Distributed Sorting System

Project Final Report, cs434

20100463 Kisuk Park

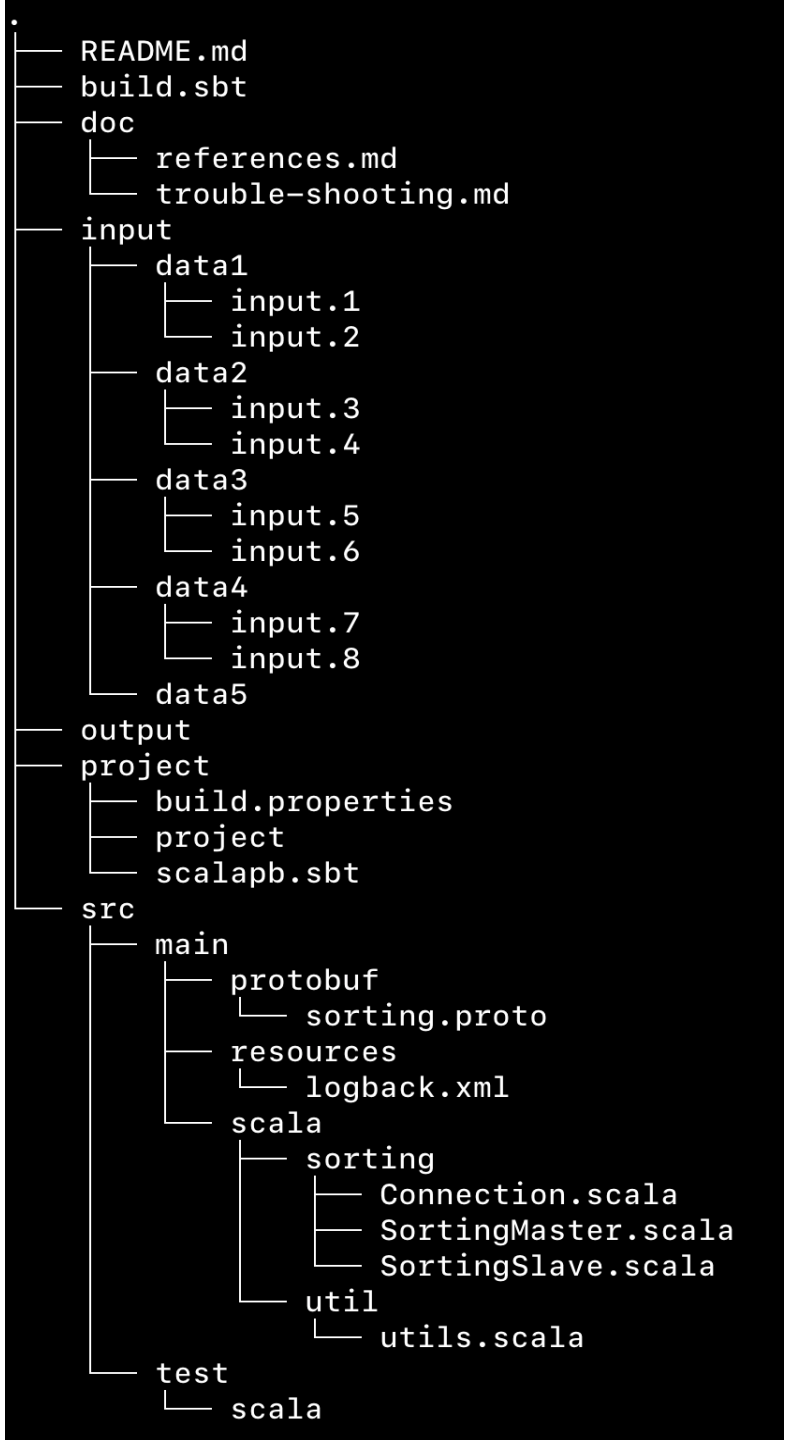
Contents

Tech Stacks

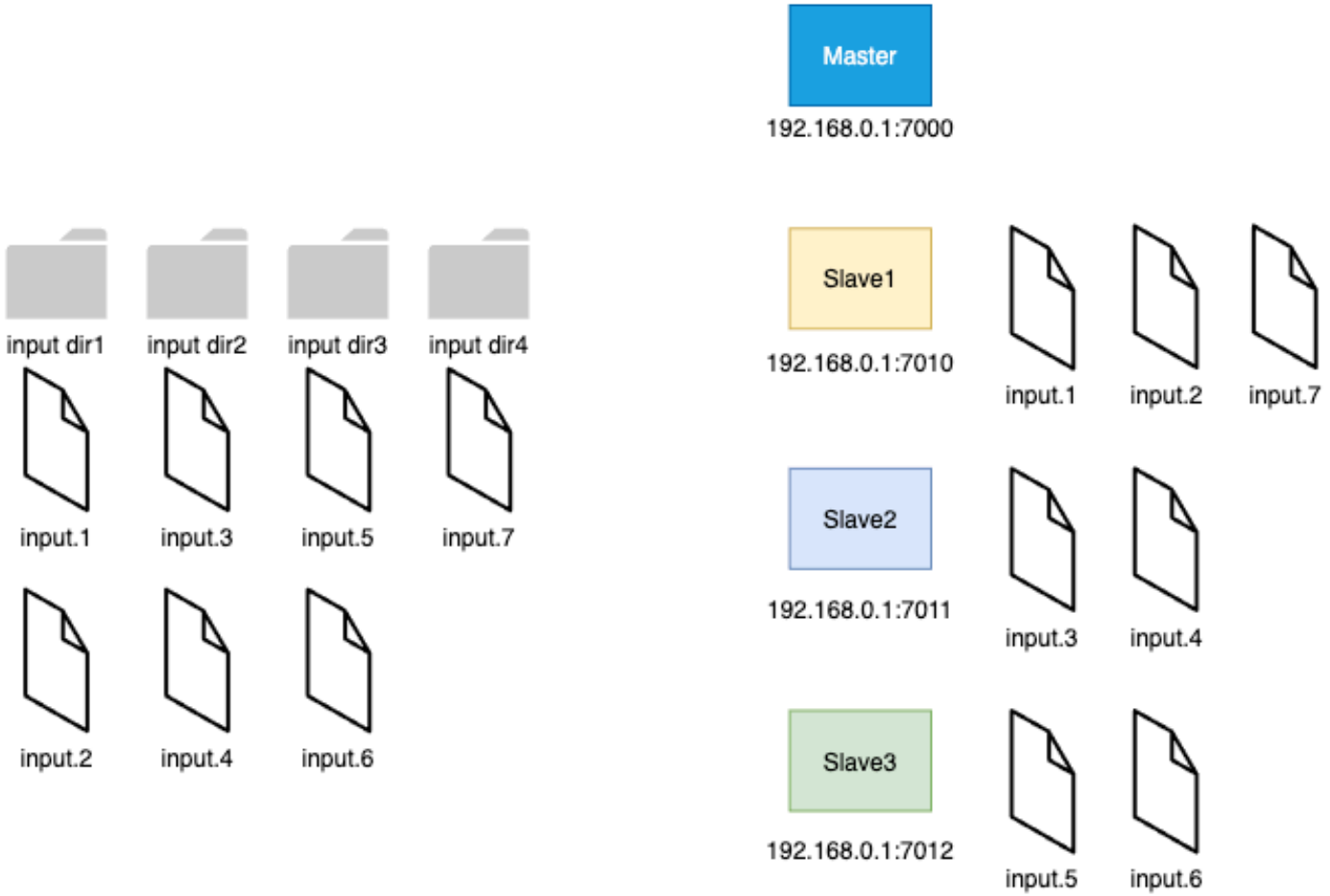
- ScalaPB: Protocol Buffers compile plugin for Scala
- Protocol Buffers(proto3): Google’s language-neutral, platform-neutral, extensible mechanism for serializing structured data
- gRPC: Open source high performance RPC framework
- Sbt: Scala build tool

