CSED332 소프트웨어설계방법

TEAM GREEN

20240505 공현성
20240985 박찬웅
20200120 차선호

Overall Plans

Week	TODO
Week 1	Planning
Week 2 (Midterm)	Design ideas
Week 3	Overall project design
Week 4	Create test code
Week 5	Creating physical code & Testing the system
Week 6 (Progress Slides Deadline)	Organizing Content, Preparing for Intermediate Presentation
Week 7	Project Improvement and Maintenance
Week 8 (Project Deadline)	Preparing for Final Presentation

Basic Features

Role Division

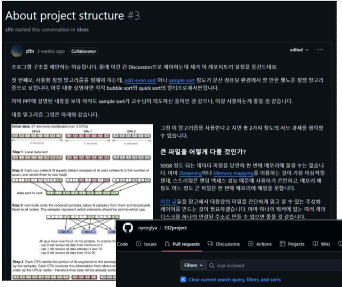
zlfn(박찬웅)	Communication & Basic Structure
nyeoglya(공현성)	Master & Argument Parser
carprefer(차선호)	Worker

Team Communication

Regular meeting (On every Sunday 8pm, At POSTECH Library GSR)
Github Discussion

Manage Git Repository, Documentation

Commit convention
Github PR
Scala Naming convention
scaladoc



Overall Project Design

Worker

object Main extends ZIOAppDefault

- Worker의 Main

ServiceImpl

- WorkerService를 이용해서 request를 실제로 처리하는 클래스
- Worker 간의 데이터 통신도 관할

WorkerService extends WorkerServiceLogic

- Worker의 기본 Logic을 다루는 클래스

Config(args: Seq[String]) extends ScallopConf(args)

- CommandLine Argument Parsing 하여 그 결과를 담는 클래스

Master

object Main extends ZIOAppDefault

- Master의 Main

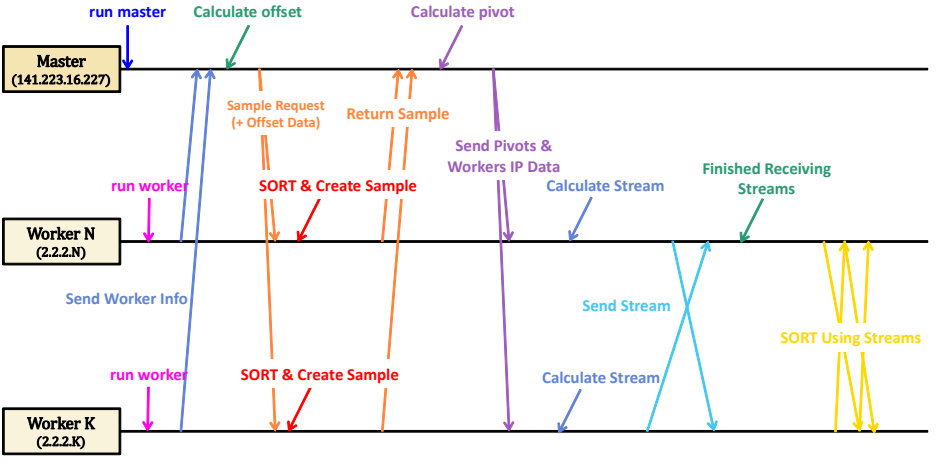
ServiceImpl(service: MasterLogic) extends MasterService

