Scaling
Up & Out

Haidar Osman

1- **Crash course in Scala**
- Classes
- Objects

2- **Actors**
- The Actor Model
- Examples I, II, III, IV

3- **Apache Spark**
- RDD & DAG
- Word Count Example

1- **Crash course in Scala**
- Classes
- Objects

2- **Actors**
- The Actor Model
- Examples I, II, III, IV

3- **Apache Spark**
- RDD & DAG
- Word Count Example

```java
public class Person {
    private String name;
    private String address;
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println("Person object is constructed");
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```

```java
public class Person {
    private String name;
    private String address;
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println("Person object is constructed");
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {

}
```

```java
public class Person {
    private String name;
    private String address;
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println("Person object is constructed");
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null



}
```

```java
public class Person {
    private String name;
    private String address;
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println("Person object is constructed");
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println("Person object is constructed")
  }


}
```

```java
public class Person {
    private String name;
    private String address;
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println("Person object is constructed");
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println("Person object is constructed")
  }
  def getName(): String = name
  def getAddress(): String = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println("Person object is constructed")
  }
  def getName(): String = name
  def getAddress(): String = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  private val welcomeNote: String = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName(): String = name
  def getAddress(): String = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  private val welcomeNote: String = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName(): String = name
  def getAddress(): String = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  private val welcomeNote: String = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName: String = name
  def getAddress: String = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  private val welcomeNote: String = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName: String = name
  def getAddress: String = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  private val welcomeNote = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName = name
  def getAddress = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  private var name: String = null
  private var address: String = null
  private val welcomeNote = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName = name
  def getAddress = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  var name: String = null
  var address: String = null
  val welcomeNote = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName = name
  def getAddress = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person {
  var name: String = null
  var address: String = null
  val welcomeNote = "Person object is constructed"
  def this(name: String, address: String) {
    this()
    this.name = name
    this.address = address
    println(welcomeNote)
  }
  def getName = name
  def getAddress = address
}
```

```java
public class Person {
    private String name;
    private String address;
    private final String welcomeNote = "Person object is constructed";
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
        System.out.println(welcomeNote);
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
}
```


```scala
class Person (var name: String, var address: String){
  val welcomeNote = "Person object is constructed"
  println(welcomeNote)



}
```

```
class Person (var name: String, var address: String){
  val welcomeNote = "Person object is constructed"
  println(welcomeNote)
}
```

```
class Person (var name: String, var address: String){
  Person.welcome
}
```

```
object Person {
  def welcome = {
    val welcomeNote = "Person object is constructed"
    println(welcomeNote)
  }
}
```

```java
public class MainClass {
    public static void main(String[] args){
        System.out.println("Hello World");
    }
}
```


```scala
object MainClass {
  def main(args: Array[String]): Unit = {
    println("Hello World")
  }
}
```

```
class Person (var name: String, var address: String){
  Person.welcome
}

object Person {
  def welcome = {
    val welcomeNote = "Person object is constructed"
    println(welcomeNote)
  }
}
```

```scala
case class Person (var name: String, var address: String){
  Person.welcome
}

object Person {
  def welcome = {
    val welcomeNote = "Person object is constructed"
    println(welcomeNote)
  }
}
```

```scala
case class Person (var name: String, var address: String){
  Person.welcome
}

object Person {
  def welcome = {
    val welcomeNote = "Person object is constructed"
    println(welcomeNote)
  }
}
```

```scala
object MainClass {
  def main(args: Array[String]): Unit = {
    val john = new Person("John","My Address")
    john match {
      case Person("John", address) =>
                println("John's adress: " + address)

      case _ =>
                println("not John")
    }
  }
}
```

1- **Crash course in Scala**
- Classes
- Objects

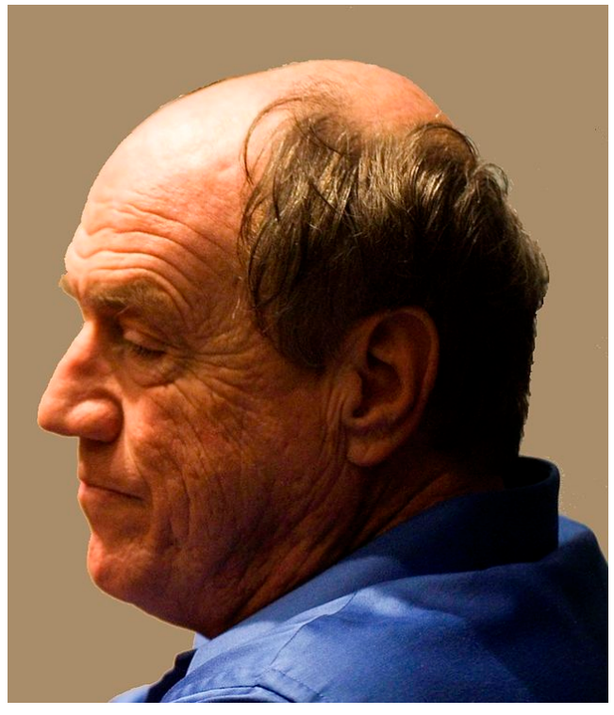
2- **Actors**
- The Actor Model
- Examples I, II, III, IV

3- **Apache Spark**
- RDD & DAG
- Word Count Example

What are actors?

- The Actors Model is “another” model for expressing computational problems.
  - It is equivalent to Turing Machine and Lambda Calculus

- Actors are not new. It is a solid and scientifically robust idea (Carl Hewitt 1973).



Why actors?

- Mutable shared state is the source of concurrency problems.
  - To protect shared state, and ensure safety, you need a locking mechanism.
- Mixing concurrency logic with business logic makes the code harder to maintain.
- There is an increased need for high performant systems.
  - *e.g.* big data applications
- Distributing a standalone application is not a straightforward process.
  - You have to deal with different protocols for establishing inter-machine communication.

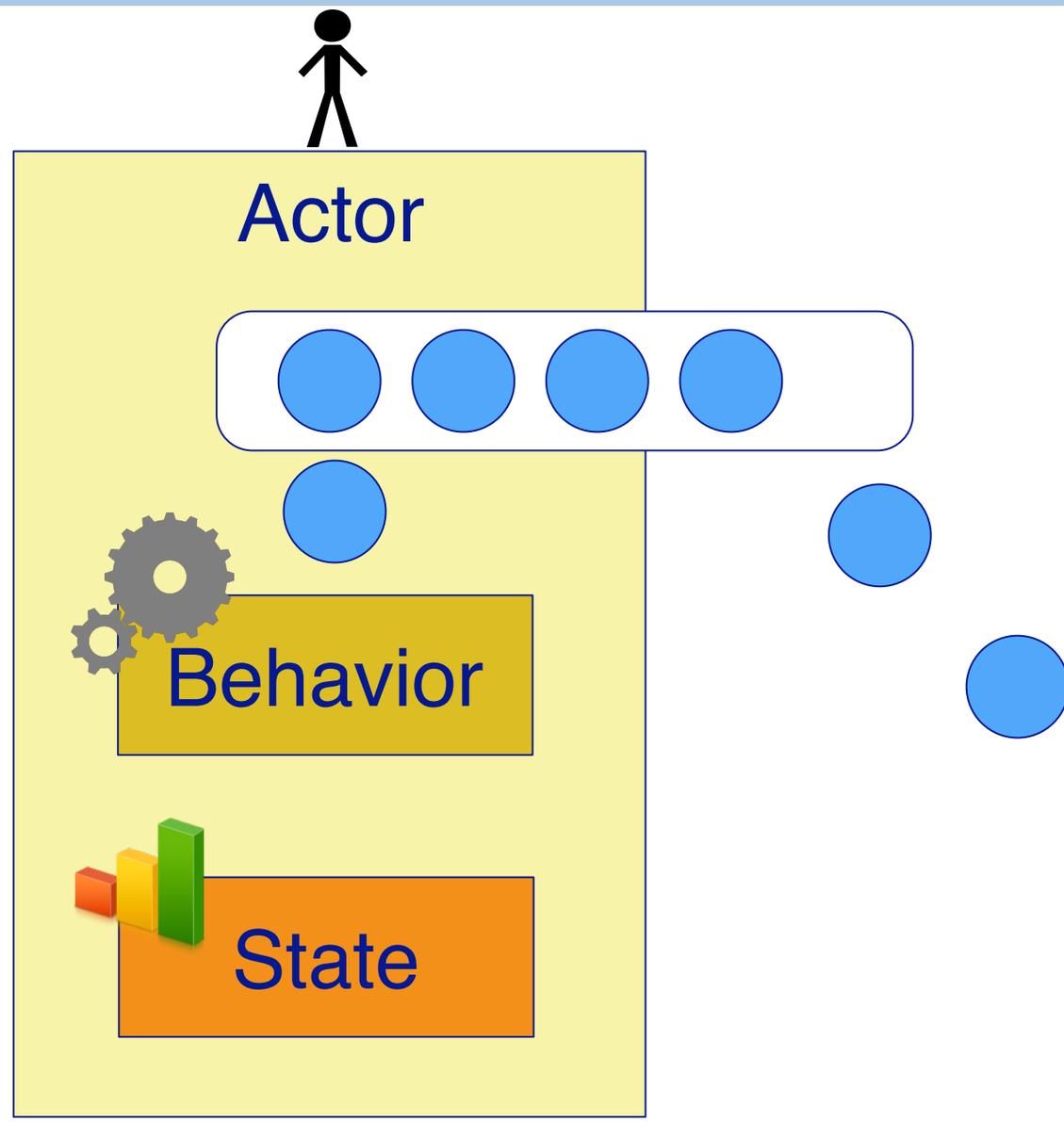
What are actors?

Actors are fundamental units of computation that embody:

- —Processing (behavior)
- —Storage (state)
- —Communication (messages)

When an actor receives a message, it can:

- — create more actors
- — send messages to other actors
- — designate what to do with the next message



Rules of bare-metal actors

> Every actor has an address.

> An actor can process one message at a time.

> Everything in an actor system is an actor.

> Message delivery is *best-effort*

> There are no guarantees that messages are received in the same order they are sent.

> Messages to actors should always be immutable to avoid shared state.

Example 1 - Simple Actor Definition