Directory structure

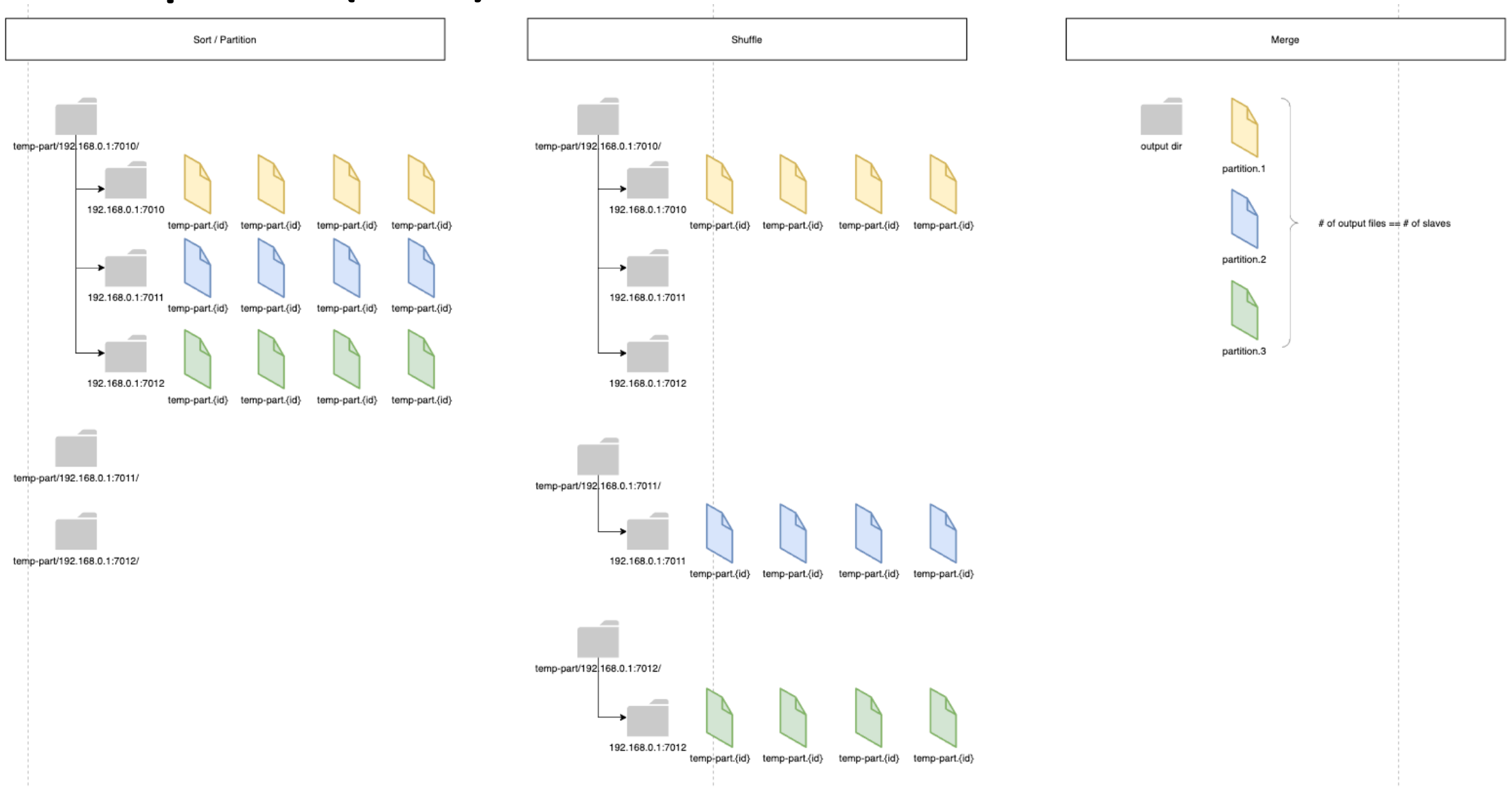
- input : Input data
- output: Merged output file will be stored here.
- src/main/protobuf/sorting.proto : protobuf declaration file
- src/main/scala : master/slave code & other codes



Blueprint (1/2)

