Prepared by Mokhtar and Ignasio

Group Project

Distributed Sorting

10.12.2024

Languages and libraries

Languages: Scala (3.5.0), Java for Protobuf stubs.
Libraries/Tools:
gRPC: Communication between master and workers.
Protobuf: Serialization of messages.
ScalaPB: Scala-specific Protobuf generation.

Dependencies

```
"io.grpc" % "grpc-netty" % "1.58.0",
"io.grpc" % "grpc-protobuf" % "1.58.0",
"io.grpc" % "grpc-stub" % "1.58.0",
"com.google.protobuf" % "protobuf-java" % "3.21.7",(or 29.0)
"com.thesamet.scalapb" %% "scalapb-runtime" % "0.11.12" % "protobuf",
"com.thesamet.scalapb" %% "scalapb-runtime-grpc" % "0.11.12"
```

Building code with scalaPB

```
Compile / PB.protoSources += baseDirectory.value / "src" / "main" /
"protobuf"

PB.targets in Compile := Seq(
  scalapb.gen() -> (Compile / sourceManaged).value,
  grpc.scalapb.gen() -> (Compile / sourceManaged).value)
```

Java protobuf

nstead of ScalaPB, we initially used Java Protobuf with gRPC, where:

1. Protoc was used to generate Java files (SortServiceGrpc and SortServiceOuterClass) manually.
2. gRPC communication was handled using Java-generated stubs within a Scala project.

This approach required:

- Manually generating Protobuf and gRPC files using the protoc compiler because of the failure of generating it via sbt.
- Adding dependencies like grpc-netty, grpc-stub, grpc-protobuf, and protobuf-java to build.sbt.

Challenges

- Protobuf and gRPC Integration:
  - Misalignment between protoc versions (29.0 vs 3.21.x).
  - Manual vs. automatic file generation conflicts in SBT.
- Cyclic Dependencies:
  - Resolution issues with generated Protobuf classes in Scala.
- Tooling Complexity:
  - IntelliJ integration with Protobuf and gRPC stubs.
  - Debugging missing or overwritten files in the target directory.

Progress Achieved

1. Implemented Features:

- Master node logic for dividing and assigning tasks.
- Worker node logic for processing and responding.
- Basic communication setup via gRPC.

2. What Works:

- Protobuf messages are generated manually.
- java files resulting from protobuf generated manually (sbt compile does not run with java protobuf and does not even buildwith scalaPB)

Lessons learned

1. Importance of tool version compatibility.
2. Challenges in integrating multiple systems (Scala, Java, gRPC, and Protobuf).
3. Debugging complex distributed systems requires persistence and attention to detail (even though I was patient and wasted more than 30 hours for the project setup and it did not produce any results)
4. Knowing the complexity of a project is difficult without actually starting to code

About AI models

- chatGPT is very poor in terms of helping in the setup, he even makes it worse sometimes
- It can be slightly better by creating discussion techniques as setting rules at the start of the conversationand reminding it of the rules every 4 messages (but wastes a lot of time in this process
- Tried Claude.ai, slightly better than chatgpt but limited conversation size in the free version