```
import akka.actor.Actor

class Counter extends Actor {
  var count = 0

  def receive = {
    case "increase" => count += 1
    case "get" => sender() ! count
    case _ => println("Unknown message")
  }
}
```

Example 1 - Simple Actor Definition

```
import akka.actor.Actor

class StatelessCounter extends Actor {
  def count(n:Int): Receive = {
    case "increase" => context.become(count(n+1))
    case "get" => sender() ! n
  }

  def receive = count(0)
}
```

Example 1 - Simple Actor Definition

```
import akka.actor.Actor

class StatelessCounter extends Actor {
  def count(n:Int): Receive = {
    case "increase" => context.become(count(n+1))
    case "get" => sender() ! n
  }

  def receive = count(0)
}
```

for the next message,
receive = count(n+1)

Example 1 - Simple Actor Definition

```
import akka.actor.Actor

class StatelessCounter extends Actor {
  def count(n:Int): Receive = {
    case "increase" => context.become(count(n+1))
    case "get" => sender() ! n
  }

  def receive = count(0)
}
```

```
type Receive = PartialFunction[Any, Unit]
```

Example 1 - Simple Actor Definition

```
import akka.actor.{ActorSystem, Props}

object Application extends App {
  val system = ActorSystem("SimpleActorSystem")
  val counter = system.actorOf(Props[Counter], "counter")
  counter ! "increase"
  counter ! "increase"
  counter ! "get"
}
```

Example 2 - Ping Pong

```
import akka.actor.{Actor, ActorRef, ActorSystem, PoisonPill, Props}
import scala.language.postfixOps
case object Ping
case object Pong
class Pinger extends Actor {
  var countDown = 5
  def receive = {
    case Pong =>
      println(s"${self.path} received pong, count down $countDown")
      if (countDown > 0) {
        countDown -= 1
        sender() ! Ping
      } else {
        sender() ! PoisonPill
        self ! PoisonPill
      }
  }
}
```

Example 2 - Ping Pong

```
class Ponger(pinger: ActorRef) extends Actor {
  def receive = {
    case Ping =>
      println(s"${self.path} received ping")
      pinger ! Pong
  }
}

object PingPongApplication extends App {
  val system = ActorSystem("PingPong")
  val pinger = system.actorOf(Props[Pinger]
                              , "pinger")
  val ponger = system.actorOf(Props(classOf[Ponger], pinger)
                              , "ponger")
  ponger ! Ping
}
```

Example 2 - Ping Pong

