The Fey Engine - A Component Based Processing System for IoT

Barbara Malta Gomes

barbaragomes@apache.org

and

Tony Faustini

tonyfaustini@apache.org

Background

- We work at Litbit (www.litbit.com) a machine learning based IoT start-up.
- Fey is a component based system written in Scala using the Akka framework
- Named after Charles Fey the inventor of the slot machine
- Fey core is a runtime that enables to scalable and fault-tolerant execution of Fey performers (software components)
- Runs on simple standalone ARM based systems like the Raspberry Pi to large clustered systems like Mesos
- iota is a project in incubation at Apache we are looking for a few good developers to contribute to the project
- We will demonstrate how to write Fey components and how to combine them in a Fey orchestration that runs on Fey core.

Overview of Fey

- Fey is a Scala framework developed using Akka Actors (<http://akka.io/>). If you are not familiar with this framework we suggested that you take sometime to look into it to get at least a basic idea of the Akka Actor model and concepts.
  The current version of Akka that is being used on Fey is `2.10.4`
  Java **SDK** (Software Development Kit) 1.8+
  Scala 2.11.8
  SBT 0.13.11 (<http://www.scala-sbt.org/>)

- Make sure you have git installed on your machine and have created a source directory for iota

  >> cd IOTA_SOURCE_DIR
  >> git clone git@github.com:apache/incubator-iota.git

- You will see three directories in the source directory
  1. Fey-core: Fey-core by itself does nothing, it is an Actor System
  2. Performers:. Performers do the actual work and they are the focus of this talk.
  3. Deployments - not relevant to this talk

Building Fey

- Building fey-core

  Fey core uses the SBT tool. In order to build the fey-core .jar go to your terminal and:
  >> cd IOTA_SOURCE_DIR\incubator-iota
  >> sbt
  >> project fey-core
  >> assembly

  In `IOTA_SOURCE_DIR/incubator-iota/fey-core/target/scala-2.11` you will find iota-fey-core.jar

- Once you have built fey-core, you must configure it. The default configuration can be found at IOTA_SOURCE_DIR/incubator-iota/fey-core\src\main\resources\application.conf

  Fey core does not create directories, the directories must be created before starting Fey. Here is a create a basic configuration for fey. Create a my-fey.conf file with the following content:

  fey-global-configuration{
  enable-checkpoint = true
  checkpoint-directory = ${HOME}"/feyCheckpoint"
  json-repository = ${HOME}"/feyJSONRepo"
  json-extension = ".json"
  jar-repository = ${HOME}"/feyJarRepo"
  log-level = “INFO"

- Create the directories that are going to be used by Fey:

  1. ${HOME}"/feyCheckpoint"
  2. ${HOME}"/feyJSONRepo"
  3. ${HOME}"/feyJarRepo"

Running Fey

- Once you have defined the my-fey.conf configuration file for fey, and created all the necessary directories you should be ready to run Fey.

  >> java -jar IOTA_SOURCE_DIR/incubator-iota/fey-core/target/scala-2.11/iota-fey-core.jar PATH_TO/my-fey.conf

- If all went well you will see the following in your console

```
[INFO] [17/02/07 12:11:17] akka.event.slf4j.Slf4jLogger [] : Slf4jLogger started
[INFO] [17/02/07 12:11:17] org.apache.iota.fey.FeyCore [akka://FEY-MANAGEMENT-SYSTEM/user/FEY-CORE] : Starting Fey Core
[INFO] [17/02/07 12:11:17] org.apache.iota.fey.GlobalWatchService [akka://FEY-MANAGEMENT-SYSTEM/user/FEY-CORE/GLOBAL_WATCH_SERVICE] : Starting Global Watcher from PRE-START
[INFO] [17/02/07 12:11:17] org.apache.iota.fey.Monitor [akka://FEY-MANAGEMENT-SYSTEM/user/FEY-MONITOR] : START | 1486498277506 | akka://FEY-MANAGEMENT-SYSTEM/user/FEY-CORE |
[INFO] [17/02/07 12:11:17] org.apache.iota.fey.Monitor [akka://FEY-MANAGEMENT-SYSTEM/user/FEY-MONITOR] : START | 1486498277514 | akka://FEY-MANAGEMENT-SYSTEM/user/FEY-CORE/JSON_RECEIVER |
[INFO] [17/02/07 12:11:17] akka.event.slf4j.Slf4jLogger [] : Slf4jLogger started
[INFO] [17/02/07 12:11:17] play.api.Play [] : Application started (Prod)

[INFO] [17/02/07 12:11:17] play.core.server.NettyServer [] : Listening for HTTP on /0:0:0:0:0:0:0:0:16666
```