- MasterService를 이용하여 통신을 처리하는 클래스
- Server의 역할을 한다

MasterLogic

- Master의 기본 Logic을 다루고 통신을 처리하는 클래스
- Client의 역할을 한다

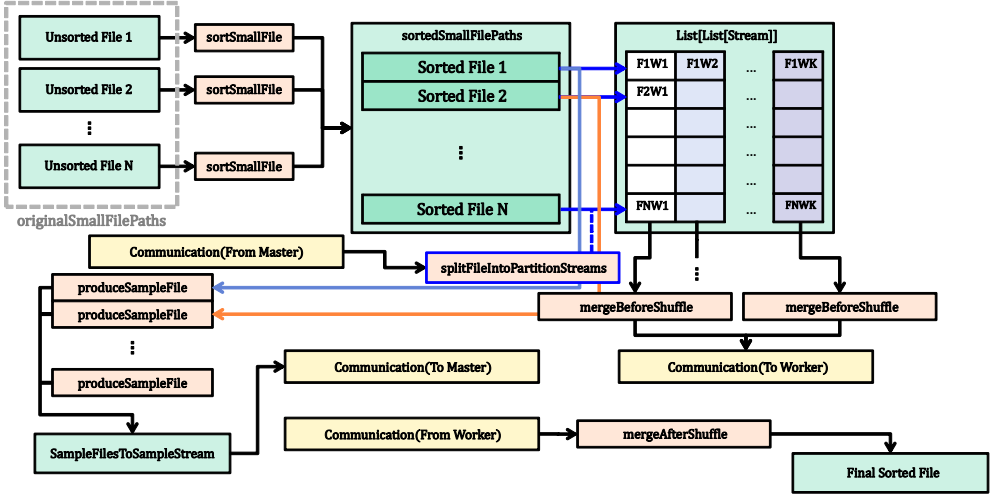
Overall Program Flow



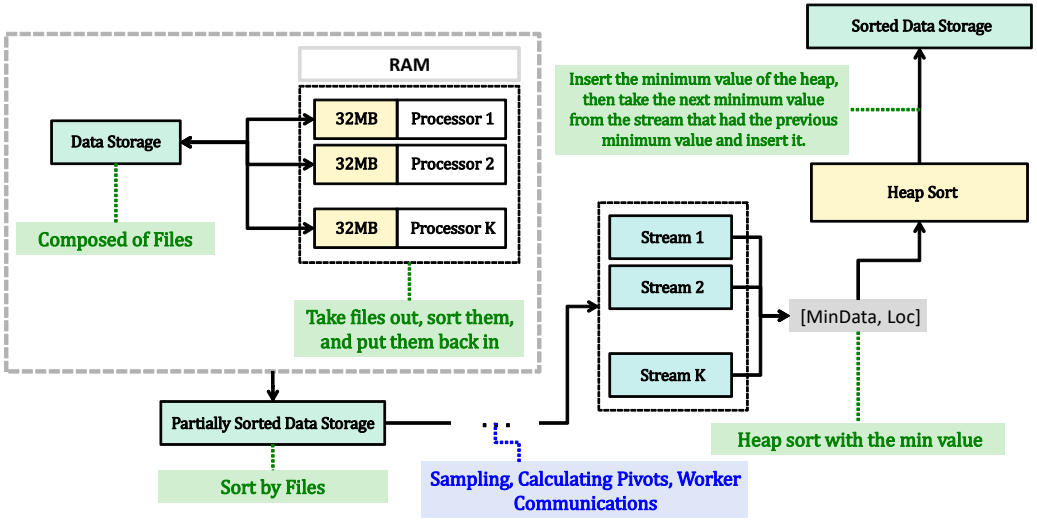
ProtoBuf Message

Name	When	Source -> Dest.		Information Contained
WorkerData	As soon as the Worker starts executing	Worker	Master	int64 **fileSize** = 1; string **workerAddress** = 2;
WorkerDataResponse	When the Master successfully receives information from the Worker	Master	Worker	x
SampleRequest	When the Master requests samples to the Worker	Master	Worker	int64 **offset** = 1;
SampleResponse	When the Worker returns the sample to the Master	Worker	Master	repeated string **pivots** = 1;
ShuffleRequest	When the Master sends pivot information and other worker information to the Worker	Master	Worker	Pivots **pivots** = 1; repeated string **workerAddresses** = 2;
ShuffleResponse	When the Worker successfully receives the pivot information	Worker	Master	x
DataResponse	When the Worker receives a stream from another Worker	Worker	Worker	x
Entity	**case class**	Any	Any	string **head** = 1; string **body** = 2;
Pivots	**case class**	Any	Any	repeated string **pivots** = 1;