```
akka://PingPong/user/ponger received ping
akka://PingPong/user/pinger received pong, count down 5
akka://PingPong/user/ponger received ping
akka://PingPong/user/pinger received pong, count down 4
akka://PingPong/user/ponger received ping
akka://PingPong/user/pinger received pong, count down 3
akka://PingPong/user/ponger received ping
akka://PingPong/user/pinger received pong, count down 2
akka://PingPong/user/ponger received ping
akka://PingPong/user/pinger received pong, count down 1
akka://PingPong/user/ponger received ping
akka://PingPong/user/pinger received pong, count down 0
```

Example 3 - Approximating π using Leibniz formula

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \pi / 4$$

$$\pi = 4 * (1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \ldots)$$

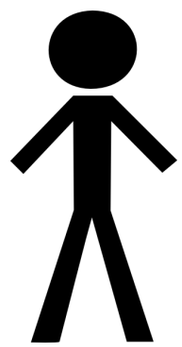
Example 3 - Approximating π using Leibniz formula

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \pi / 4$$

$$\pi = 4 * (1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \ldots)$$

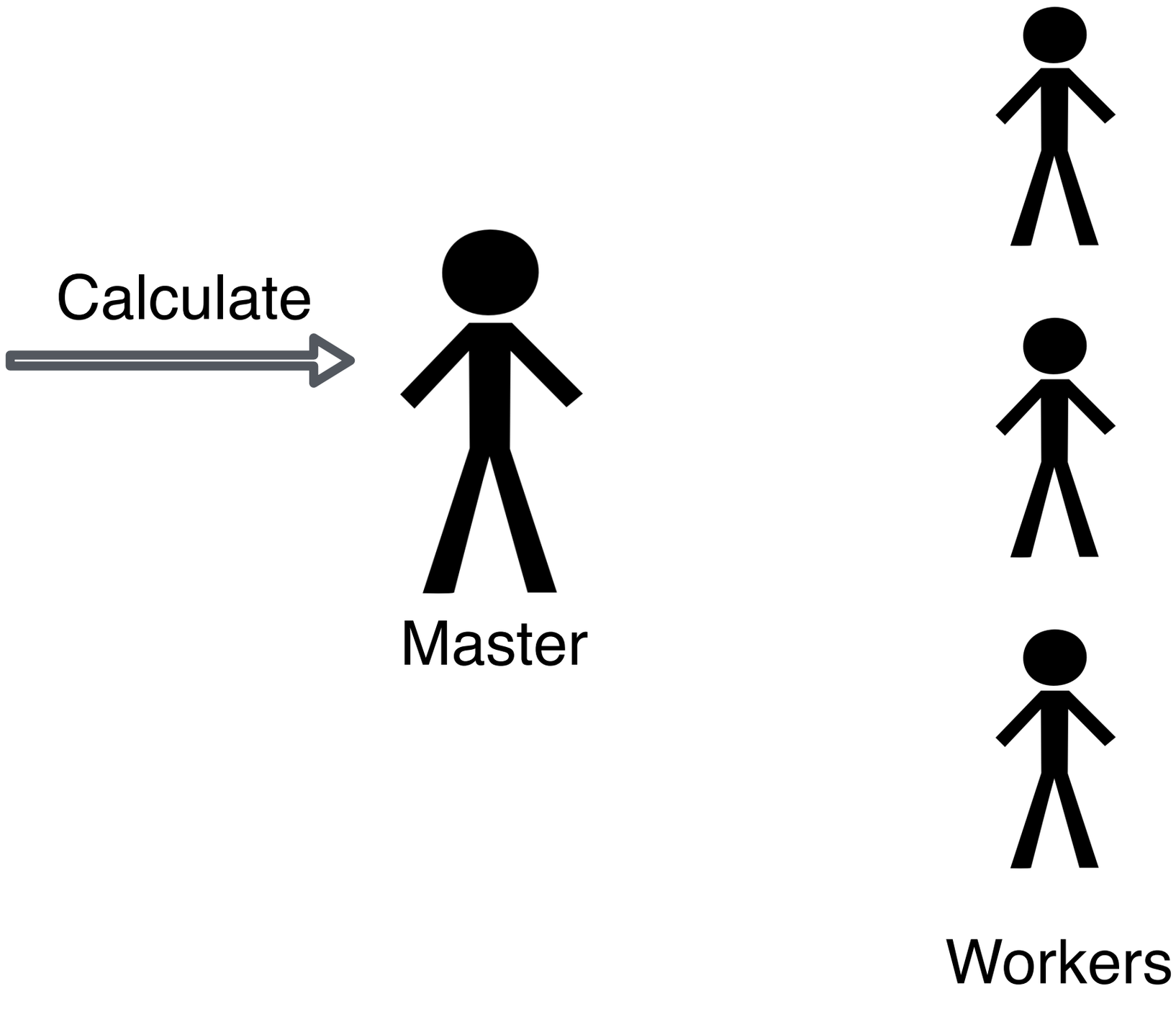
Example 3 - Designing The Actor System

Master

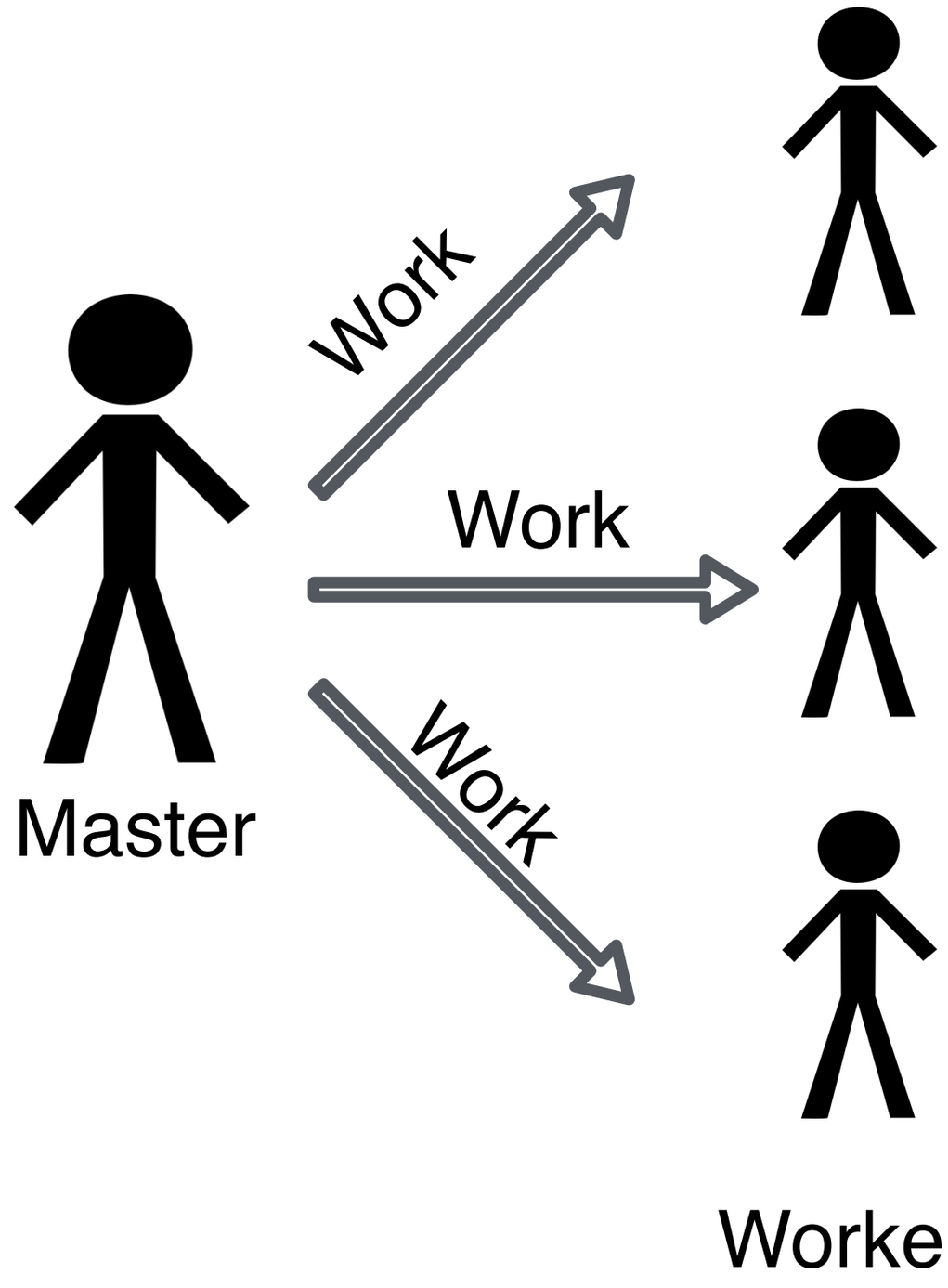


Workers

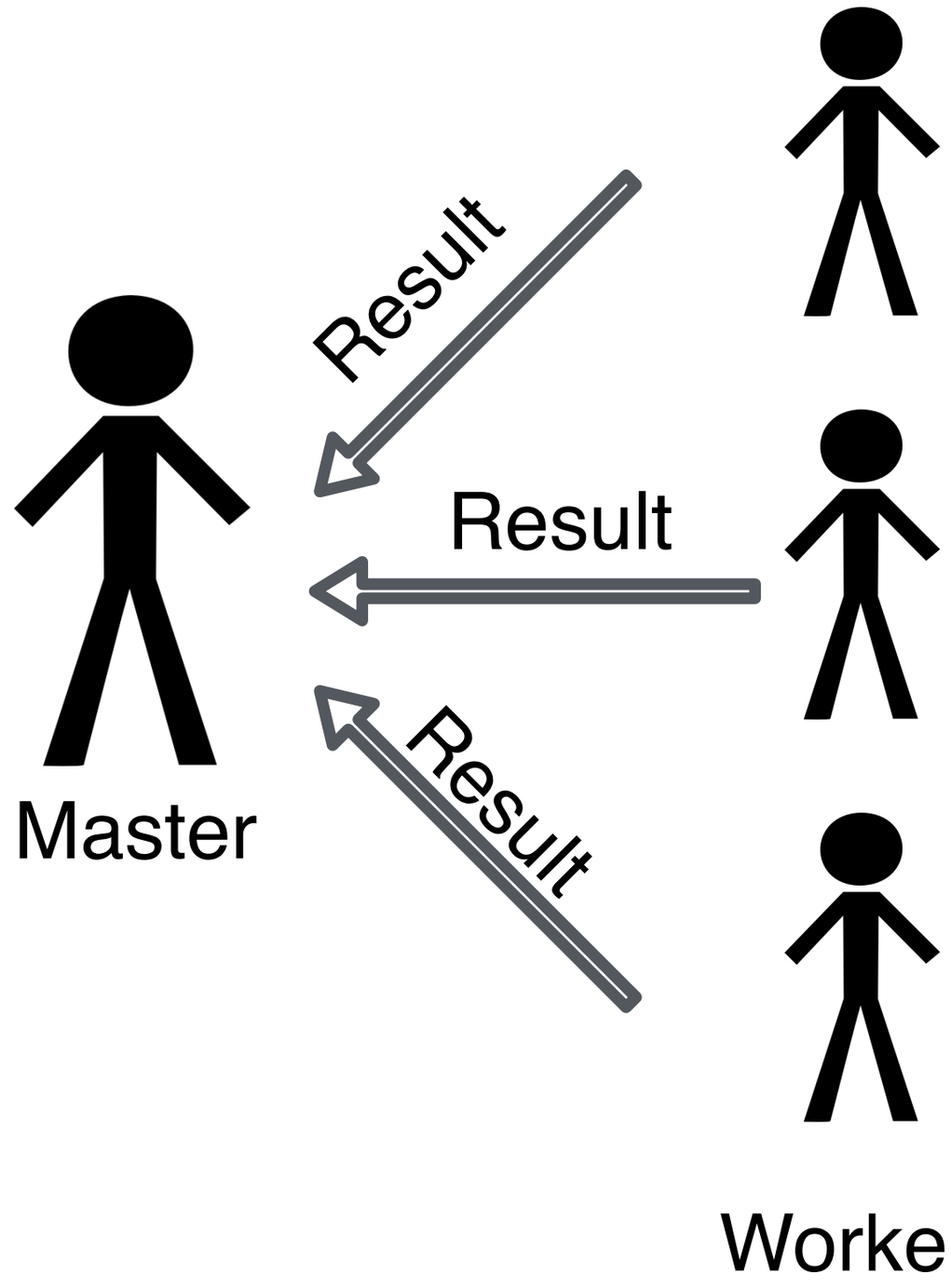
Example 3 - Designing The Actor System



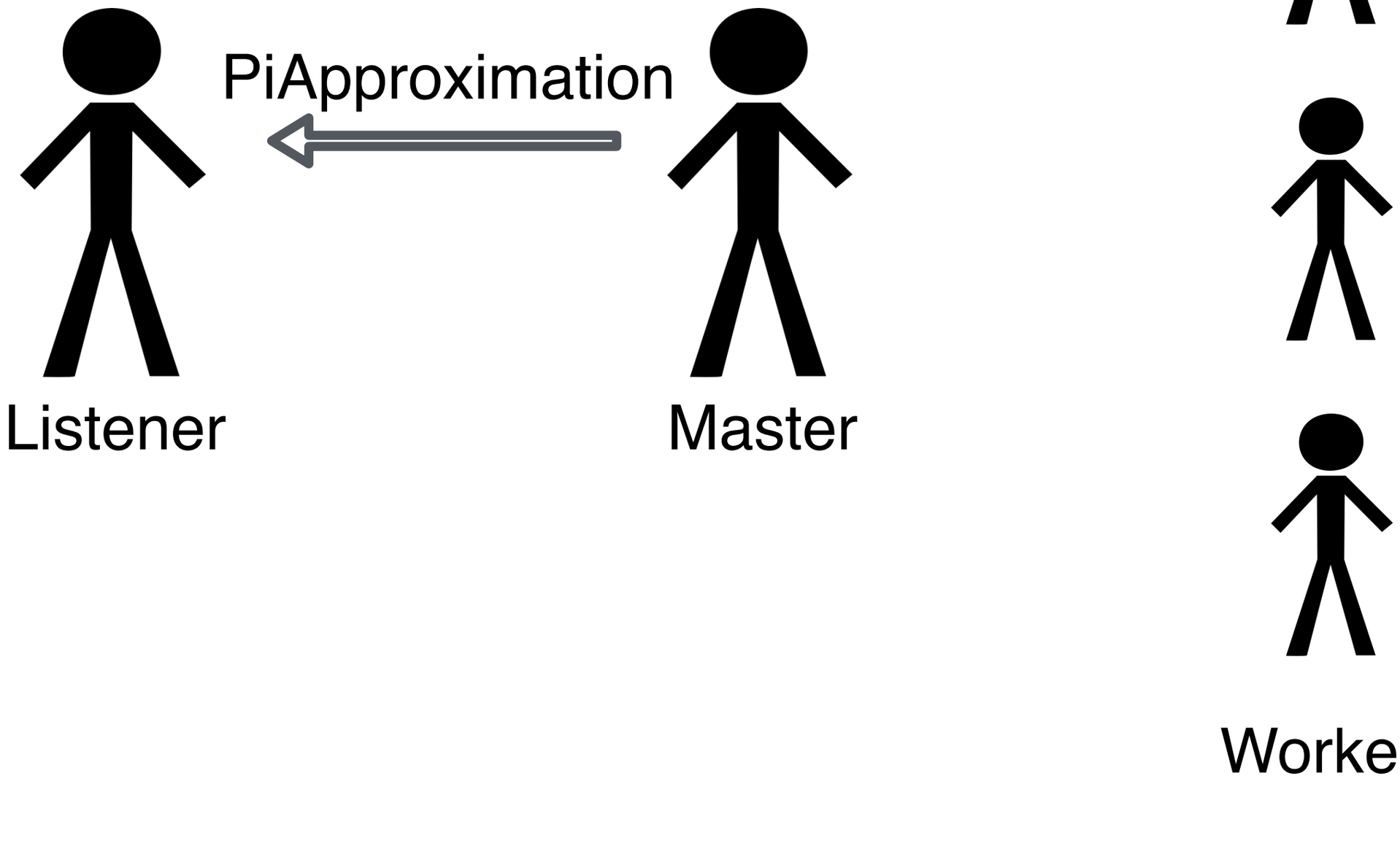
Example 3 - Designing The Actor System



Example 3 - Designing The Actor System



Example 3 - Designing The Actor System

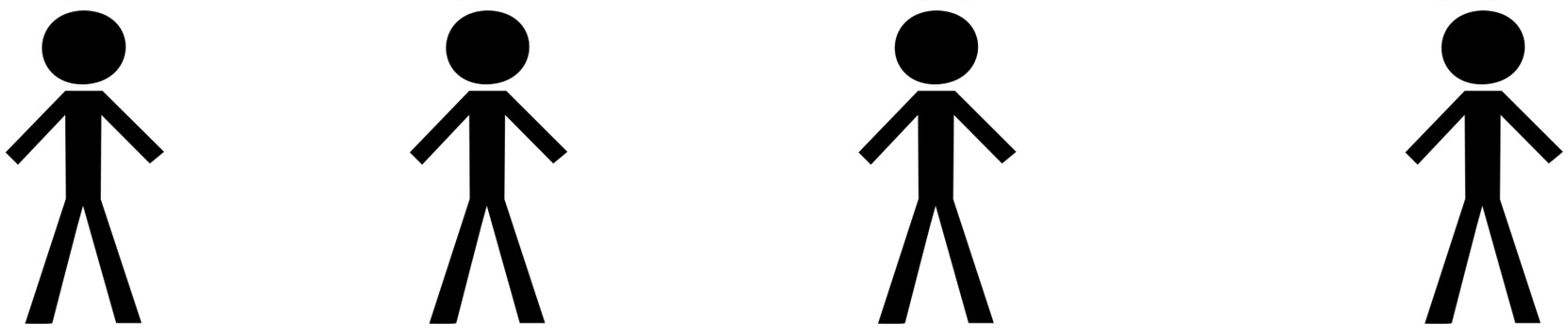


Example 3 - Approximating π using Leibniz formula

$$\pi = 4 * (1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \ldots)$$

Example 3 - Approximating π using Leibniz formula

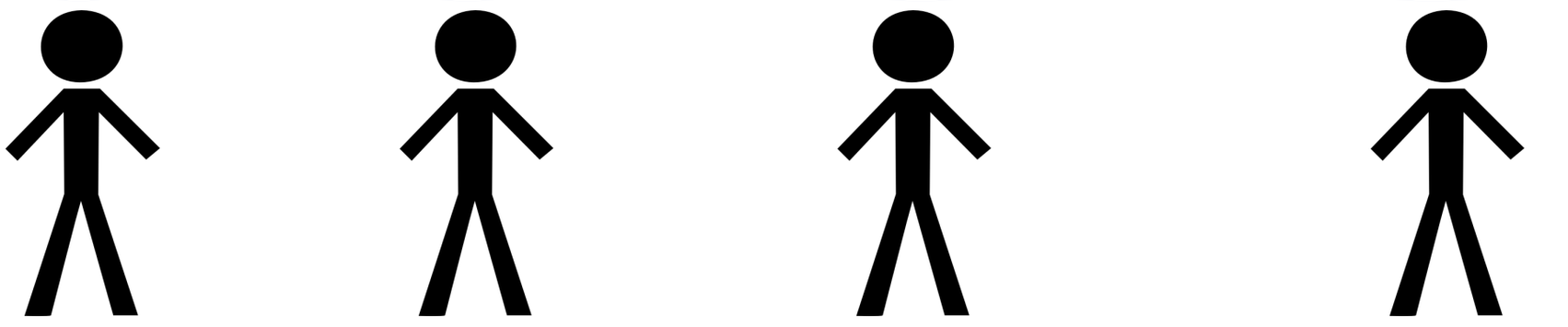
$$\pi = 4 * (1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \ldots)$$



Example 3 - Approximating π using Leibniz formula

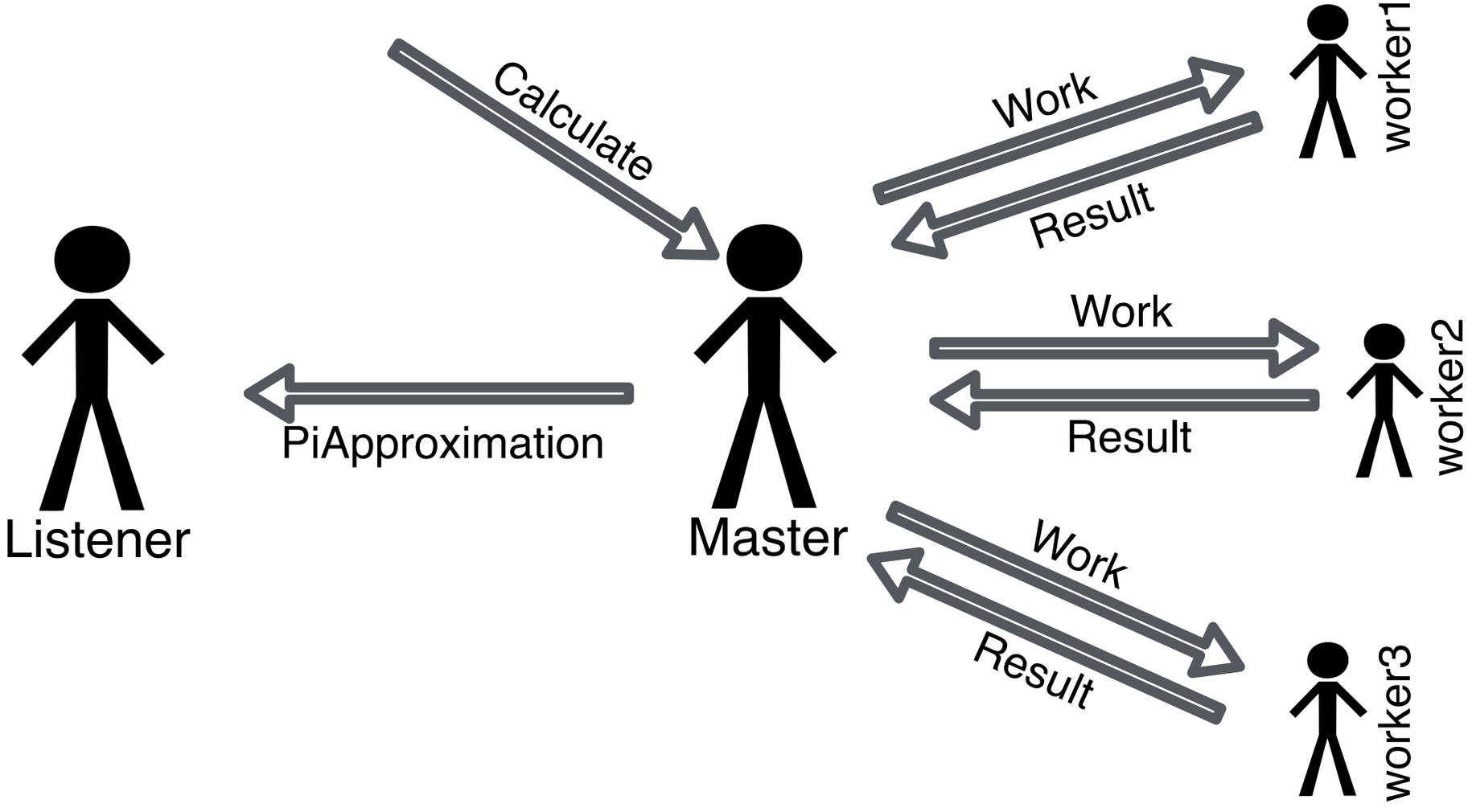
System configurations:
Number of workers
Number of messages
Number of elements per message

$$\pi = 4 * (1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \ldots)$$



Example 3 - Messages

