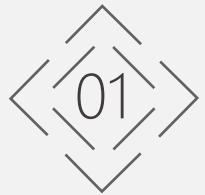


# Movie Recommendation System

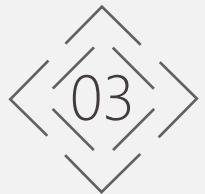
Yunlu Liao zheng  
Lingfeng Zhou  
Cheng Jin



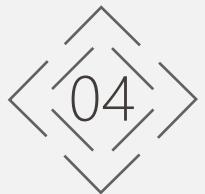
## Goal Review and Completion Report



## System Structure



## Demo Presentation and Test Result



## Implementation Detail



# Goal Review & Completion Report

# Goal

Build a scalable movie recommendation System to answer the question "What movie should I watch tonight?". It will return some related movies based on the given movie.

## Acceptance criteria:

The system should **return answers within 3 seconds and can handle maximum 10 requests per second.**

## Three modules of the system:

**Controller:** The frontend contains webserver, user interface, session management, authentication.

**Model:** The model contains main logic of generating response and manipulating data.

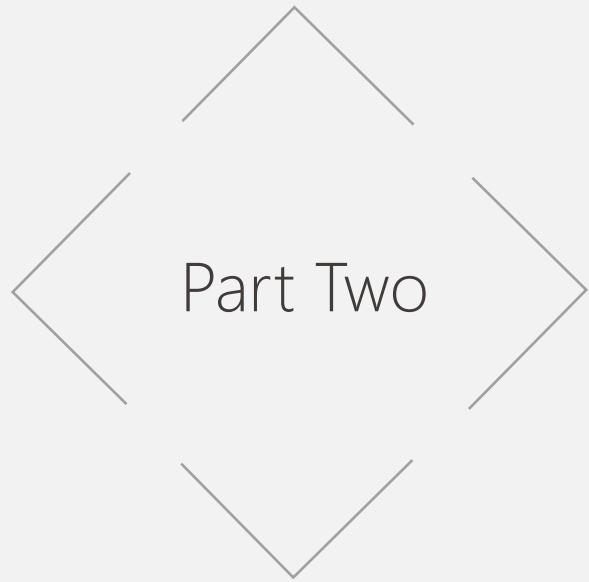
**Datacenter:** The data manager contains all calculations and low-level data management.

# Completion

**Controller:** Controller was implemented as a stateless server based on Scala Play framework. As it is stateless, it is safe to deploy multiple servers, users can access each of them.

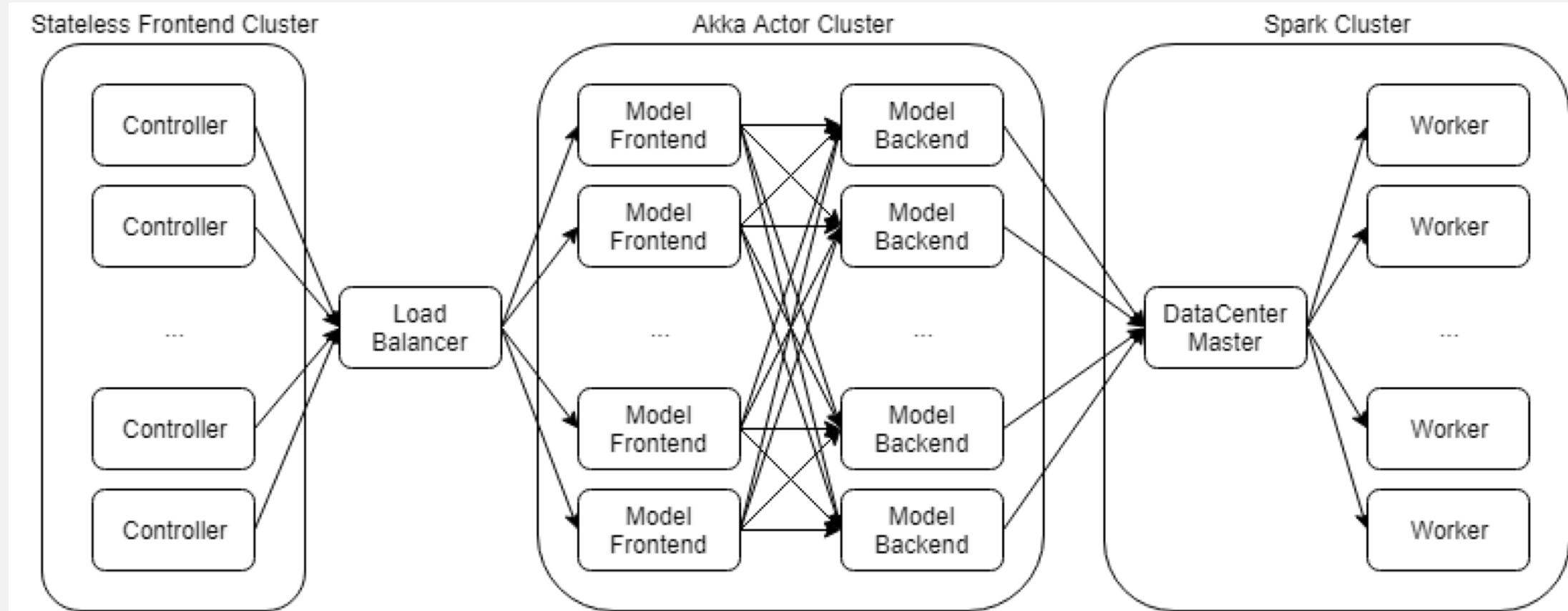
**Model:** Model part was implemented as a frontend/backend cluster based on Akka Actor Cluster. The frontend handled requests comes from Controller, and forward requests to backend. The backend is used to do real works.