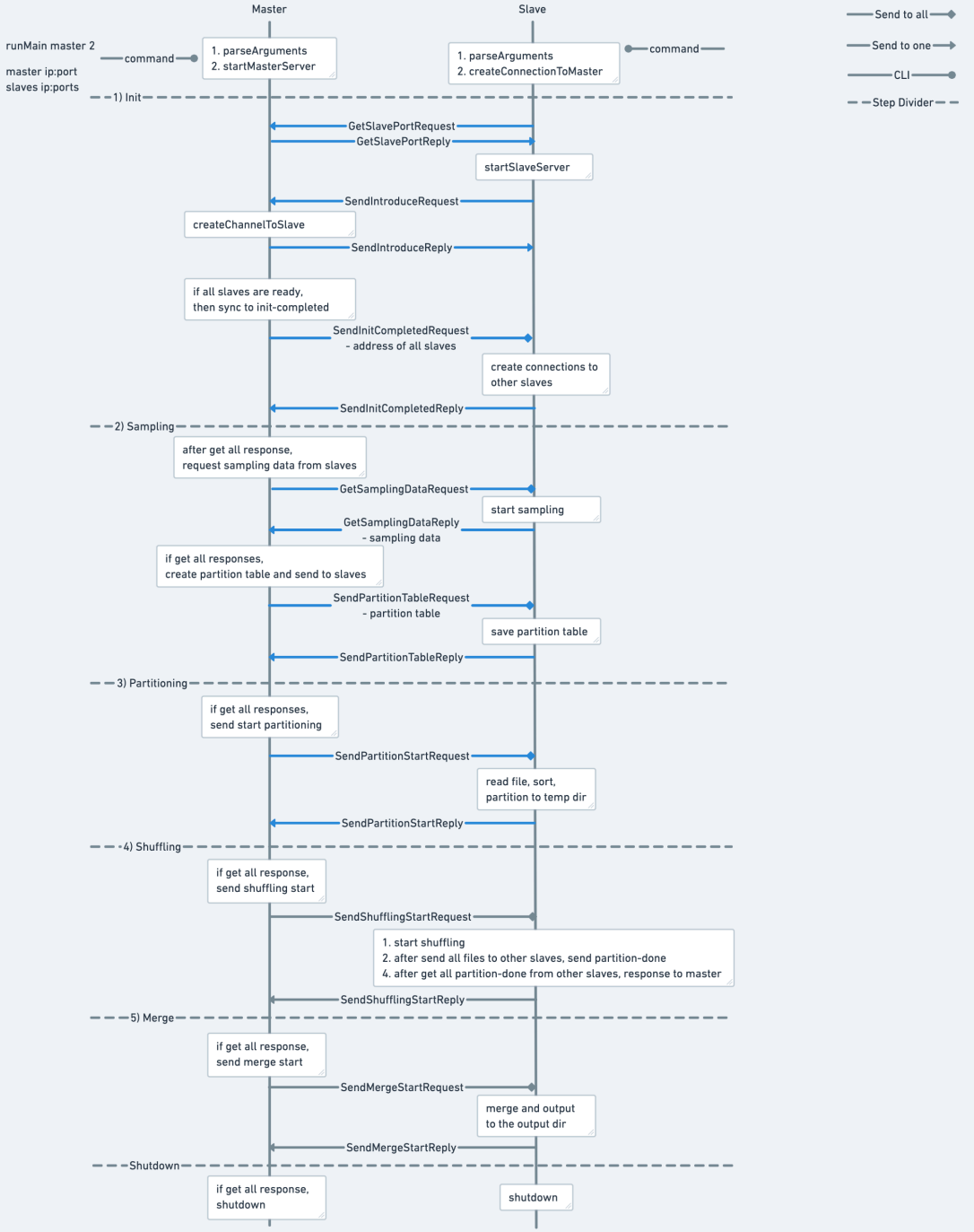


Blueprint (2/2)



Sequence Diagram

- Start with CLI
  - Run Master -> Run Each Slave
- Step1) Init
  - Run Master & Slave Servers
  - Establish connections b/w Master & Slaves
  - Sync slaves' address table
- Step2) Sampling
  - Request sampling data from slaves
  - Send partition key to slaves
- Step3) Partitioning
  - Notify partitioning start to slaves
  - For each slave, read file/sort/create temp dir/partitioning
- Step4) Shuffling
  - Notify shuffling start to slaves
  - Shuffling to each other
- Step5) Merge
  - Notifiy merge start to slaves
  - Merge



Demo

- Open sbt console
  - runMaster

```
sbt:cs434-project> runMain sorting.SortingMaster 3
```