```
sealed trait PiMessage
case object Calculate extends PiMessage
case class Work(start: Int, nrOfElements: Int) extends PiMessage
case class Result(value: Double) extends PiMessage
case class PiApproximation(pi: Double, duration: Duration)
```



Example 3 - Worker

```
class Worker extends Actor {

  def calculatePiFor(start: Int, nrOfElements: Int): Double ={
    var acc = 0.0
    for (i <- start until (start + nrOfElements))
      acc += 4.0 * (1 - (i % 2) * 2) / (2 * i + 1)
    acc
  }

  def receive = {
    case Work(start, nrOfElements)=>
      sender ! Result(calculatePiFor(start, nrOfElements))
  }
}
```

Example 3 - Worker

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \pi / 4$$

```
class Worker extends Actor {

  def calculatePiFor(start: Int, nrOfElements: Int): Double ={
    var acc = 0.0
    for (i <- start until (start + nrOfElements))
      acc += 4.0 * (1 - (i % 2) * 2) / (2 * i + 1)
    acc
  }

  def receive = {
    case Work(start, nrOfElements)=>
      sender ! Result(calculatePiFor(start, nrOfElements))
  }
}
```

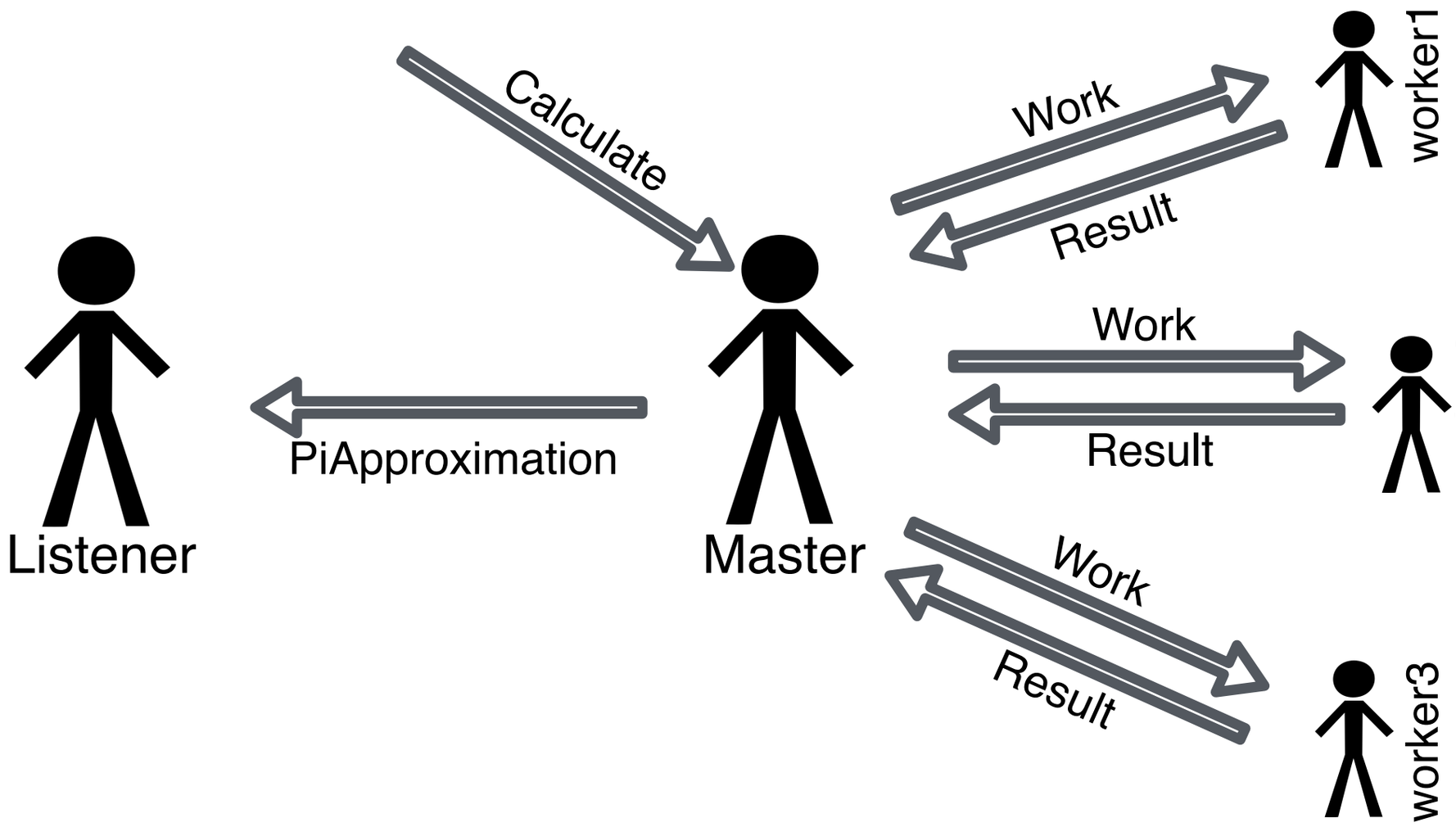
$(-1)^n$ $2n+1$

Example 3 - Master 1/2

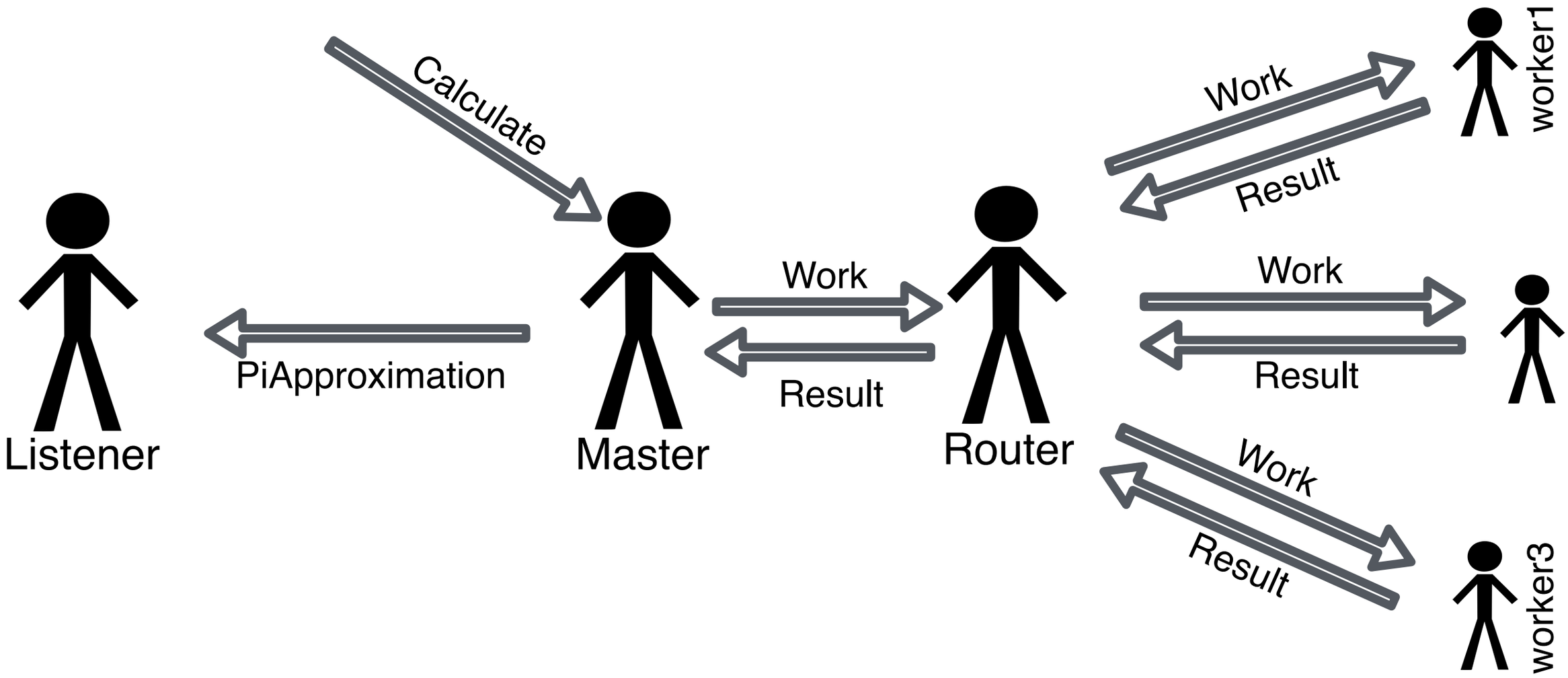
```
class Master(nrOfWorkers: Int,
             nrOfMessages: Int,
             nrOfElements: Int,
             listener: ActorRef) extends Actor {