- Go to your browser and type in <http://localhost:16666/fey/activeactors>.
  You should see one Actor System and 2 Actors running:

  ![Fey actor hierarchy](./images/Fey-Active-Actors.png)

- Open another tab in your browser and go to <http://localhost:16666/fey/monitoringevents>
  You should see the START event for the two Actors running on Fey:

  ![Fey actor hierarchy](./images/Fey-Active-Actors-Monitoring.png)

- You are now running Fey

Our First Fey Performer

- A Print Message Performer

```
package org.apache.iota.fey.performer

import akka.actor.ActorRef
import org.apache.iota.fey.FeyGenericActor

import scala.collection.immutable.Map
import scala.concurrent.duration._

class PrintMessage(override val params: Map[String, String] = Map.empty,
                   override val backoff: FiniteDuration = 1.minutes,
                   override val connectTo: Map[String, ActorRef] = Map.empty,
                   override val schedulerTimeInterval: FiniteDuration = 30.seconds,
                   override val orchestrationName: String = "",
                   override val orchestrationID: String = "",
                   override val autoScale: Boolean = false) extends FeyGenericActor {

  override def onStart : Unit = {   }

  override def onStop : Unit = {   }

  override def onRestart(reason: Throwable) : Unit = {
    // Called after actor is up and running - after self restart
  }

  override def customReceive: Receive = {    }

  override def processMessage[T](message: T, sender: ActorRef): Unit = {
    log.info(T)
  }

  override def execute() : Unit = {   }

}
```

Our Second Fey Performer

- A Time Message Performer

```
package org.apache.iota.fey.performer

import akka.actor.ActorRef
import org.apache.iota.fey.FeyGenericActor

import scala.collection.immutable.Map
import scala.concurrent.duration._

class PrintMessage(override val params: Map[String, String] = Map.empty,
                   override val backoff: FiniteDuration = 1.minutes,
                   override val connectTo: Map[String, ActorRef] = Map.empty,
                   override val schedulerTimeInterval: FiniteDuration = 30.seconds,
                   override val orchestrationName: String = "",
                   override val orchestrationID: String = "",
                   override val autoScale: Boolean = false) extends FeyGenericActor {

  override def onStart : Unit = {  }

  override def onStop : Unit = {   }

  override def onRestart(reason: Throwable) : Unit = {
    // Called after actor is up and running - after self restart
  }

  override def customReceive: Receive = { }

  override def processMessage[T](message: T, sender: ActorRef): Unit = {  }

  override def execute() : Unit = {
   val ts = java.lang.System.currentTimeMillis().toString
   propagateMessage(ts)

  }

}
```

Our First Fey Orchestration

- Our First Orchestration

```
{
  "guid": “First Orchestration UUID",
  "command": "CREATE",
  "timestamp": "591997890",
  "name": “First Orchestration",
  "ensembles": [
    {
      "guid": “Apache Con 2017",
      "command": "NONE",
      "performers": [
        {
          "guid": “Time Message",
          "schedule": 1000,
          "backoff": 0,
          "source": {
            "name": "apachecon.jar",
            "classPath": "org.apache.iota.fey.performer.timesender",
            "parameters": {
            }
          }
        },
        {
          "guid": “Print Message",
          "schedule": 0,
          "backoff": 0,
          "source": {
            "name": "apachecon.jar",
            "classPath": "org.apache.iota.fey.performer.printmessage",
            "parameters": {
            }
          }
        }
      ],
      "connections": [
                        {
                          “Time Message": [ “Print Message" ]
                        }
                      ]
    }
  ]
}
```

