# Project Presentation

Team Grey

# Experiment

On VM Clusters
use 4 workers
user 64 input blocks per worker

# Correctness

## 1. Does the master start? ✅

**Master** Waiting for 4 clients

```
[info] running cs332.master.Master 4
Dec 15, 2022 4:21:25 AM cs332.master.Master cs332$master$Mas
ter$$start
INFO: Server has client Count of 4
Dec 15, 2022 4:21:25 AM cs332.master.Master cs332$master$Mas
ter$$start
INFO: Server started, listening on 50030
2.2.2.101:50030
```

Server address

# Correctness

## 2. Does each worker connect to the master? ✅

**Worker0**

```
[info] running cs332.worker.Worker 2.2.2.101:50030 -I /input
-data/ -O /output-data/
Dec 15, 2022 4:22:30 AM cs332.worker.Worker register
INFO: REGISTER : success as worker 0
```

**Worker1**

```
[info] running cs332.worker.Worker 2.2.2.101:50030 -I /input
-data/ -O /output-data/
Dec 15, 2022 4:22:30 AM cs332.worker.Worker register
INFO: REGISTER : success as worker 1
```

**Worker2**

```
[info] running cs332.worker.Worker 2.2.2.101:50030 -I /input
-data/ -O /output-data/
Dec 15, 2022 4:22:30 AM cs332.worker.Worker register
INFO: REGISTER : success as worker 2
```

**Worker3**

```
[info] running cs332.worker.Worker 2.2.2.101:50030 -I /inpu
t-data/ -O /output-data/
Dec 15, 2022 4:22:30 AM cs332.worker.Worker register
INFO: REGISTER : success as worker 3
```

**Master**

```
INFO: Server started, listening on 50030
2.2.2.101:50030
2.2.2.103, 2.2.2.104, 2.2.2.105, 2.2.2.106
```

After connections from all workers,

Registered Workers addresses Logged

# Correctness

## 3. Does the master collect sample data? ✅

**All Workers**

```
Dec 15, 2022 4:25:19 AM cs332.worker.Worker sample
INFO: SAMPLE : sampling is done
Dec 15, 2022 4:25:19 AM cs332.worker.Worker sendSample
INFO: Will try to send file  ...
Dec 15, 2022 4:25:19 AM cs332.worker.Worker sendSample
INFO: SAMPLE : sending sample is done
```

**Master**

```
grey@vm01:~/332project/master/sampled$ ls -l
total 16968
-rw-rw-r-- 1 grey grey 2323200 Dec 15 04:25 sample.0
-rw-rw-r-- 1 grey grey 2323200 Dec 15 04:25 sample.1
-rw-rw-r-- 1 grey grey 2323200 Dec 15 04:25 sample.2
-rw-rw-r-- 1 grey grey 2323200 Dec 15 04:25 sample.3
-rw-rw-r-- 1 grey grey 8069120 Dec 15 04:25 sortedSamples
grey@vm01:~/332project/master/sampled$ ls -l
total 9076
-rw-rw-r-- 1 grey grey 9292800 Dec 15 04:25 sortedSamples
```

# Correctness

## 4. Does the master return distribution keys back to workers? ✅

**Master**

```
Dec 15, 2022 4:25:27 AM cs332.master.Master$SorterImpl$$anon
$1 onCompleted
INFO: PIVOT : setting pivot is done
```

**Worker0**

```
workerPivots 0 : Worker(2.2.2.103:50060,Some(Pivot(   "{Y>:#%
`,7nqOBx`!5 ,UnknownFieldSet(Map())),UnknownFieldSet(Map())
)
Dec 15, 2022 4:25:27 AM cs332.worker.Worker$$anon$1 onNext
INFO: File upload status :: SUCCESS
```

**Worker1**

```
workerPivots 1 : Worker(2.2.2.104:50060,Some(Pivot(7nv7&wc{"
*,ON&CC[^Pup,UnknownFieldSet(Map())),UnknownFieldSet(Map())
)
Dec 15, 2022 4:25:27 AM cs332.worker.Worker$$anon$1 onNext
INFO: File upload status :: SUCCESS
```