  var pi: Double = 0
  var nrOfResults: Int = 0
  var start: Long = _
  val workerRouter = context.actorOf(
    Props[Worker].withRouter(RoundRobinRouter(nrOfWorkers)),
    name = "workerRouter")
```

Example 3 - Master 1/2



Example 3 - Master 1/2



Example 3 - Master 1/2

```
class Master(nrOfWorkers: Int,
             nrOfMessages: Int,
             nrOfElements: Int,
             listener: ActorRef) extends Actor {

  var pi: Double = 0
  var nrOfResults: Int = 0
  var start: Long = _
  val workerRouter = context.actorOf(
    Props[Worker].withRouter(RoundRobinRouter(nrOfWorkers)),
    name = "workerRouter")
```

Example 3 - Master 2/2

```
def receive = {
  // handle messages ...
  case Calculate =>
    start= System.currentTimeMillis
    for (i <- 0 until nrOfMessages)
      workerRouter ! Work(i * nrOfElements, nrOfElements)
  case Result(value) =>
    pi += value
    nrOfResults += 1
    if (nrOfResults == nrOfMessages) {
      // Send the result to the listener
      val currentTime=System.currentTimeMillis
      val executionTime=Duration.create(currentTime - start, TimeUnit.MILLISECONDS)
      listener ! PiApproximation(pi, executionTime)
      // Stops this actor and all its supervised children
      context.stop(self)
    }
}
```

Example 3 - Listener

```
class Listener extends Actor {
  def receive = {
    case PiApproximation(pi, duration) ⇒
      println("\n\tPi approximation: \t\t%s\n\tCalculation time: \t%s"
        .format(pi, duration))
      context.system.shutdown()
  }
}
```

Example 3 - Putting It Together

```
object Pi extends App {

  calculate(nrOfWorkers = 4, nrOfElements = 10000, nrOfMessages = 10000)

  // actors and messages ...

  def calculate(nrOfWorkers: Int, nrOfElements: Int, nrOfMessages: Int) {
    // Create an Akka system
    val system = ActorSystem("PiSystem")

    // create the result listener, which will print the result and shutdown the system
    val listener = system.actorOf(Props[Listener], name = "listener")

    // create the master
    val master = system.actorOf(Props(new Master(
      nrOfWorkers, nrOfMessages, nrOfElements, listener)),
      name = "master")

