

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)

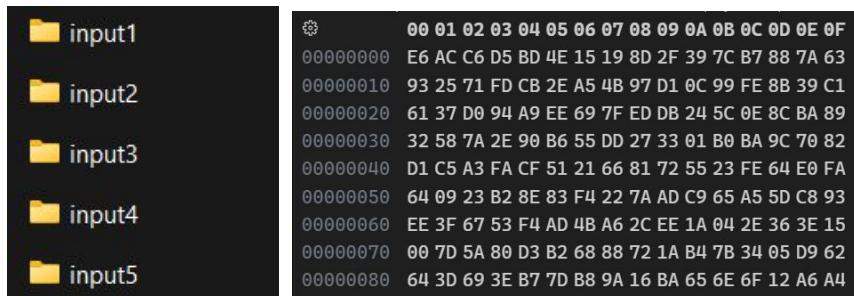
Original solution was inefficient,
so we made a PowerShell
script to automate the
commands

```
1 Write-Host "Creating output directories..." -ForegroundColor Cyan
2 for ($i = 1; $i -le 5; $i++) {
3     New-Item -ItemType Directory -Path "output$i" -Force | Out-Null
4 }
5 Write-Host "Created 5 output directories" -ForegroundColor Green
6
7 Write-Host "Starting Master server..." -ForegroundColor Cyan
8 Start-Process powershell -ArgumentList "-NoExit", "-Command", "sbt 'runMain Master.UserInterface 5'"
9
10 Write-Host "Waiting for Master to start..." -ForegroundColor Yellow
11 Start-Sleep -Seconds 5
12
13 Write-Host "Starting 5 workers..." -ForegroundColor Cyan
14 for ($i = 1; $i -le 5; $i++) {
15     $cmd = "sbt 'runMain Worker.UserInterface localhost:30040 -I input$i -O output$i'"
16     Start-Process powershell -ArgumentList "-NoExit", "-Command", $cmd
17     Write-Host "Started Worker $i" -ForegroundColor Gray
18     Start-Sleep -Seconds 2
19 }
20
21 Write-Host "`n All workers started!" -ForegroundColor Green
22 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



```
1 using System;
2 using System.IO;
3
4 class Program
5 {
6     static void Main(string[] args)
7     {
8         Console.WriteLine("Generating test data for 20 workers...");
9         for (int worker = 1; worker <= 5; worker++)
10         {
11             GenerateRecords($"input{worker}/data", 10000);
12         }
13
14         Console.WriteLine("Done! Generated 20 files with 1MB each.");
15     }
16
17     static void GenerateRecords(string path, int numRecords)
18     {
19         var random = new Random();
20         var directory = Path.GetDirectoryName(path);
21         if (!string.IsNullOrEmpty(directory))
22             Directory.CreateDirectory(directory);
23
24         using var file = File.Create(path);
25
26         for (int i = 0; i < numRecords; i++)
27         {
28             var record = new byte[100];
29             for (int j = 0; j < 10; j++)
30                 record[j] = (byte)random.Next(0, 256);
31             for (int j = 10; j < 100; j++)
32                 record[j] = (byte)random.Next(0, 256);
33             file.Write(record, 0, 100);
34         }
35         Console.WriteLine($"Created {path} ({numRecords * 100:N0} bytes)");
36     }
37 }
38
```

It works with our samples

Example one of our output file

```
object k: Gousse-Gousse *
@main Gousse-Gousse *
def main(): Unit =
    print("\n")
    "./output5/partition.4".print_n_first_keyvalue(10)

main()
```

k x

C:\Users\pixel\.jdk\openjdk-25\bin\java.exe ...

```
215 236 201 37 90 41 234 80 212 254
215 247 62 203 46 250 245 176 15 93
215 251 41 238 176 40 176 164 81 230
216 4 110 149 167 181 66 228 20 118
216 10 143 112 59 112 178 91 28 166
216 11 156 17 107 31 195 211 236 132
216 14 248 37 140 131 136 238 227 236
216 25 13 163 189 174 6 155 124 195
216 25 241 175 61 110 13 180 253 10
216 30 125 223 109 229 145 250 244 237

Process finished with exit code 0
```

DataDefinition.scala I : Extension syntax

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

```
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

```
160 object IO_OPERATION: ⚠ Gousse-Gousse +1 *
161   def read(path: Path, read_full_file: Boolean = true): Option[DataArray | DataStream] = {
162     | if (read_full_file) read_full(path) else read_slow(path)
163   }
164
165   def read_full(path: Path): Option[DataArray] = { ⚠ Gousse-Gousse +1
166     | val file = new BufferedInputStream(new FileInputStream(path))
167     | try {
168       | val return_value: DataArray = file.readAllBytes()
169       | Some(return_value)
170     | } catch {
171       | case _: Throwable => Option.empty
172     | } finally {
173       | file.close()
174     | }
175   }
```


Protobuf + gRPC communication

Master and Workers run on different machines

Need to send complex data structures over network

What we used:

- Protocol Buffers (Protobuf)
- Defines message structure in common.proto
- Generates java classes

common.proto → protoc compiler → Generated classes → Our code

- gRPC
- Handles network communication

```
message SampleRequest {  
    int32 worker_id = 1;  
    repeated bytes samples = 2;  
}
```

```
public final class Common {  
    public static final class SampleRequest {  
        private int workerId;  
        private List<ByteString> samples;  
  
        public int getWorkerId() { return workerId; }  
        public List<ByteString> getSamplesList() { return samples; }  
  
        public static Builder newBuilder() { return new Builder(); }  
  
        public static final class Builder {  
            public Builder setWorkerId(int value) { ... }  
            public Builder addAllSamples(Iterable<ByteString> values) { ... }  
            public SampleRequest build() { ... }  
        }  
    }  
}
```

```
val request = Common.SampleRequest.newBuilder()  
    .setWorkerId(workerId)  
    .addAllSamples(samplesProto)  
    .build()
```


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