Project Progress

Software Design Methods

Weekly progress review

- Week 1: Project setup
- Week 2: Configure gRPC + Protobuf
- Week 3: Simple Architecture
- Week 4: Detailed Software Architecture
- Week 5: Setup the main skeleton of Worker

Our weekly progress will be documented here:

Week 1:

Goal:

- (Together): Setting up the project

Week 2:

Goal:

- (Vincent): Learn the basic of gRPC + Protobuf
- (Adam): Find some other things to implement in the project, maybe a parser or anything else useful

Week 3:

Goal:

- An archiecture of the project has been made. It can be found in 332projectdocumentsarchitecture 1.0.pdf
- For the next week, we will try to implement smaller things from the architecture and make it work:

Week 4:

Goal:

- A more detailed architecture of the worker has been meticulously designed. Throughout next week, we will start the coding process.

Week 5:

Goal:

-

Communication

- We have meetings every Sunday where we:
  - Reflect on our weekly progress
  - Discuss what we should work on until the next meeting
  - Delegate tasks

Documentation

- Document progress in a README file after meeting
- Helps us track our work history and let us set goals for the next week
- We can still use the Git history to see what we have changed over time

```
PS C:\Users\pixel\OneDrive\Dokumenter\GitHub\332project> git log
commit c99eccbcb65134e73252136b3a98150d5c895730 (HEAD -> main, origin/main, origin/HEAD)
Author: 4sela <adawa21@student.sdu.dk>
Date:   Sun Nov 16 22:38:44 2025 +0900

    spelling error

commit 5f8210de634fbc0f7efbe2ce34322fe6547c18e6
Author: 4sela <adawa21@student.sdu.dk>
Date:   Sun Nov 16 22:37:28 2025 +0900

    updated readme

commit a5291315a6eae2e2255b29f9bd9a6f331b2c4bae
Author: Gousse-Gousse <149082743+Gousse-Gousse@users.noreply.github.com>
Date:   Sun Nov 16 22:18:31 2025 +0900

    add the basic squeleton of the project

commit 544fc70135264796ca0863119e5277cb3d226613
Author: 4sela <adawa21@student.sdu.dk>
Date:   Sun Nov 16 20:45:39 2025 +0900

    push

commit a2dd31ba2db7229b78070244fe4c4a9848dd1fee
Author: 4sela <adawa21@student.sdu.dk>
Date:   Sun Nov 9 22:42:02 2025 +0900

    Readme update
```

Design of the System

- For the system design, we have two models:
  - The basic
  - The detailed system architecture

Basic Design

- General design of both Worker and Master:
  - System requirements
  - Timeline of the run-time
  - Description of functions

- Gives us a broad idea of the system without the intricacies
- Slightly outdated and lacks nuance

Architechture

Worker

Communicating Handler
- Worker ⇔ Master
- Worker ⇔ Worker

Memory Handler

Partitioning Unit

Sort Unit | Sort Unit

Merge Unit

Command handler/ Main worker Unit...

Master

Sample

Network / Worker manager

State Manager

User Main Interface

History / Logger / Transaction history

Master

Sample

Network / Worker manager

State Manager

User Main Interface

History / Logger / Transaction history

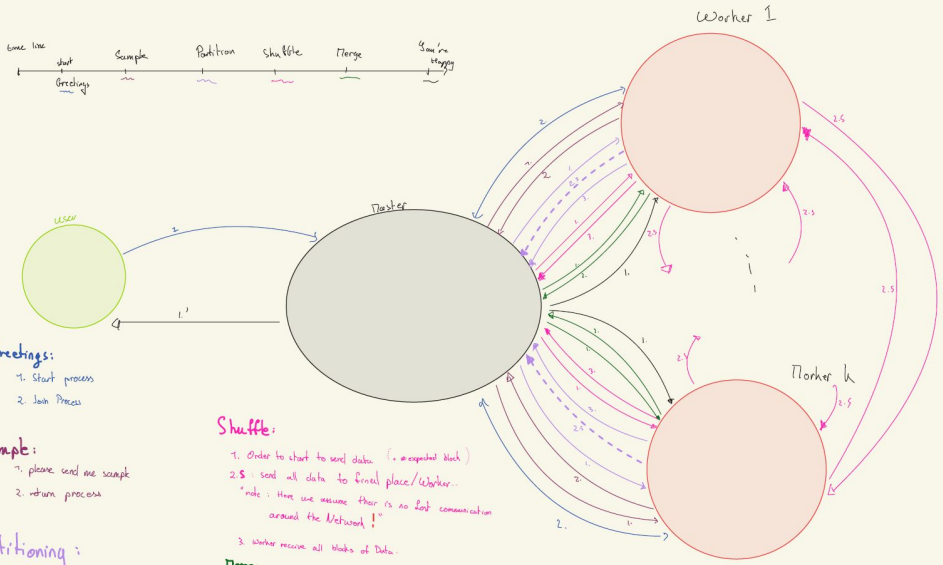
Sampler
- the sample receive
- Sort the sample
- Compute the range