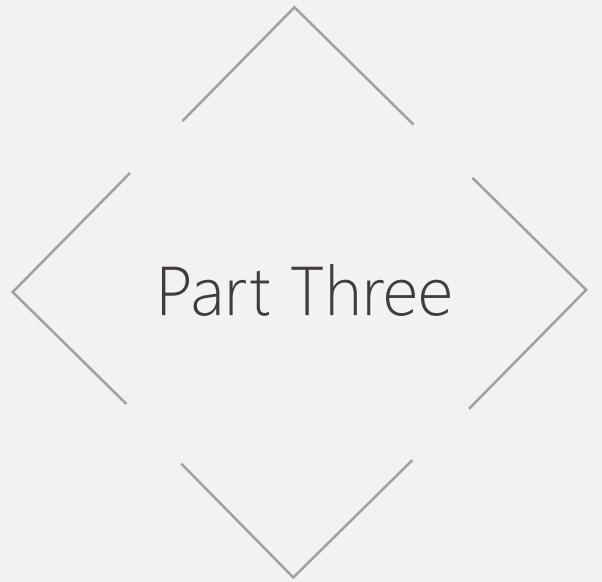
**Datacenter:** Datacenter part was implemented as a Spark Cluster, which manages over 1,000,000 movies.



# System Structure

# System Structure





# Demo Presentation & Test Result

# Test Result

Summary Report.jmx (/home/sraw/Summary Report.jmx) - Apache JMeter (2.13.20170723)

File Edit Search Run Options Help

Test Plan

Thread Group

- HTTP Header Manager
- HTTP Request
- Summary Report
- View Results Tree

WorkBench

**Summary Report**

Name: Summary Report

Comments:

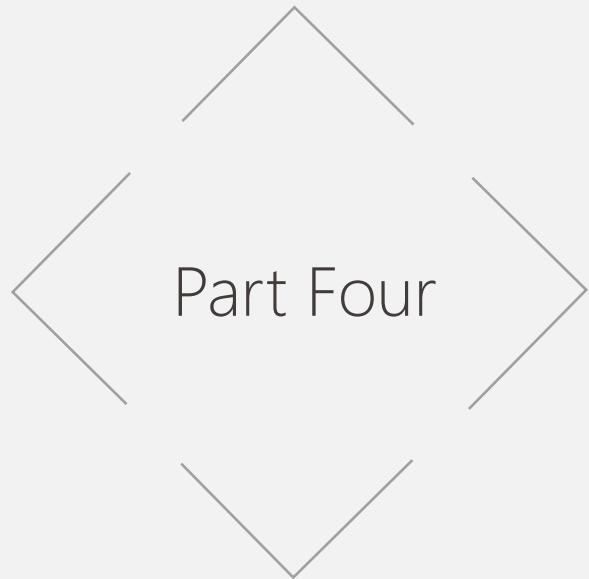
Write results to file / Read from file

Filename  Browse... Log/Display Only:  Errors  Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	10	1578	449	2574	697.62	0.00%	2.9/sec	3.64	1295.0
TOTAL	10	1578	449	2574	697.62	0.00%	2.9/sec	3.64	1295.0