**Worker2**

```
workerPivots 2 : Worker(2.2.2.105:50060,Some(Pivot(ON'orgo{3
+,g4)C~gc@a7,UnknownFieldSet(Map())),UnknownFieldSet(Map())
)
Dec 15, 2022 4:25:27 AM cs332.worker.Worker$$anon$1 onNext
INFO: File upload status :: SUCCESS
```

**Worker3**

```
workerPivots 3 : Worker(2.2.2.106:50060,Some(Pivot(g4*g/#WG
jC,~~{wcPMY]q,UnknownFieldSet(Map())),UnknownFieldSet(Map(
)))
Dec 15, 2022 4:25:27 AM cs332.worker.Worker$$anon$1 onNext
INFO: File upload status :: SUCCESS
```

**Worker.scala** 173 line

```
    def sendSample(): Unit = {
        logger.info("Will try to send file "  + " ...")
        val streamObserver: StreamObserver[PivotRequest] = n
ewStub.getWorkerPivots(
            new StreamObserver[PivotResponse] {
                override def onNext(response: PivotResponse)
: Unit =  {
                    workerPivots = response.workerPivots.toL
ist

                    workerCount = workerPivots.length
                    // print("WORKERPIVOTS " + workerPivots
)
                    System.out.println("workerPivots " + wor
kerOrder.toString + " : " + workerPivots(workerOrder) )
                    sendSampleLatch.countDown
                    assert(workerPivots != null)
```

# Correctness

## 5. Do workers pass intermediate data between each other? ☐

**Worker 0** ← **Worker 3**

```
Dec 15, 2022 4:26:53 AM worker.server.WorkerFileServer worke
r$server$WorkerFileServer$$start
INFO: Worker file server started, listening on 8080
```

Worker 3 Shuffled Directory

```
grey@vm06:~/332project/worker/shuffled3$ ls
0.partition30.0  0.partition30.4  partition33.2
0.partition30.1  0.partition30.5  partition33.3
0.partition30.2  partition33.0    partition33.4
0.partition30.3  partition33.1    partition33.5
```

*partition[A][B].partitioned Order*
partition sent from A to B

```
INFO: REGISTER : try to connecting...
Dec 15, 2022 4:26:54 AM worker.client.WorkerFileClient regi
ster
INFO: REGISTER : file server registration success
Dec 15, 2022 4:26:55 AM worker.client.WorkerFileClient shuf
fle
INFO: Will try to get files from other workers
Dec 15, 2022 4:26:55 AM worker.client.WorkerFileClient shuf
fle
INFO: responses<iterator>
Dec 15, 2022 4:26:55 AM worker.client.WorkerFileClient shuf
fle
INFO: Writing file...
Dec 15, 2022 4:26:55 AM worker.client.WorkerFileClient shuf
fle
INFO: Writing file...
Dec 15, 2022 4:26:55 AM worker.client.WorkerFileClient shuf
fle
```

⋮

```
Dec 15, 2022 4:26:57 AM worker.client.WorkerFileClient shuf
fle
INFO: Writing file...
Dec 15, 2022 4:26:57 AM cs332.worker.Worker shuffle
INFO: SHUFFLE : connect to WorkerFileServer 0 as WorkerFile
Client 3
```

**Stuck** in the middle of Shuffling Process

# Correctness

## 6. Does the master print a sequence of workers? ✅

Right after the workers connected,

**Master**

```
INFO: Server started, listening on 50030
2.2.2.101:50030
2.2.2.103, 2.2.2.104, 2.2.2.105, 2.2.2.106
```

Worker Order: 0 → 1 → 2 → 3

# Correctness

**7. Is the output sorted in each worker?** ☐

**8. # of records in the input == # of records in the output?** ☐

**Prior Stuck** in the partition process…

# Project Management

# Milestone By Week

**Week 1** Determine Direction and Find Things to Study ✅

**Week 2** Midterm ✅

**Week 3** Body Design(1) + Testing Environment Setting (1) ✅

**Week 4** Body Design(2) + Testing Environment Setting (2) ✅

**Week 5** Start Coding + Fix Some Design ✅

**Week 6** Finish Coding + Design Unit Test ✅

~~**Week 7** Debugging + Testing~~ Finish Coding

~~**Week 8** Finish~~ Finish Coding (due effect…😅)