    // start the calculation
    master ! Calculate
  }
}
```

With actors …

> It is easier to build scalable software

- Scalability is a deployment problem

> It is a convenient abstraction for modeling

- Similarly to OOP

> Fault tolerance can be implemented on top

- Actors can fail and get replaced at run time

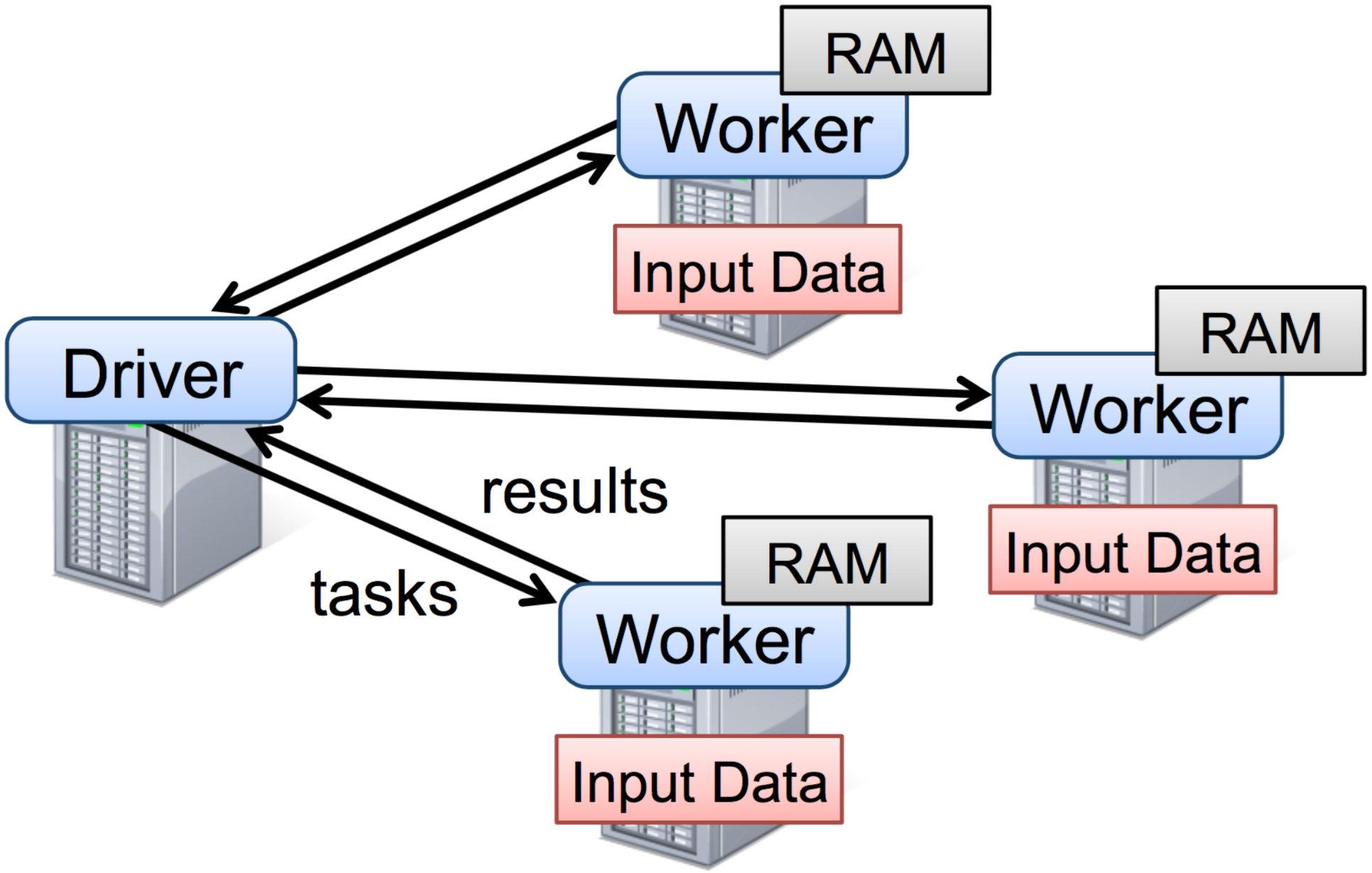
1- **Crash course in Scala**
- Classes
- Objects

2- **Actors**
- The Actor Model
- Examples I, II, III, IV

3- **Apache Spark**
- RDD & DAG
- Word Count Example

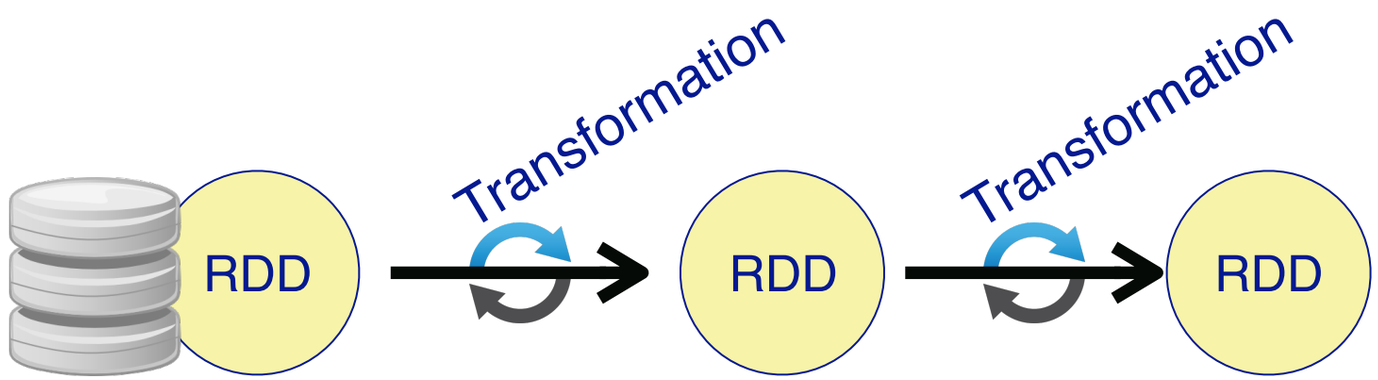
```
05:52:19  Info        AboutLog 170       0         "Changing App Context: {0: 1} -> {0: 4097}"
05:52:19  Info        Status   163       0         "INSTALL SPACE REQUIRED: {0: 10169266} KB; SPACE AVAILABLE: {1: 123510776} KB; LOC: {2: C:\}"
05:52:19  Info        Step     25        55502     "User advanced from '{0: Connecting}' page to '{1: Summary}' page."
05:52:19  Info        Selection 38       55612     "The following lines show the default selections for the Summary page when it first appeared."
05:52:19  Info        Selection 99       0         "This is a new installation of 2016 SP1.0."
05:52:19  Info        Selection 102      0         "[Download Options: {Operation:=Install only (do not download) (Enabled: 1)}, {Background downloader:=Do not use (Enabled: 1)}] (Enable
05:52:19  Info        Selection 103      0         "[Installation Location: {Installation location:=C:\Program Files\SOLIDWORKS Corp (Enabled: 1)}, {Install from:=G:\SOLIDWORKS DOWNLO
05:52:19  Info        Selection 104      0         "[Toolbox/Hole Wizard Options: {Toolbox installation location:=C:\SOLIDWORKS Data (Enabled: 1)}, {Toolbox installation method:=New Too
05:52:19  Info        Selection 105      0         "Estimated installation size: 9.7 GB"
05:52:23  Info        Step     25        55502     "User advanced from '{0: Summary}' page to '{1: Products To Install}' page."
05:53:24  Info        Selection 37       55611     "The user clicked the button representing the {0: OK} operation on the {1: Products To Install} page."
05:53:24  Info        Selection 36       55610     "The user changed the installation action of {0: PhotoView 360 Network Render Client} to: {1: Install PhotoView 360 Network Render Client
05:53:24  Info        Selection 36       55610     "The user changed the installation action of {0: SOLIDWORKS Electrical} to: {1: Do not install SOLIDWORKS Electrical 2016 SP1.0}"
05:53:24  Info        Selection 36       55610     "The user changed the installation action of {0: API Tools} to: {1: API Tools 2016 SP1.0 (install manually)}"
05:53:24  Info        Selection 36       55610     "The user changed the installation action of {0: SOLIDWORKS Workgroup PDM x64 Client Add-in} to: {1: Do not install SOLIDWORKS Work
05:53:24  Info        Selection 36       55610     "The user changed the installation action of {0: SOLIDWORKS Workgroup PDM Add-in} to: {1: Do not install SOLIDWORKS Workgroup PDM
05:53:24  Info        Status   163       0         "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
05:53:25  Info        Status   163       0         "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
05:53:25  Info        Step     25        55502     "User advanced from '{0: Products To Install}' page to '{1: Summary}' page."
05:53:27  Info        Step     25        55502     "User advanced from '{0: Summary}' page to '{1: Download Options}' page."
05:53:33  Info        Selection 37       55611     "The user clicked the button representing the {0: OK} operation on the {1: Download Options} page."
05:53:33  Info        Step     25        55502     "User advanced from '{0: Download Options}' page to '{1: Summary}' page."
05:53:37  Info        Step     25        55502     "User advanced from '{0: Summary}' page to '{1: Installation Location}' page."
05:53:48  Warning     Message  9         55200     "Dialog Shown: {0: C:\Program Files\SOLIDWORKS 2016 This path does not exist. Do you want to create it now?}.  User selected: {1: Yes}"
05:53:48  Info        Status   163       0         "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
05:53:48  Info        Step     23        55500     "Page '{0: Installation Location}' is being reshown or refreshed."
05:53:50  Info        Selection 37       55611     "The user clicked the button representing the {0: OK} operation on the {1: Installation Location} page."
05:53:50  Info        Status   163       0         "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
05:53:50  Info        Status   163       0         "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
05:53:50  Info        Step     25        55502     "User advanced from '{0: Installation Location}' page to '{1: Summary}' page."
05:53:56  Info        Step     25        55502     "User advanced from '{0: Summary}' page to '{1: Toolbox Options}' page."
05:54:37  Info        Step     23        55500     "Page '{0: Toolbox Options}' is being reshown or refreshed."
05:54:43  Info        Selection 37       55611     "The user clicked the button representing the {0: OK} operation on the {1: Toolbox Options} page."
05:54:44  Info        Status   163       0         "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506648} KB; LOC: {2: C:\}"
05:54:44  Info        Step     25        55502     "User advanced from '{0: Toolbox Options}' page to '{1: Summary}' page."
05:54:58  Info        Selection 37       55611     "The user clicked the button representing the {0: <U>N</U>ext} operation on the {1: Summary} page."
05:54:58  Info        Selection 39       55613     "The following lines show the selections for the Summary page when the user selected to begin a download or installation."
05:54:58  Info        Selection 99       0         "This is a new installation of 2016 SP1.0."
05:54:58  Info        Selection 102      0         "[Download Options: {Operation:=Install only (do not download) (Enabled: 1)}, {Background downloader:=Do not use (Enabled: 1)}] (Enable
05:54:58  Info        Selection 103      0         "[Installation Location: {Installation location:=C:\Program Files\SOLIDWORKS 2016 (Enabled: 1)}, {Install from:=G:\SOLIDWORKS DOWNLO
05:54:58  Info        Selection 104      0         "[Toolbox/Hole Wizard Options: {Toolbox installation location:=C:\Program Files\SOLIDWORKS 2016\SOLIDWORKS 2016 DATA (Enabled: 1)}
05:54:58  Info        Selection 105      0         "Estimated installation size: 9.0 GB"
05:54:58  Info        Selection 43       55617     "The source folder that will be used is: {0: G:\SOLIDWORKS DOWNLOADS\SOLIDWORKS 2016 x64 SP01}"
05:54:58  Info        Selection 44       55617     "The Installation Manager Source Folder that will be used is: {0: G:\SOLIDWORKS DOWNLOADS\SOLIDWORKS 2016 x64 SP01\sldIM}"
05:54:59  Info        Step     25        55502     "User advanced from '{0: Summary}' page to '{1: Install Progress}' page."
05:54:59  Info        Selection 107      0         "Writing serial numbers to the registry..."
```

High-Level Spark Architecture



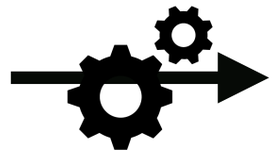
Resilient Distributed Dataset

- RDD is a partitioned collection of records
- RDD is read-only (immutable)
- RDDs can only be created through deterministic operations on:
  - data from stable storage, or
  - other RDDs

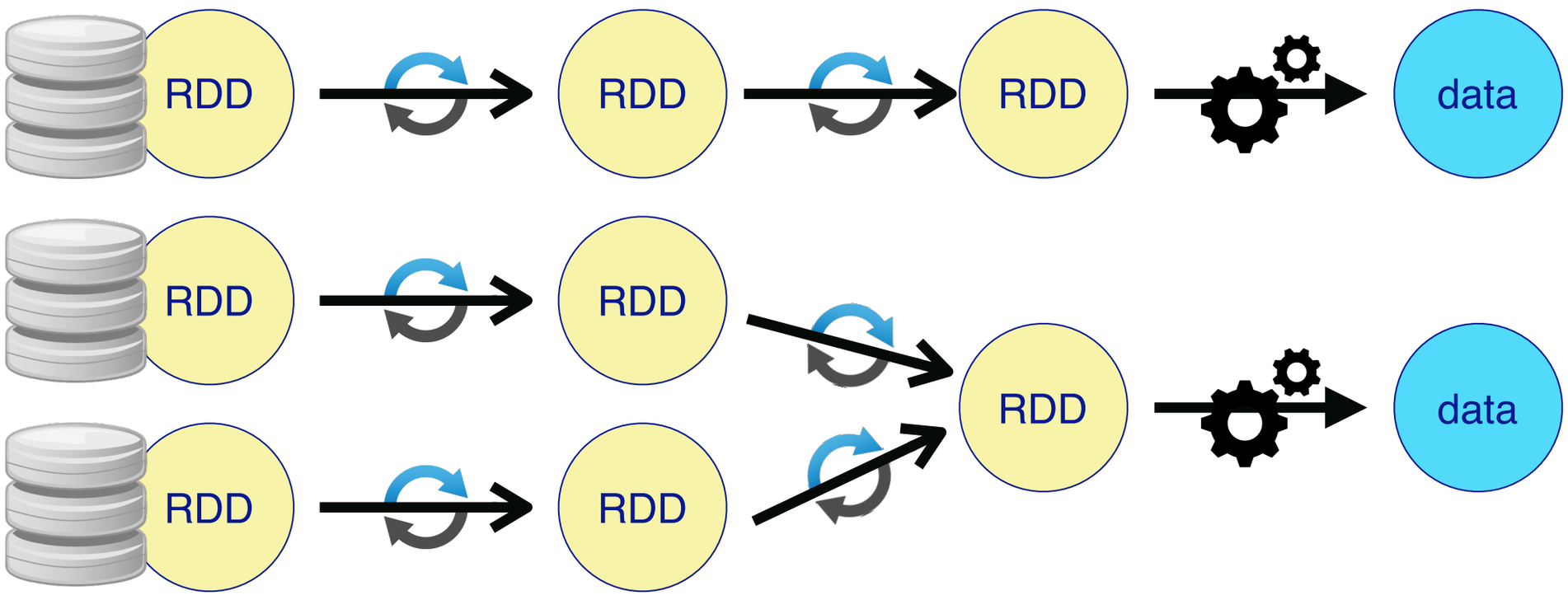


Directed Acyclic Graph (DAG)

Transformation

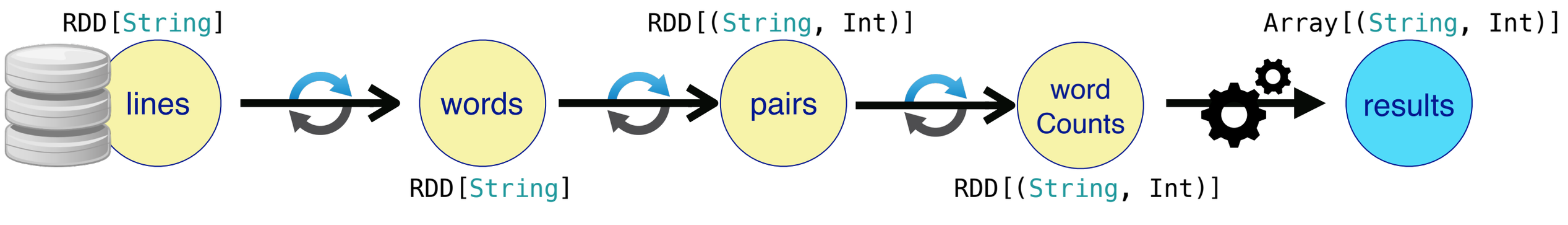


Action



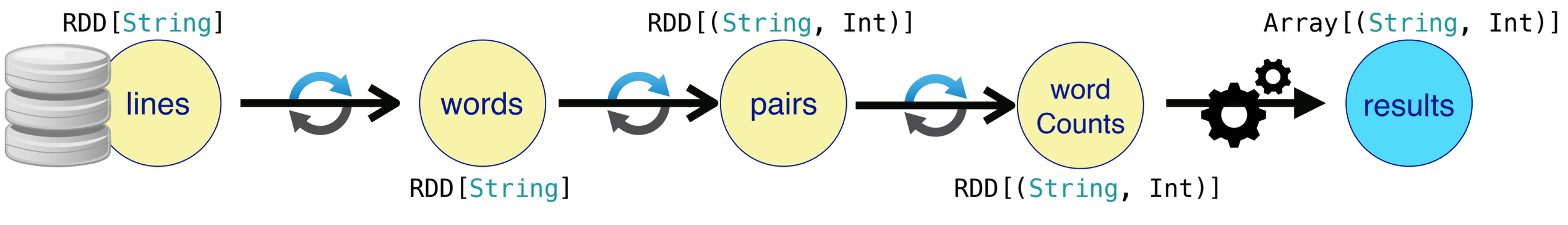
Word Count Example