Include group name in label?   Save Table Header

**Criteria:** As we can see from the report, the minimum time for 10 request is 449 milliseconds, maximum time is 2574 milliseconds, less than 3 seconds.



# Implementation Detail

# Akka Http

```
class ModelFrontend(port: Int) extends HttpRoute {
    val config: com.typesafe.config.Config = Config.getFrontendConfig(port)

    implicit val system: ActorSystem = ActorSystem("ModelCluster", config)
    implicit val executor: ExecutionContextExecutor = system.dispatcher
    implicit val materializer: ActorMaterializer = ActorMaterializer()
    val logger = Logging(system, getClass)

    val postSender: ActorRef = system.actorOf(Props[PostSender], name = "frontend")

    private val mode = config.getString(path = "frontend.mode")

    private val interface: String = config.getString(path = s"frontend.$mode.interface")
    private val HttpServerPort: Int = config.getInt(path = s"frontend.$mode.port")

    def start :Unit {
        start(HttpServerPort)
    }

    private def start(HttpServerPort: Int): Unit = {
        Http().bindAndHandle(route, interface, HttpServerPort) onComplete {
            case Success(_) => logger.info(s"Server listens on $interface:$HttpServerPort")
            case Failure(exception) => exception.getCause match {
                case _: java.net.BindException => start(HttpServerPort + 1)
                case causeException => causeException.printStackTrace()
            }
        }
    }
}
```

# Akka actor cluster

```
class StaticModel(config: Config, dataSourceOption: Option[RequestEntity => HttpResponseMessage]) extends Actor with ActorLogging with JSONSupport {
    def this(config: Config) = this(config, None)

    import context.dispatcher
    val cluster = Cluster(context.system)

    override def preStart(): Unit = cluster.subscribe(self, classOf[MemberUp])
    override def postStop(): Unit = cluster.unsubscribe(self)

    final implicit val materializer: ActorMaterializer = ActorMaterializer(ActorMaterializerSettings(context.system))

    val http = Http(context.system)

    def register(member: Member): Unit = {
        if (member.hasRole(role = "frontend")) context.actorSelection(RootActorPath(member.address) / "user" / "frontend") ! BackendRegistration(member)
    }

    def receive: PartialFunction[Any, Unit] = {
        case Movie(title) => handleRequest(Marshal(MovieRequest(title, 3)).to[RequestEntity])
        case HttpResponse(StatusCodes.OK, _, entity, _) =>
            log.debug(s"Receive movie info: $entity")
            handleMovieInfo(Unmarshal(entity.withContentType(ContentTypes.`application/json`)).to[MovieResponse])
        case resp@HttpResponse(code, _, _, _) =>
            log.info("Request failed, response code: " + code)
            resp.discardEntityBytes()
        case state: CurrentClusterState =>
            state.members.filter(_.status == MemberStatus.Up).foreach(register)
        case MemberUp(m) => register(m)
    }
}
```

# Spark cluster

```
object DoKmeans {
    val spark = SparkSession.builder
        .master(Config.getSparkMaster())
        .appName(Config.getSparkName())
        .getOrCreate()
    spark.sparkContext.setLogLevel(Config.getLogLevel())
    import spark.implicits._

    def train(): DataFrame = {

        val file_basic = "DataCenter/src/main/resources/title.basics.tsv"
        val df_basic_raw = spark.read.format( source = "csv")
            .option("header", "true")
            .option("inferSchema", "true")
            .option("delimiter", "\t")
            .load(file_basic)
            .cache()
        df_basic_raw.printSchema()
        val df_basic = df_basic_raw.filter( conditionExpr = "titleType == 'movie'")
            .filter( conditionExpr = "runtimeMinutes != '\\\\\\\\N' AND genres != '\\\\\\\\N'")
            .drop( colNames = "endYear", "isAdult")
        //    val genre = df_basic.select("genres").map(attributes => attributes.toString().split(",").head)

        val file_names = "DataCenter/src/main/resources/name.basics.tsv"
        val df_names_raw = spark.read.format( source = "csv")
```

