Final Presentation

Team Yellow
Adam Awad, Vincent Tesnière

Commands for running Master/Workers

Used two commands, one for Master, one for Workers

```
sbt "runMain Master.UserInterface 5"
```

```
[info] running Master.UserInterface 5
============================================================
  DISTRIBUTED SORTING - MASTER NODE
============================================================
  Expected workers: 5
  Port: 30040
============================================================

Master server started on 169.254.7.35:30040

Waiting for 5 workers to connect...

Worker 1 registered at 169.254.7.35:49975
Received 10 samples from Worker 1
Worker 2 registered at 169.254.7.35:49984
Received 10 samples from Worker 2
Worker 3 registered at 169.254.7.35:49988
Received 10 samples from Worker 3
Worker 4 registered at 169.254.7.35:49990
Received 10 samples from Worker 4
Worker 5 registered at 169.254.7.35:49992
Transition: INIT -> SAMPLING
All workers connected!
```

```
sbt "runMain Worker.UserInterface localhost:30040 -I input1 -O output1"
sbt "runMain Worker.UserInterface localhost:30040 -I input2 -O output2"
sbt "runMain Worker.UserInterface localhost:30040 -I input3 -O output3"
sbt "runMain Worker.UserInterface localhost:30040 -I input4 -O output4"
sbt "runMain Worker.UserInterface localhost:30040 -I input5 -O output5"
```

```
[info] running Worker.UserInterface localhost:30040 -I input1 -O output1
============================================================
  DISTRIBUTED SORTING - WORKER NODE
============================================================
  Master: localhost:30040
  Input directories: input1
  Output directory: output1
============================================================

Starting Worker...
Worker server started on 169.254.7.35:49984
Connecting to Master at localhost:30040...
Successfully connected! Assigned Worker ID: 2
```

Commands for running Master/Workers (extra)

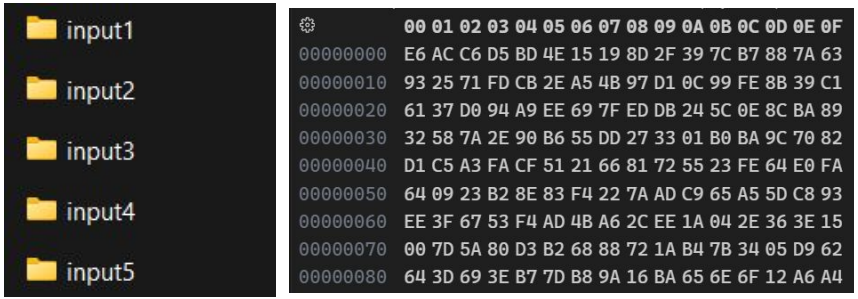
Original solution was inefficient, so we made a PowerShell script to automate the commands

```powershell
Write-Host "Creating output directories..." -ForegroundColor Cyan
for ($i = 1; $i -le 5; $i++) {
    New-Item -ItemType Directory -Path "output$i" -Force | Out-Null
}
Write-Host "Created 5 output directories" -ForegroundColor Green

Write-Host "Starting Master server..." -ForegroundColor Cyan
Start-Process powershell -ArgumentList "-NoExit", "-Command", "sbt 'runMain Master.UserInterface 5'"

Write-Host "Waiting for Master to start..." -ForegroundColor Yellow
Start-Sleep -Seconds 5

Write-Host "Starting 5 workers..." -ForegroundColor Cyan
for ($i = 1; $i -le 5; $i++) {
    $cmd = "sbt 'runMain Worker.UserInterface localhost:30040 -I input$i -O output$i'"
    Start-Process powershell -ArgumentList "-NoExit", "-Command", $cmd
    Write-Host "  Started Worker $i" -ForegroundColor Gray
    Start-Sleep -Seconds 2
}

Write-Host "`n All workers started!" -ForegroundColor Green
Write-Host "Watch the Master window for progress..." -ForegroundColor Yellow
```