  - runSlave

```
sbt:cs434-project> runMain sorting.SortingSlave 192.168.29.1
40:7001 -I ./input/data1 ./input/data2 ./input/data3 ./input
/data4 ./input/data5 -O ./output
```

- Master's log

```
- Master starting  192.168.29.140:7001
- Connect to slave 192.168.29.140:7011, waiting 2 more...
- Connect to slave 192.168.29.140:7012, waiting 1 more...
- Connect to slave 192.168.29.140:7013, waiting 0 more...
- Complete init step
- Get sampling data from slaves
- Generate partitioning table List(?PlL3:# AQ, _*^=uVJ+1X)
```

- Slaves' log

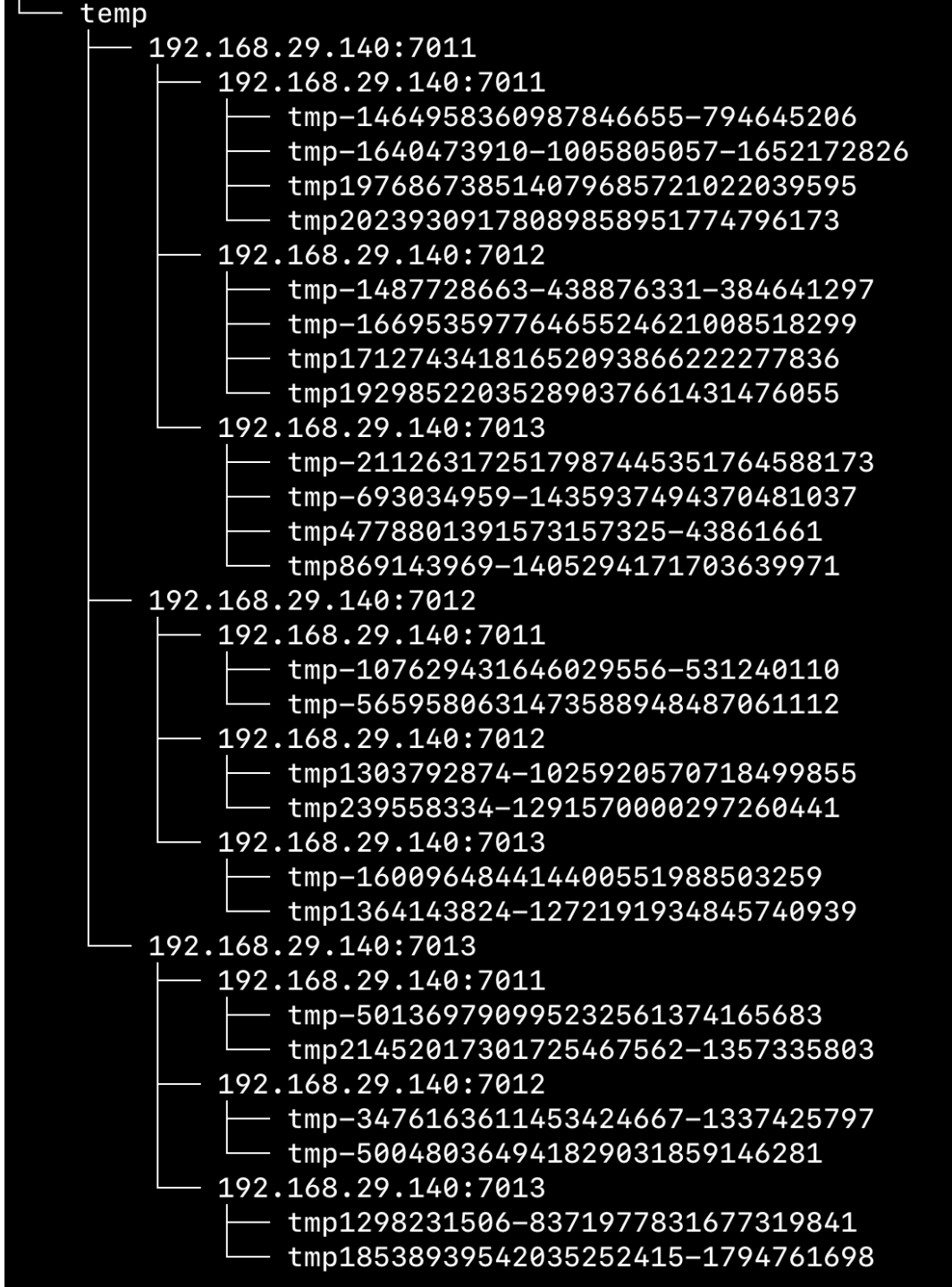
```
- Connect to master 192.168.29.140:7001
- Connect to other slave 7012
- Connect to other slave 7013
- Sampling from ./input/data1/input.1
- Receive partitioning table Vector(?PlL3:# AQ, _*^=uVJ+1X)
- data files List(./input/data1/input.1, ./input/data1/input.2, ./input/data4/input.7, ./input/data4/input.8)
- Start Reading/Sorting/Partitioning ./input/data1/input.1
- - Read total 10000 data
- - Sort data
- - Partitioning done
- Start Reading/Sorting/Partitioning ./input/data1/input.2
- - Read total 10000 data
- - Sort data
- - Partitioning done
- Start Reading/Sorting/Partitioning ./input/data4/input.7
- - Read total 10000 data
- - Sort data
- - Partitioning done
- Start Reading/Sorting/Partitioning ./input/data4/input.8
- - Read total 10000 data
- - Sort data
- - Partitioning done
```

```
- Connect to master 192.168.29.140:7001
- Connect to other slave 7011
- Connect to other slave 7013
- Sampling from ./input/data2/input.4
- Receive partitioning table Vector(?PlL3:# AQ, _*^=uVJ+1X)
- data files List(./input/data2/input.4, ./input/data2/input.3)
- Start Reading/Sorting/Partitioning ./input/data2/input.4
- - Read total 10000 data
- - Sort data
- - Partitioning done
- Start Reading/Sorting/Partitioning ./input/data2/input.3
- - Read total 10000 data
- - Sort data
- - Partitioning done
```

```
- Connect to master 192.168.29.140:7001
- Connect to other slave 7011
- Connect to other slave 7012
- Sampling from ./input/data3/input.6
- Receive partitioning table Vector(?PlL3:# AQ, _*^=uVJ+1X)
- data files List(./input/data3/input.6, ./input/data3/input.5)
- Start Reading/Sorting/Partitioning ./input/data3/input.6
- - Read total 10000 data
- - Sort data
- - Partitioning done
- Start Reading/Sorting/Partitioning ./input/data3/input.5
- - Read total 10000 data
- - Sort data
- - Partitioning done
```