# Optimization

```
object GetResults {
    val spark = SparkSession.builder
        .master(Config.getSparkMaster())
        .appName(Config.getSparkName())
        .getOrCreate()
    spark.sparkContext.setLogLevel(Config.getLogLevel())
    import spark.implicits._
    val df = spark.read.parquet( path = "DataCenter/src/main/resources/result.parquet").cache()
    def getmovies(title: String, limit: Int): String = {

        //read result select required rows

        val movie_df = df.select( col = "primaryTitle", cols = "genresSeq", "director","actor","rms",
        "averageRating", "numVotes","year","prediction","distanceFromCenter")

        val newNames = Seq("title", "genres", "director", "actor", "length", "score", "voteNumber",
        "year","prediction","distanceFromCenter")

        val dfRenamed = movie_df.toDF(newNames: _*)
        //get cluster
        val cluster = dfRenamed.select( col = "title", cols = "prediction").filter( condition = $"title"==title).limit(1)("predict
        //sort with distance
        val related_df: Dataset[Row] = dfRenamed.filter( condition = $"prediction" === cluster).sort(asc( columnName = "distanceF

        val all = related_df.filter( condition = $"prediction" === cluster)
        .drop( colNames = "prediction","distanceFromCenter")
        val related = all.filter( condition = $"title"!=title).limit(limit)
        val origin_df = all.filter( condition = $"title"==title)
        val res = origin_df.union(related).na.fill( value = "").toJSON.collect()
        val origin_json = res(0)
        val related_json = res.drop(1).mkString("\\\"related\\\": [", ",", ", ", "]\" ")

        "{ \"origin\": ".concat(origin_json).concat( str = ",").concat(related_json).concat( str = " }"
    }
}
```

Only one action used

# Unit Test

The screenshot shows a Scala unit test named `StaticModelSpec` within a project structure. The test file contains code for interacting with a `StaticModel` actor and asserting its behavior against JSON responses.

```
class StaticModelSpec() extends TestKit(ActorSystem("StaticModelSpec")) with ImplicitSender
with WordSpecLike with Matchers with BeforeAndAfterAll {

  def GetResult(path: String): MovieResponse = {
    val dataSource: String => RequestEntity => HttpResponse =
      _ => HttpResponse(entity = entity, headers = List(headers.`Content-Type`(ContentTypes.`application/json`)))
    val fileStream = getClass.getResourceAsStream(path)
    val entity = Source.fromInputStream(fileStream).mkString
    val staticModel = TestActorRef(Props(classOf[StaticModel]), ConfigFactory.empty(), Some(dataSource(entity)))
    implicit val timeout: Timeout = 3.seconds
    val future = staticModel ? Movie("asdfasd")
    val Success(result) = future.mapTo[MovieResponse].value.get
    result
  }

  override def afterAll: Unit = {
    TestKit.shutdownActorSystem(system)
  }

  "StaticModel actor" must {
    "work with response1.json" in {
      val result = GetResult(path = "response1.json")
      result shouldBe MovieResponse(MovieInfo("Test", List(), "", "", 100, 8.9, 1000, 1980), List())
    }
    "work with response2.json" in {
      val result: MovieResponse = GetResult(path = "response2.json")
      result.related.length shouldBe 6
    }
  }
}
```

The test passes with 2 of 2 tests in 781 ms. The log output shows several INFO messages from the Akka system, indicating interactions with the `StaticModelSpec` actor system.

```
[INFO] [12/15/2018 20:34:51.836] [StaticModelSpec-akka.remote.default-remote-dispatcher-6] [akka://StaticModelSpec@127.0.0.1]
[INFO] [12/15/2018 20:34:51.838] [StaticModelSpec-akka.remote.default-remote-dispatcher-6] [akka://StaticModelSpec@127.0.0.1]
[INFO] [12/15/2018 20:34:51.849] [StaticModelSpec-akka.actor.default-dispatcher-2] [akka://StaticModelSpec/deadLetters] Mess
[INFO] [12/15/2018 20:34:51.850] [StaticModelSpec-akka.actor.default-dispatcher-2] [akka://StaticModelSpec/deadLetters] Mess
[INFO] [12/15/2018 20:34:51.851] [StaticModelSpec-akka.actor.default-dispatcher-4] [akka://StaticModelSpec/deadLetters] Mess
[INFO] [12/15/2018 20:34:51.862] [StaticModelSpec-akka.remote.default-remote-dispatcher-5] [akka://StaticModelSpec@127.0.0.1]
```