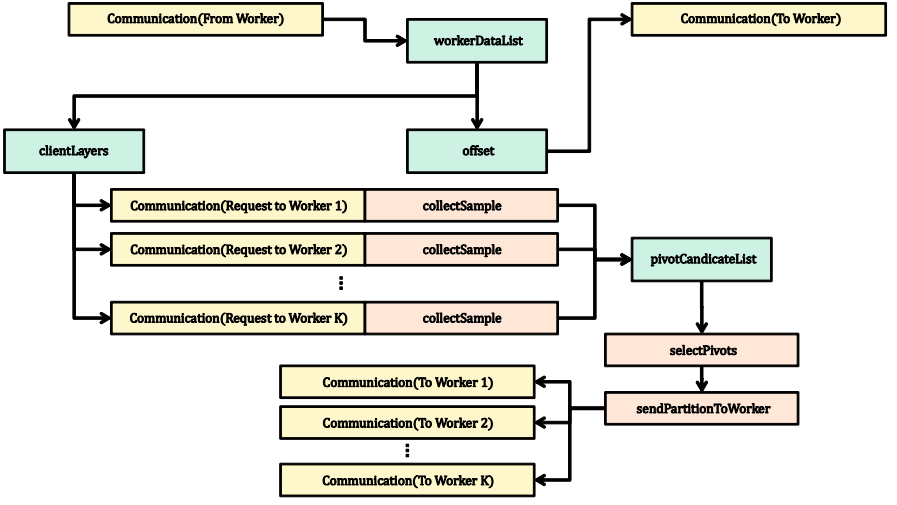
Specific Implementation - WorkerLogic & ServiceImpl



Specific Implementation - Disk Based Sorting (Worker)



Specific Implementation - MasterLogic



Specific Implementation - Environment & Libraries

Environment	Linux(Ubuntu, Arch), Windows
Logging	zio-logging 2.1.12 **(Plan)**
Key Features	JDK 22, Scala 2.13.15, SBT 1.10

Library Name	Versions	Purpose
scalapb-runtime, scalapb-runtime-grpc	0.11.17	gRPC (communication), Protobuf (serialization)
zio, zio-streams	2.1.12	Safer functional programming
zio-test, zio-test-sbt, zio-test-junit	2.1.12	Test ZIO apps
junit, junit-interface, scalatest	4.10, 0.13.3, 3.0.8	Test basic functionality
scallop	5.1.0	Commandline argument parser
sbt-assembly	0.15.0	Make .jar file from scala code
sbt-protoc	1.0.6	To Compile protobuf files

Current Progress : Implement(Abstract) & Testing

	Done	Currently Working
Master	Get workerDataList	Sending pivots to Workers
	Calculate offset	
	Collecting samples from worker	
	Pivot selecting	
Worker	Entire logics about sorting	Communication between Workers
	Communication with Master	

Current Progress	
Unit Test	WorkerLogic 테스트 중 MasterLogic 구성 중
Integration Test	구성 중
System Test	구성 중

직접 써보며 팀원들이 느낀 ZIO 장단점

장점

안전한 비동기 처리	Future, Promise와 달리 예외를 안전하게 처리함. 비동기 코드가 동기 코드처럼 읽히게 해서 익숙해지면 코드 이해가 빨라짐. 비동기 작업을 Effect로 다뤄서 컴파일 타임에 비동기 작업의 오류를 잡을 수 있음.
오류 처리	IO[E, A] 타입 쓰면 성공/실패 명시적으로 구분됨. 다양한 오류를 구체적으로 다룰 수 있음.
함수형 프로그래밍	순수한 함수형 프로그래밍 지원함. 그래서 비동기 테스트 하기가 용이함.