# Goal of The Project for Individual Member

**Heewoo Lee**

- basic functionalitiess of gRPC server - client interaction
  - start server
  - register client
  - file transfer
  - asynchronous response that waits all clients to send request
- code merge
  - unify and modify argument types if needed
  - manage temporary files made within process
  - fix logics and errors that are found when codes are merged
- util functions
  - handy helper functions which are needed for both master and worker
- logging and assertions

# Goal of The Project for Individual Member

**Jeongwon Choi**

- environment
  - implement script for generate input file (using gensort)
  - create Dockerfile
- master function
  - Implement function to set pivots to workers
- worker function
  - Implement sampling function
- shuffling network
  - Implement workerFileClient, workerFileServer
  - Implement register and shuffling logic

# Goal of The Project for Individual Member

**Mingyeol Kim**

- master function
  - Implement validation function: **validationWorkerOrdering**
- worker function
  - Implement external merge sort function: **externalMergeSort**
  - Implement Partitioning function: **partitionByPivot**
  - Implement mergeDone-related function: **extractMinMaxKey**
- enhancing performance
  - To be reasonably fast
  - To be memory-safe

    → Several details described in the later slide

# Design Changed

## Deletion of partition information exchange functionality

**Before:**
each workers send information of partitioned file (name and number) to master
**After:**
master do not get partitioned file name. Instead return list of address of workers that each worker should require file

## client - server implementation

**Before:**
each worker and server was started just by calling main function of each object
**After:**

- worker continuous try to connect with server using while loop  (polling)
    - worker tries to register itself until any response returns
    - worker able to register even when server starts after client

# Implementation for better Performance

## Memory-safe

**: Necessary to process a large file which cannot fit in a memory**

- Problem: BufferedSource.getLines()(.toList)
  fetch whole contents of the file to one variable → NOT memory-safe
- Solved: Implement a function **getPartialLines** +) tailrec
  - read *{numLines}* lines from the File at once
  - Target: **partitionByPivot, sampling**

```
@tailrec
def getPartialLines(numLines: Int, lines: scala.collection.immutable.Queue[String], scanner: Scanner): scala.collection.immutable.Queue[String] = {
    if (numLines == 0 || !scanner.hasNextLine()) lines
    else getPartialLines(numLines - 1, lines :+ scanner.nextLine(), scanner)
}
```

# Implementation for better Performance

## Execution Speed

**: Optimized to handle a large amount of data in a reasonable time**

- **partitionByPivot**
  write a large number of lines to partition files
  - **Avoid O($n^2$) by StringBuffer**
    n = # of lines to be written at once to one partition file
- When append operation(:+) is needed (n = sequence length)
  - List : O(n)
  - scala.collection.immutable.**Queue : O(1)**
  - Target: **partitionByPivot, sampling**

# What you learned from the project

## Assertions and logging

Clearly, using assertion

- reduced the effort of debugging
- ensure process is satisfying intermediate conditions to be fulfilled

## Solid intermediate data management before cooperation

Matching data types of function parameters and return types were not enough

- fixing where data to be read and to be stored while merging took significant amount of time
- would first define file directory tree for intermediate datas and files

## Efficiency of working together in same place (office)

Offline meeting is more efficient that online meeting

- It was more efficient to do own task individually in same place
- We could make significant progress and communicate well in offline