Questions

Support Slides

Build.Scala

- build.scala

```
import sbt._
import sbt.Keys._
import sbtassembly.AssemblyPlugin.autoImport._

object ModuleDependencies {

  import Dependencies._
  val FeyDependencies           = compile(akka_actor,typesafe_config,playJson,slf4j,log4jbind,sprayCan,
                                          sprayRouting,jsonValidator,javaFilter,codec,apacheIO,playNetty) ++ test(akka_testkit, scala_test)
  val ApacheConDependencies     = provided(akka_actor, fey)
}

object IotaBuild extends Build {

  import BuildSettings._

  lazy val parent = Project(
    id = "iota",
    base = file("."),
    aggregate = Seq(apachecon, fey),
    settings = rootbuildSettings ++ Seq(
      aggregate in update := false
    )
  )

  lazy val fey = Project(
    id = "fey-core",
    base = file("./fey-core"),
    settings = BasicSettings ++ FeybuildSettings ++ Seq(
      libraryDependencies ++= ModuleDependencies.FeyDependencies

    ))

  lazy val apachecon = Project(
    id = "apachecon",
    base = file("./performers/apachecon"),
    settings = BasicSettings ++ ApcheConbuildSettings ++ Seq(
      libraryDependencies ++= ModuleDependencies.ApacheConDependencies,
      assemblyJarName in assembly := "apachecon.jar"
    )
  )
}
```

Build Settings 1

- build settings 1

```
import sbt._
import sbt.Keys._
import sbtassembly.AssemblyPlugin.autoImport._

object BuildSettings {

  import Dependencies.Resolvers._

  val ParentProject = "iota"
  val Fey = "fey-core"
  val ApacheCon = "apachecon"


  val Version = "1.0"
  val ScalaVersion = "2.11.8"


  lazy val rootbuildSettings = Defaults.coreDefaultSettings ++ Seq(
    name := ParentProject,
    version := Version,
    scalaVersion := ScalaVersion,
    organization := "org.apache.iota",
    description := "Apache iota build",
    scalacOptions := Seq("-deprecation", "-unchecked", "-encoding", "utf8", "-Xlint")
  )

  lazy val BasicSettings = Seq(
    organization := "org.apache.iota",
    maxErrors := 5,
    ivyScala := ivyScala.value map {
      _.copy(overrideScalaVersion = true)
    },
    triggeredMessage := Watched.clearWhenTriggered,
    resolvers := allResolvers,
    fork := true,
    connectInput in run := true
  )
```

Build Settings 2

- build settings 2

```
lazy val FeybuildSettings = Defaults.coreDefaultSettings ++ Seq(
    name := Fey,
    version := "1.0-SNAPSHOT",
    scalaVersion := ScalaVersion,
    description := "Framework of the event processing / actions engine for IOTA",
    scalacOptions := Seq("-deprecation", "-unchecked", "-encoding", "utf8", "-Xlint"),
    mainClass := Some("org.apache.iota.fey.Application"),
    assemblyJarName in assembly := "iota-fey-core.jar",
    publishTo := {
      val nexus = "s3://maven.litbit.com/"
      if (isSnapshot.value)
        Some("snapshots" at nexus + "snapshots")
      else
        Some("releases"  at nexus + "releases")
    },
    publishMavenStyle := true,
    conflictManager := ConflictManager.latestRevision,
    assemblyMergeStrategy in assembly := {
      case "reference.conf"   => MergeStrategy.concat
      case "application.conf"   => MergeStrategy.concat
      case PathList("org", "slf4j", xs @ _*)  => MergeStrategy.last
      case PathList("META-INF", "io.netty.versions.properties") => MergeStrategy.last
      case PathList("scala", "xml", xs @ _*)          => MergeStrategy.last
      case x =>
        val oldStrategy = (assemblyMergeStrategy in assembly).value
        oldStrategy(x)
    },
    //All tests on Fey are integrated tests.
    //All tests need to be execute sequentially
    parallelExecution in Test := false,
    testOptions in Test += Tests.Cleanup( () => {
      print("\nCLeaning up")
      removeAll("/tmp/fey/test")
      def removeAll(path: String) = {
        def getRecursively(f: File): Seq[File] =
          f.listFiles.filter(_.isDirectory).flatMap(getRecursively) ++ f.listFiles
        getRecursively(new File(path)).foreach{f =>
          if (!f.delete()) println(s"could not delete ${f.getAbsolutePath}")}
      }
    })
  )

  lazy val ApacheConbuildSettings = Defaults.coreDefaultSettings ++ Seq(
    name := ApacheCon,
    version := Version,
    scalaVersion := ScalaVersion,
    description := "ApacheCon Application",
    scalacOptions := Seq("-deprecation", "-unchecked", "-encoding", "utf8", "-Xlint"),
    mainClass := Some("org.apache.iota.fey.performer.Application")
  )


}
```