C# script

To speed up the process, we added a macro written in C#

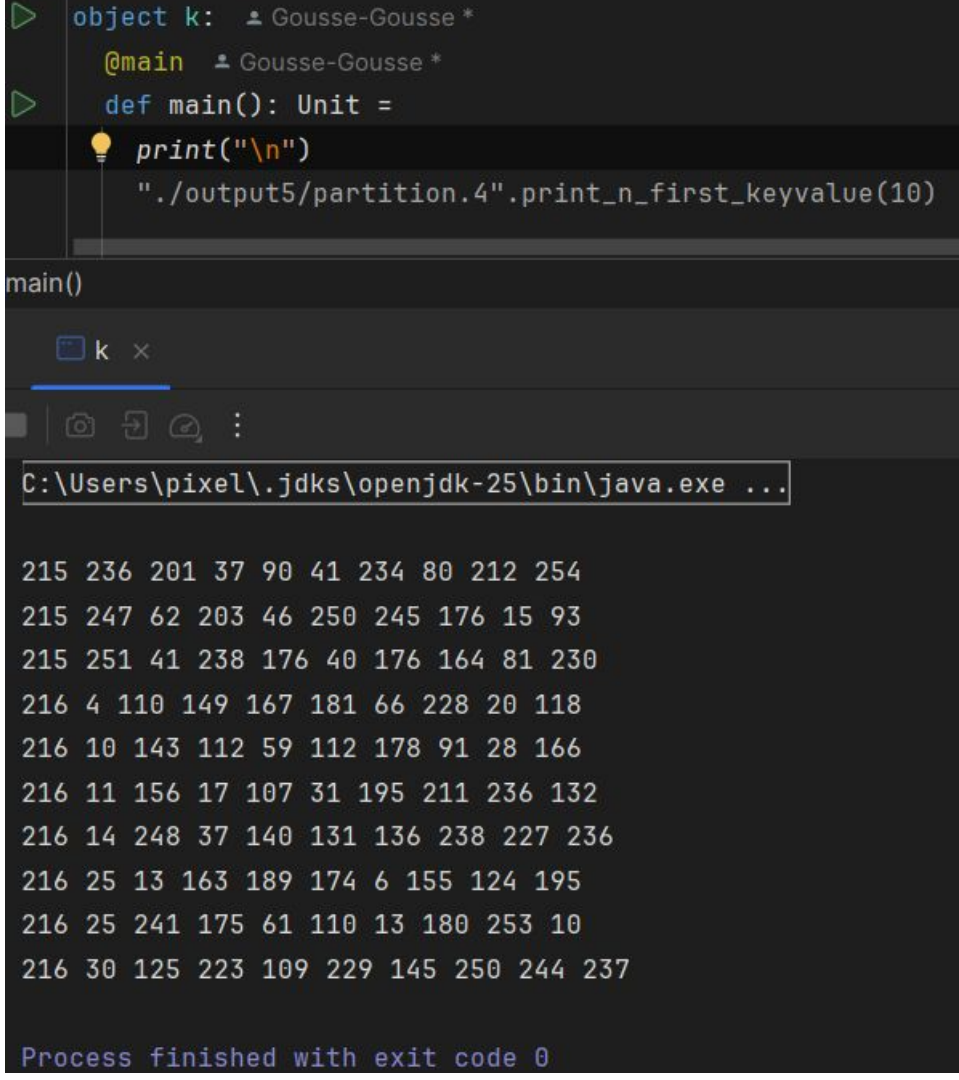
Used to generate datasets to test with




```
using System;
using System.IO;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Generating test data for 20 workers...");
        for (int worker = 1; worker <= 5; worker++)
        {
            GenerateRecords($"input{worker}/data", 10000);
        }

        Console.WriteLine("Done! Generated 20 files with 1MB each.");
    }

    static void GenerateRecords(string path, int numRecords)
    {
        var random = new Random();
        var directory = Path.GetDirectoryName(path);
        if (!string.IsNullOrEmpty(directory))
            Directory.CreateDirectory(directory);

        using var file = File.Create(path);

        for (int i = 0; i < numRecords; i++)
        {
            var record = new byte[100];
            for (int j = 0; j < 10; j++)
                record[j] = (byte)random.Next(0, 256);
            for (int j = 10; j < 100; j++)
                record[j] = (byte)random.Next(0, 256);
            file.Write(record, 0, 100);
        }
        Console.WriteLine($"  Created {path} ({numRecords * 100:N0} bytes)");
    }
}

```

It works with our samples

Example one of our output file



DataDefinition.scala I : Extension syntax

We used extension syntax to create some library features for every useful types

```scala
19  type Path = String
20  type JavaPath = java.nio.file.Path
129 extension (path: JavaPath)  Gousse-Gousse
130   def isDirectory: Boolean =  Gousse-Gousse
131     java.nio.file.Files.isDirectory(path)
132
133   def isNormalFile: Boolean =  Gousse-Gousse
134     java.nio.file.Files.isRegularFile(path)
135
136   def exist: Boolean =  Gousse-Gousse
137     java.nio.file.Files.exists(path)
```

DataDefinition.scala II : Companion Objects

To make our lives easier, we implemented these functions in the companion objects too

```scala
object IO_OPERATION:  Gousse-Gousse +1 *
  def read(path: Path, read_full_file: Boolean = true): Option[DataArray | DataStream] = {
    if (read_full_file) read_full(path) else read_slow(path)
  }

  def read_full(path: Path): Option[DataArray] = {  Gousse-Gousse +1
    val file = new BufferedInputStream(new FileInputStream(path))
    try {
      val return_value: DataArray = file.readAllBytes()
      Some(return_value)
    } catch {
      case _: Throwable => Option.empty
    } finally {
      file.close()
    }
  }
```

Protobuf

We used Protobuf for Master-Worker communication

Set it up with common.proto

Generated Java classes with protoc

Ways to improve further thinking

- Hard to have a good idea of how perfect the system is
- We cannot try our system over the cluster before the scheduled timeslice
- If we have more clusters: Use heap for merging data
- If we have more data: Be careful to use progressive reads

Final result with cluster

Why it failed, because we didn’t use Java 8