Distributed Sorting

Final Presentation

20190629 김균서
20150271 정용준

Previously…

Week 6 : Implement Workers & Master functions (Now)
Week 7 : Implement network communication between Workers & Master / Testing
Week 8 : Testing & Debugging

Overall Progress

- Worker & Master Code locally Completed

  (sorting, sample data, partitioning, pivot, stage setting)

- Network (gRPC) Incomplete

  (shuffling, worker-worker, worker-master)

- Code Merging Incomplete

- Implementing Incomplete

```
// projects
lazy val distSorting = project
  .in(file("."))
  .settings(name := "distSorting")
  .settings(commonSettings)
  .aggregate(network, master, worker)

lazy val network = project
  .in(file("./network"))
  .settings(name := "network")
  .settings(commonSettings)
  .settings(
    mainClass in Compile := Some("distSorting.network.Main")
  )
```

```
service MasterService {
  // Registers a worker with the master node.
  rpc AddWorker (WorkerRegistration) returns (OperationResponse) {}

  // Reports task status and results from worker to master.
  rpc SubmitTaskResult (TaskResultReport) returns (OperationResponse) {}
}

service WorkerService {
  // Requests a task assignment from the master.
  rpc GetTask (TaskRequest) returns (OperationResponse) {}
}
```

Causes of Failure

We could trace most of our failures from our textbook - The Mythical Man-Month

- Optimalism:
  Underestimated time required for Coding & Merging implementations

- Gutless Estimating:
  Should’ve considered members’ development skills and project experiences
  Overconfidence in AI-assisted coding

- Lack of Competence
  Inexperienced with git, remote server, team-based programming project,
  Lack of programming skills, knowledge of building network connections

Causes of Failure

- Rare online communications:
  Relied on face-to-face meetings for much of its progress
  → single omission led to a big delay & communication errors

- Lack of detailed design:
  missing detailed commands for each task
  predetermined class, func, var names and object structure might help

- Document:
  leaving documents for each meeting would change a lot
  learned from other teams

Lessons Learned

- Learned the importance of project manager

  (issues with schedule management and communication)

- Needed enough time for merging

  (Met 3~4 days before the deadline to merge individually written code, Executed code locally without sharing with others)

- Team project challenges

  (lacked team coding experience & underestimated the challenges compared to solo projects)

- Self-Directed Learning Project

  (Independently solving the project felt like working in a company, showing the high bar of the professional world.)

Q&A

QnA