Network Manager
- the ip of all the actor
- send the request
- receive workers information
- Communicate with the other part of the worker

State Manager
- current state
- previous state
- worker state ...
- manage the worker depending of the current state
- change of state (update) when the condition are fulfilled!

User Main Interface
- handler of the request of the user
- Send message to the user
- start / kill the process
  - ↳ interact with process

Logger / History of Translation:
"keep in memory every transact"
→ to faster for fast recovery



Greetings:
1. start process
2. Join Process

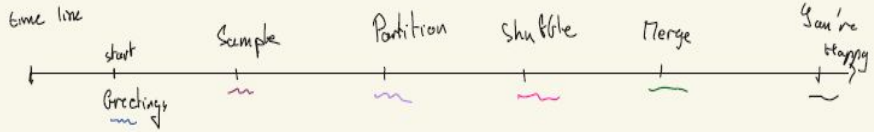
Sample:
1. please send me sample
2. return process

Partitioning:
1. Send the range of partitioning + please start to partition you're stuff!
2. $* Send Meta-data each time you're <: partition ; 2. sort ; store in temp file > a new block of data.
   (*$: many blocks will be created)
3. Worker has finish to partition the memory

Shuffle:
1. Order to start to send data (+ # expected block)
2. $: send all data to final place/Worker...
   "note : Here we assume that is no fast communication around the Network !"
3. Worker receive all blocks of Data.