## Importance of divide tasks considering the dependency

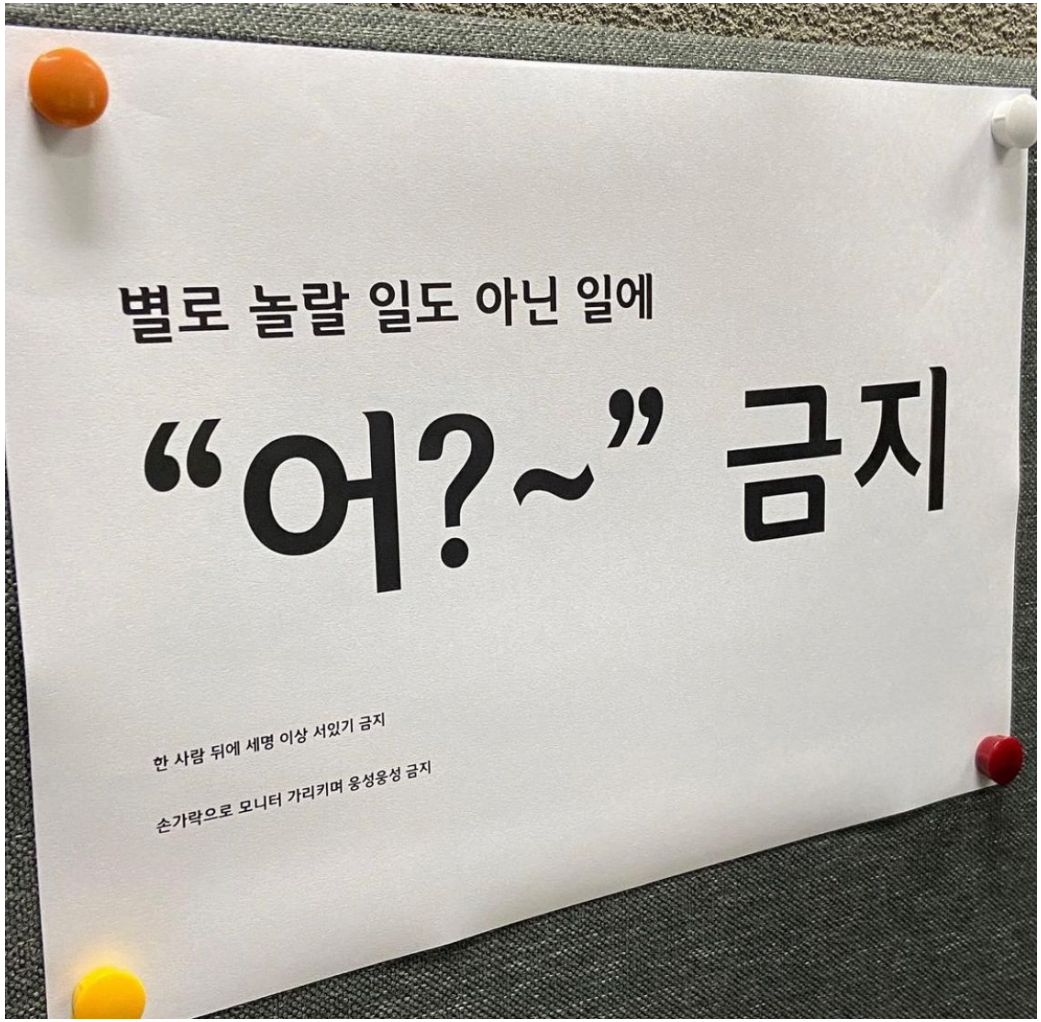
Diving tasks also quite important process

- We use semi-democratic method
- First, we divide tasks that don’t have any dependency each other
- Then each member has sudo authority for that task

# Needs for Testing Automation

Time consuming to manually input the same test commands multiple times

- Automation implemented in some degree through sbt setting
- Higher degree of automation may reduce the cost of time further
- Time is Gold: Always being ready to be in a testable environment


별로 놀랄 일도 아닌 일에
“어?~” 금지
한 사람 뒤에 세명 이상 서있기 금지
손가락으로 모니터 가리키며 웅성웅성 금지