```
val conf = new SparkConf()
                .setMaster(“local[2]")
                .setAppName("WordCounter")
val sparkContext = new SparkContext(conf)
val lines = sparkContext.textFile("/var/log/commerce.log")
val words = lines.flatMap(line => line.split(" "))
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey((count1,count2)=>count1+count2)
val results = wordCounts.take(100)
results.foreach(resultPair => println(resultPair))
```



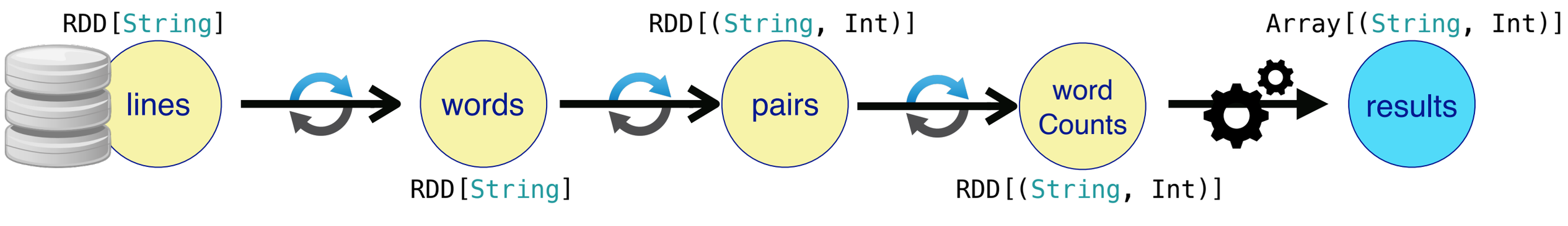
Word Count Example

```
val conf = new SparkConf()
                .setMaster(“local[2]")
                .setAppName("WordCounter")
val sparkContext = new SparkContext(conf)
val lines = sparkContext.textFile("/var/log/commerce.log")
val words = lines.flatMap(line => line.split(" "))
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey((count1,count2)=>count1+count2)
val results = wordCounts.take(100)
results.foreach(resultPair => println(resultPair))
```



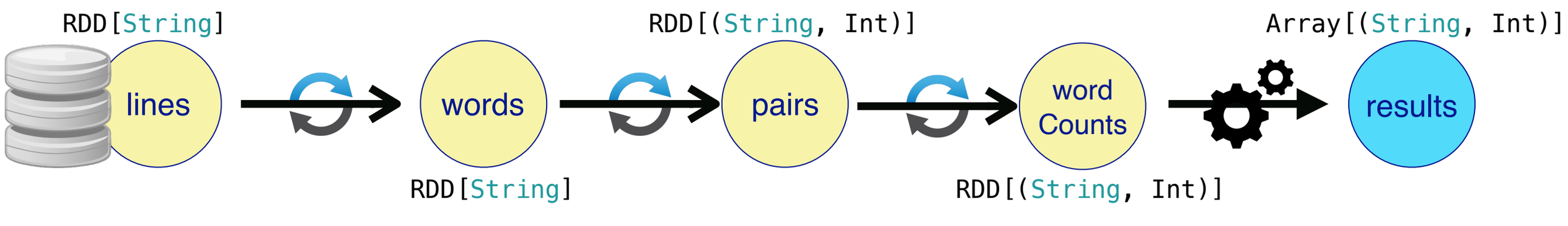
Word Count Example

```
val conf = new SparkConf()
               .setMaster(“local[2]")
               .setAppName("WordCounter")
val sparkContext = new SparkContext(conf)
val lines = sparkContext.textFile("/var/log/commerce.log")
val words = lines.flatMap(line => line.split(" "))
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey((count1,count2)=>count1+count2)
val results = wordCounts.take(100)
results.foreach(resultPair => println(resultPair))
```



Word Count Example

```
val conf = new SparkConf()
               .setMaster(“local[2]")
               .setAppName("WordCounter")
val sparkContext = new SparkContext(conf)
val lines = sparkContext.textFile("/var/log/commerce.log")
val words = lines.flatMap(line => line.split(" "))
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey((count1,count2)=>count1+count2)
val results = wordCounts.take(100)
results.foreach(resultPair => println(resultPair))
```



References

> The original actor paper [Hewitt73]:
"A universal modular ACTOR formalism for artificial intelligence"

> Hewitt explaining actors:
https://channel9.msdn.com/Shows/Going+Deep/Hewitt-Meijer-and-Szyperski-The-Actor-Model-everything-you-wanted-to-know-but-were-afraid-to-ask

> Akka Tutorial: https://doc.akka.io/docs/akka/2.0/intro/getting-started-first-scala.html

> Scaling up and out with actors: https://www.youtube.com/watch?v=3jbqTxstlC4

> RDD original paper: *"Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing"*

Exercise

- Implement a naïve actor framework in Java where:
  - each actor is a monitor Java class
  - each actor has a mailbox where messages from other actors are saved
  - messages are asynchronous

- Using your framework, reimplement the three examples in the lecture.