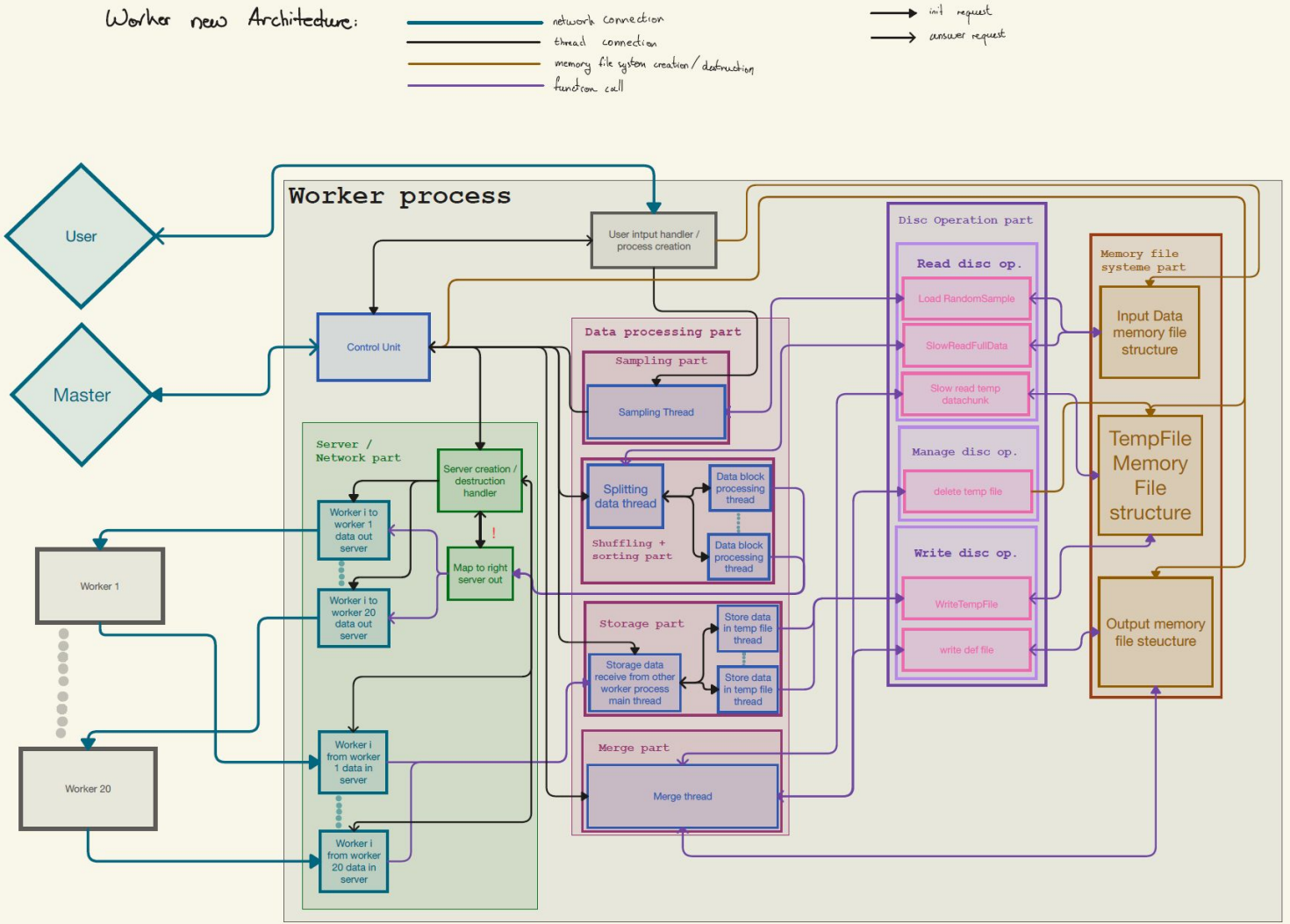
Merge:
1. Start Merging
2. Worker finish to merge

End:
1. & 1' : end of the process

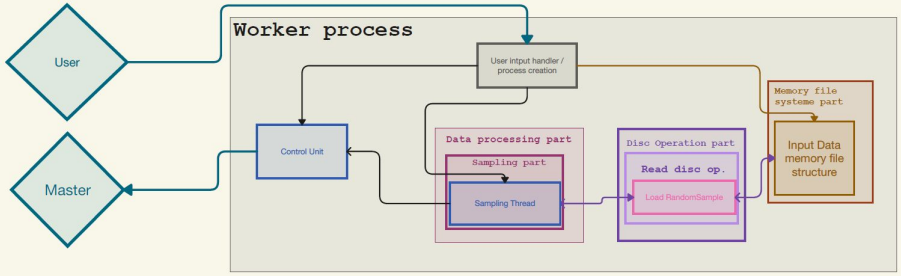


Detailed system architecture

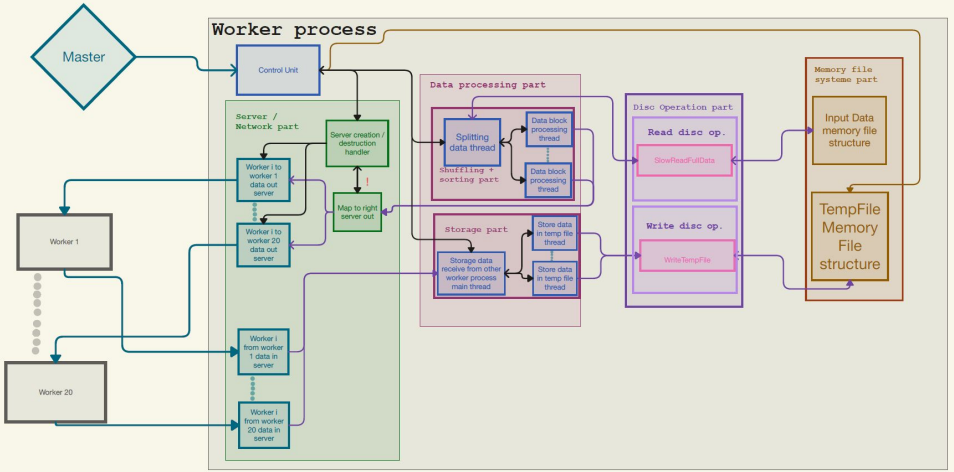
- Contains more precise details and nuances of the Architecture
- Only Worker system has been designed so far (aim to finish Master this week)