Result

- Environment
  - 1 Master
  - 3 Slaves
  - Input 4 directories with 2 files each (total 8 files)
- Result
  - Slave 1 : run 2 dir & 4 files
  - Slave 2, 3 : run 1 dir & 2 files
  - Steps
    - **[v] Init Step**
    - **[v] Sampling**
    - **[v] Partitioning -> partition into ./temp directory**
    - [x] Shuffling
    - [x] Merge
  - Restrictions
    - Using blocking stub
    - # of input dir >= # of slaves



End of project

What I've learned

- (+) Documentation matters
  - todos, temporal notes, design doc helped to reduce confusions.
- (+) Small size commits
  - Easy to rollback, Less debugging time, Easy to understand current situation.
- (-) Underestimated schedule
- (-) Absence of test code
- (-) Need refactoring : Using lint, prettier Tool

Trouble Shooting

- 'Gensort' not working on MacOS
  - Run ubuntu machine via Docker and generate data files to mounted volume.
  - This should be announced before the project begin, if needed.

References

- Tutorial of ScalaPB's gRPC support
  - https://scalapb.github.io/docs/grpc
- Simple gRPC & Scala sample code on Github
  - https://github.com/xuwei-k/grpc-scala-sample
- Official Page for required tech stacks
  - Protocol buffer: https://developers.google.com/protocol-buffers
  - gRPC: https://grpc.io/
- https://scalapb.github.io/docs/grpc/
- https://github.com/xuwei-k/grpc-scala-sample/blob/master/grpc-scala/src/main/scala/io/grpc/examples/helloworld/HelloWorldServer.scala