단점

프로젝트 비용 증가	배우기 어렵고 복잡한 개념이 많음. 학습하는 데 꽤 많은 시간을 썼다. 제공하는 기능은 많은데 이걸 100% 활용하기 어려움
기존 코드와 호환성 부족함	Future, Promise를 사용한 코드를 그대로 가져다 쓸 수가 없음. Test, Main도 ZIO에서 만든 것을 써야 하기 때문에 별도의 시간을 씀.
디버깅이 어려움	에러를 정교하게 처리하게 돕는 반면에 디버깅이 까다로워서 복잡한 오류를 추적하기 좀 힘듦.
생태계 부족 및 빠른 변화	신생 라이브러리라서 업데이트별로 기능들이 굉장히 빠르게 바뀜. 버전별로 함수나 기능들이 달라서 이걸 알아보고 적용하기가 복잡함.

Thank you
Q&A

Appendix) Current Progress : Implementation(Specific)

Master	class Config(args: Seq[String]) extends ScallopConf(args) lazy val offset: Long def addClient(clientAddress: String, clientSize: BigInt): Unit def collectSample(client: Layer[Throwable, WorkerServiceClient]): ZIO[WorkerServiceClient, Throwable, Pivots] def selectPivots(pivotCandicateZIOList: ZIO[Any, Throwable, List[Pivots]]): ZIO[Any, Throwable, Pivots]
	def run() def sendPartitionToWorker(client: Layer[Throwable, WorkerServiceClient], pivots: ZIO[Any, Throwable, Pivots]): ZIO[Any, Throwable, Unit]
Comm.	override def run: ZIO[Environment with ZIOAppArgs with Scope,Any,Any] def builder def serverLive: ZLayer[MasterLogic, Throwable, zio_grpc.Server] class ServiceImpl(service: MasterLogic) extends MasterService val masterClientLayer: ZLayer[Config, Throwable, MasterServiceClient] val sendDataToMaster: ZIO[MasterServiceClient with WorkerServiceLogic, Throwable, WorkerDataResponse] def serverLive: ZLayer[WorkerServiceLogic, Throwable, zio_grpc.Server]
	def getSamples(request: SampleRequest): IO[StatusException,Pivots] def startShuffle(request: ShuffleRequest): IO[StatusException,SortResponse] def sendData(request: Stream[StatusException,Entity]): IO[StatusException,DataResponse]

Appendix) Current Progress : Implementation(Specific)

Worker	object PathMaker def saveEntities(index: Integer, data: Stream[Throwable,Entity]): Unit def getFileSize(): Int def getDataStream(partition: Pivots): List[Stream[Throwable,Entity]] def getSampleStream(offset: Integer, size: Integer): List[String] def sortStreams(data: List[Stream[StatusException,Entity]]): Stream[Throwable,Entity] def readFile(filePath : String) : List[Entity] def writeFile(filePath : String, data : List[Entity]) : Unit def sortSmallFile(filePath : String) : String def produceSampleFile(filePath : String, stride : Integer) : String def sampleFilesToSampleStream(filePaths : List[String]) : List[String] def splitFileIntoPartitionStreams(filePath : String, pivots : List[String]) : List[Stream[Throwable, Entity]] def mergeBeforeShuffle(partitionStreams : List[Stream[Throwable, Entity]]) : Stream[Throwable, Entity] def mergeAfterShuffle(workerStreams : List[Stream[Throwable, Entity]]) : Stream[Throwable, Entity] val originalSmallFilePaths : List[String] val sortedSmallFilePaths : List[String]
	def inputEntities: Stream[Throwable, Entity] def getFileSize(): Int def saveEntities(index: Integer, data: Stream[Throwable, Entity]) def getSampleStream(offset: Integer, size: Integer): List[String] def getDataStream(partition: Pivots): List[Stream[Throwable, Entity]] def sortStreams(data: List[Stream[StatusException, Entity]]): Stream[Throwable, Entity] def inputEntities: Stream[Throwable, Entity] def zio2entity(zio : ZIO[Any, Throwable, Entity]) : Entity