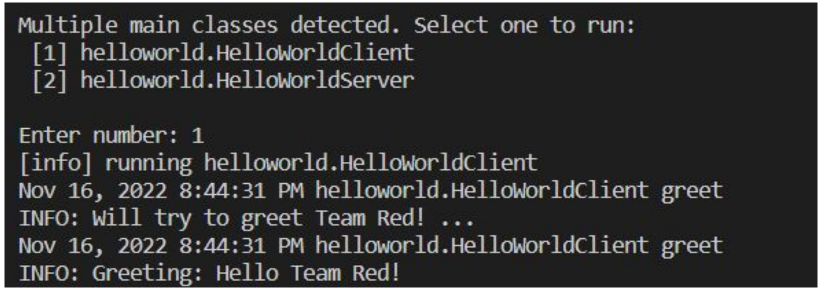
- Proper service/message design of each phase containing essential data
- Provide Server-Client connection and file system

Progress - Implementation

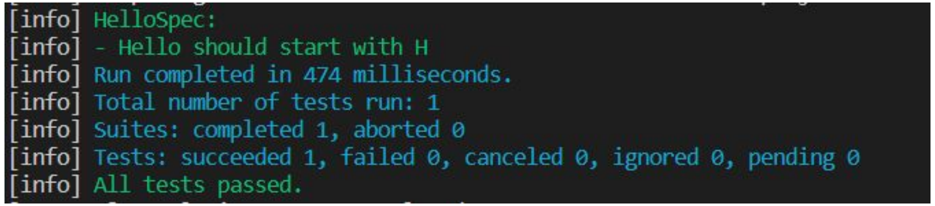
Simple example of Server-Client and test + Project setup (Directory, build.sbt …)



```
Multiple main classes detected. Select one to run:
 [1] helloworld.HelloWorldClient
 [2] helloworld.HelloWorldServer

Enter number: 2
[info] running helloworld.HelloWorldServer
Nov 16, 2022 1:50:21 PM helloworld.HelloWorldServer helloworld$HelloWorldServer$$start
INFO: Server started, listening on 50051
```



```
Multiple main classes detected. Select one to run:
 [1] helloworld.HelloWorldClient
 [2] helloworld.HelloWorldServer

Enter number: 1
[info] running helloworld.HelloWorldClient
Nov 16, 2022 8:44:31 PM helloworld.HelloWorldClient greet
INFO: Will try to greet Team Red! ...
Nov 16, 2022 8:44:31 PM helloworld.HelloWorldClient greet
INFO: Greeting: Hello Team Red!
```



```
[info] HelloSpec:
[info] - Hello should start with H
[info] Run completed in 474 milliseconds.
[info] Total number of tests run: 1
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
```

Self-review of last 4 weeks…

What team “RED” did well (+)

1. Dividing tasks equally to members
2. Never hesitate asking trivial things which leads to same sync

Important Lessons (-)

1. Don’t be optimistic
2. Set detailed milestones/issues for every single problems
3. Reduce communication overhead
4. Make documentations and note references