# Unit Test

# Unit test

The screenshot shows a Scala project structure and a code editor with a test file open. The project structure includes packages like ClassificationActor, ClassificationRoutes, Config, DoKmeans, GetResults, JsonSupport, MovieServer, OneHencoder, and Util. A 'test' directory contains 'scala' and 'com.csye7200.datacenter' packages, which further contain 'ActorsSpec' and 'RoutesSpec' files. The code editor displays the 'RoutesSpec.scala' file, which is a WordSpec test for 'ClassificationRoutes'. It imports akka.actor.ActorRef, akka.http.scaladsl.marshalling.Marshal, akka.http.scaladsl.model.{HttpHeader, HttpRequest, MessageEntity}, akka.http.scaladsl.testkit.{RouteTestTimeout, ScalatestRouteTest}, org.scalatest.Matchers, org.scalatest.WordSpec, org.scalatest.concurrent.ScalatestFutures, scala.concurrent.duration.\_, and akka.testkit.TestDuration. The test suite overrides classificationActor and defines lazy val routes = classificationRoutes. It then performs a 'GetMovieRoutes' check, creating an Info object and sending a Post request to '/movies/element/getMovie' with the entity. The status is checked to be OK. The 'Run' tab at the bottom shows 'All Tests Passed' with a duration of 11s 231ms. The log output shows Spark UI and BlockManager logs.

```
import akka.actor.ActorRef
import akka.http.scaladsl.marshalling.Marshal
import akka.http.scaladsl.model.{HttpHeader, HttpRequest, MessageEntity}
import akka.http.scaladsl.testkit.{RouteTestTimeout, ScalatestRouteTest}
import org.scalatest.Matchers
import org.scalatest.WordSpec
import org.scalatest.concurrent.ScalatestFutures
import scala.concurrent.duration._
import akka.testkit.TestDuration

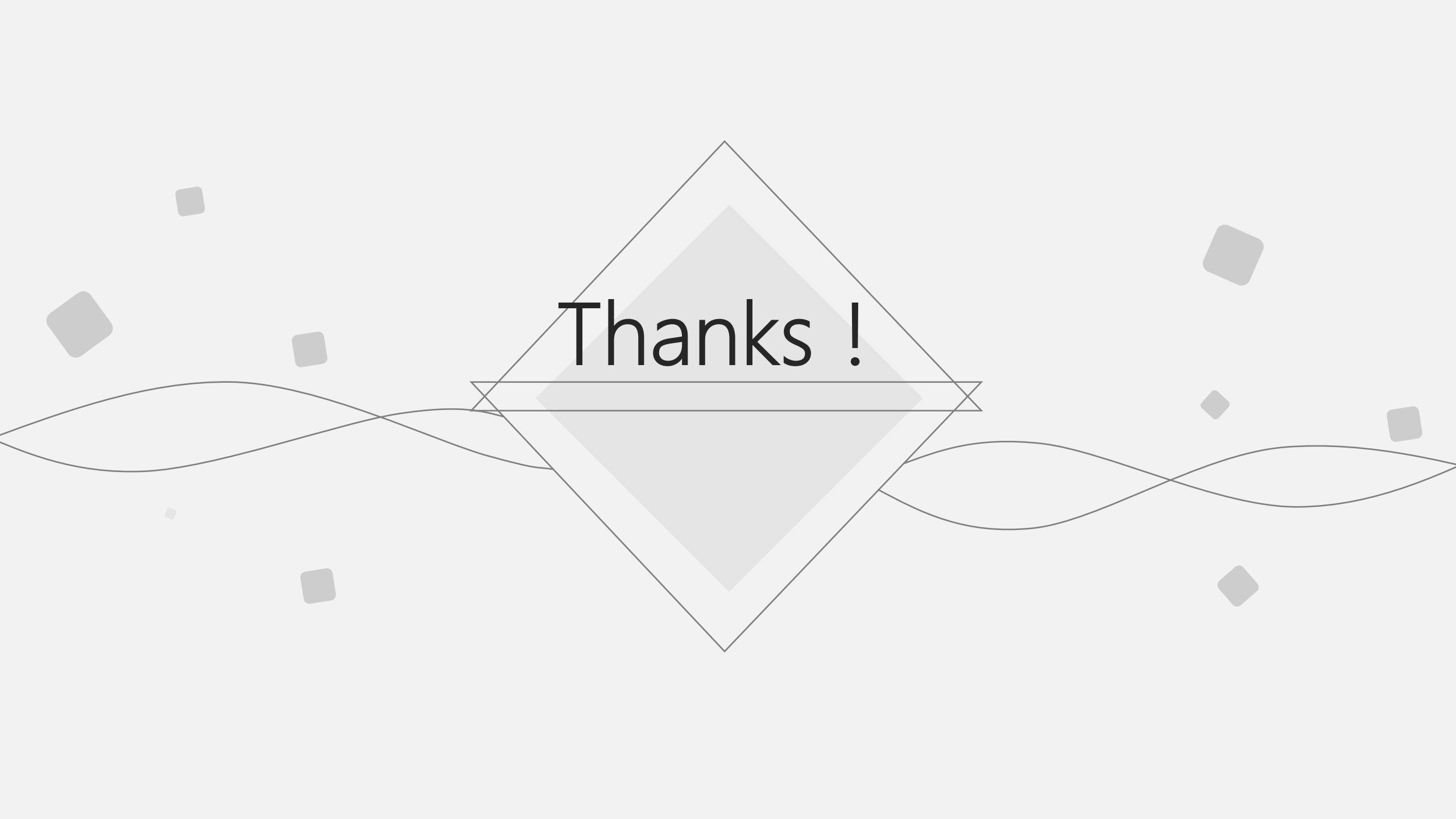
class RoutesSpec extends WordSpec with Matchers with ScalatestRouteTest
with ClassificationRoutes {
  override val classificationActor: ActorRef = system.actorOf(ClassificationActor.props, name = "classificationActor")