Worker new Architecture:
network connection
thread connection
memory file system creation / destruction
function call
init request
answer request
User
Master
Worker process
User intput handler / process creation
Control Unit
Server / Network part
Server creation / destruction handler
Worker i to worker 1 data out server
Worker i to worker 20 data out server
Map to right server out
Worker 1
Worker 20
Worker i from worker 1 data in server
Worker i from worker 20 data in server
Data processing part
Sampling part
Sampling Thread
Splitting data thread
Data block processing thread
Shuffling + sorting part
Data block processing thread
Storage part
Store data in temp file thread
Storage data receive from other worker process main thread
Store data in temp file thread
Merge part
Merge thread
Disc Operation part
Read disc op.
Load RandomSample
SlowReadFullData
Slow read temp datachunk
Manage disc op.
delete temp file
Write disc op.
WriteTempFile
write def file
Memory file systeme part
Input Data memory file structure
TempFile Memory File structure
Output memory file steucture

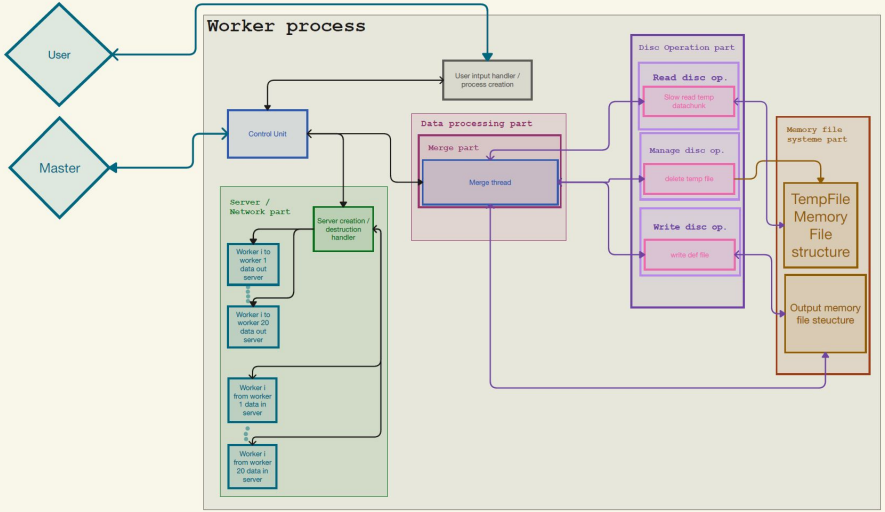
I Lauch process & join network



II partionning, data block processing & shuffelling

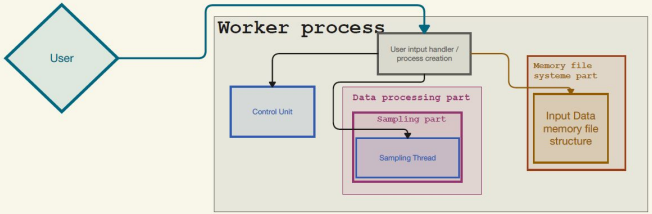


III Merge & End process



I Lauch process & join network

I.a launch worker, start the 1st threads



Specifications

- Using the latest versions of Java and Scala (openjdk-25, 3.7)
- Reason:
  - (Main) Performance: VM optimization
  - Access to modern tools
- Protobuf to communicate our data
- Master hasn’t been fully designed yet so we have no solution for logging yet

Milestone

Completed	Goals to reach
Basic Architecture	Full Master Architecture (+ Skeleton)
Full Worker Architecture	Networking/Sockets
Implement Worker Skeleton	Logging