  lazy val routes = classificationRoutes
  implicit val timeout = RouteTestTimeout(20.seconds.dilated)
  "GetMovieRoutes" should {
    "return ok" in {
      val info = Info("Jana, the Girl from the Bohemian Forest", 3)
      val infoEntity = Marshal(info).to[MessageEntity].futureValue

      val request = Post("/movies/element/getMovie").withEntity(infoEntity)
      request ~> routes ~> check{
        status should === (StatusCodes.OK)
      }
    }
  }
}

Run: RoutesSpec
Tests passed: 1 of 1 test - 11s 231 ms
All Tests Passed 11s 231ms
18/12/15 21:50:57 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/12/15 21:50:57 INFO DiskBlockManager: Created local directory at /private/var/folders/lh/n7_2c43x3vb6mx8l6kjrm6c0000gr
18/12/15 21:50:57 INFO MemoryStore: MemoryStore started with capacity 912.3 MB
18/12/15 21:50:57 INFO SparkEnv: Registering OutputCommitCoordinator
18/12/15 21:50:57 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/12/15 21:50:57 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://10.0.0.104:4040
18/12/15 21:50:57 INFO Executor: Starting executor ID driver on host localhost
18/12/15 21:50:57 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 52227.
18/12/15 21:50:57 INFO NettyBlockTransferService: Server created on 10.0.0.104:52227
18/12/15 21:50:57 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
18/12/15 21:50:57 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.0.0.104, 52227, None)
18/12/15 21:50:57 INFO BlockManagerMasterEndpoint: Registering block manager 10.0.0.104:52227 with 912.3 MB RAM, BlockManagerId(driver, 10.0.0.104, 52227, None)
18/12/15 21:50:57 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.0.0.104, 52227, None)
18/12/15 21:50:57 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.0.0.104, 52227, None)
```

Process finished with exit code 0



Thanks !