01: Getting Started

Installation

hands-on lab: 20 min

Installation:

Let's get started using Apache Spark,
in just four easy steps…

spark.apache.org/docs/latest/

(for class, please copy from the USB sticks)

Step 1: *Install Java JDK 6/7 on MacOSX or Windows*

oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

- follow the license agreement instructions
- then click the download for your OS
- need JDK instead of JRE (for Maven, etc.)

(for class, please copy from the USB sticks)

Step 1: *Install Java JDK 6/7 on Linux*

this is much simpler on Linux…

```
sudo apt-get -y install openjdk-7-jdk
```

Step 2: *Download Spark*

we'll be using Spark 1.0.0
see **spark.apache.org/downloads.html**

1. download this URL with a browser
2. double click the archive file to open it
3. connect into the newly created directory

(for class, please copy from the USB sticks)

Step 3: *Run Spark Shell*

we'll run Spark's interactive shell…

```
./bin/spark-shell
```

then from the "scala>" REPL prompt,
let's create some data…

```
val data = 1 to 10000
```

Step 4: *Create an RDD*

create an **RDD** based on that data…

```
val distData = sc.parallelize(data)
```

then use a filter to select values less than 10…

```
distData.filter(_ < 10).collect()
```

Step 4: *Create an RDD*

create an

```
val distData = sc.parallelize(data)
```

then use a filter to select values less than 10 ...

```
d
```

Checkpoint:
what do you get for results?

gist.github.com/ceteri/f2c3486062c9610eac1d#file-01-repl-txt

Installation: *Optional Downloads: Python*

For Python 2.7, check out *Anaconda* by Continuum Analytics for a full-featured platform:

store.continuum.io/cshop/anaconda/

Anaconda

Installation: *Optional Downloads: Maven*

Java builds later also require Maven, which you can download at:

maven.apache.org/download.cgi

maven

03: Getting Started

Spark Deconstructed

lecture: 20 min

Spark Deconstructed:

Let’s spend a few minutes on this Scala thing…

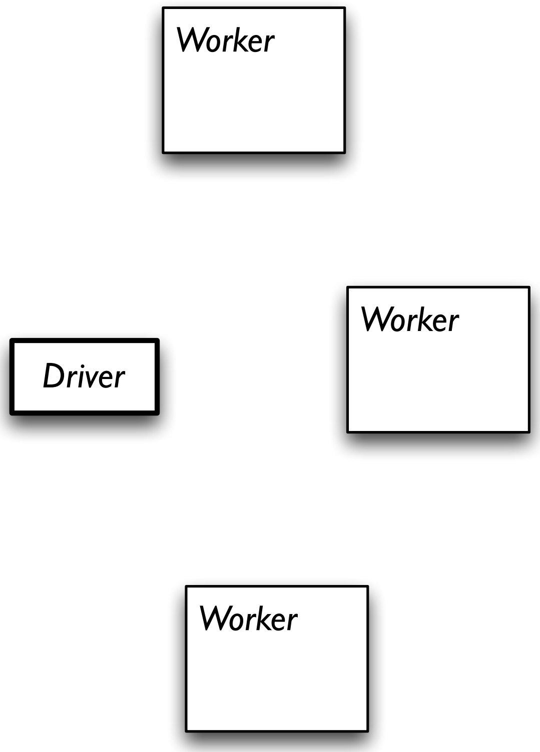
scala-lang.org/

Spark Deconstructed: *Log Mining Example*

```
// load error messages from a log into memory
// then interactively search for various patterns
// https://gist.github.com/ceteri/8ae5b9509a08c08a1132

// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

Spark Deconstructed: *Log Mining Example*

We start with Spark running on a cluster…
submitting code to be evaluated on it:



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

discussing the other part

Spark Deconstructed: *Log Mining Example*

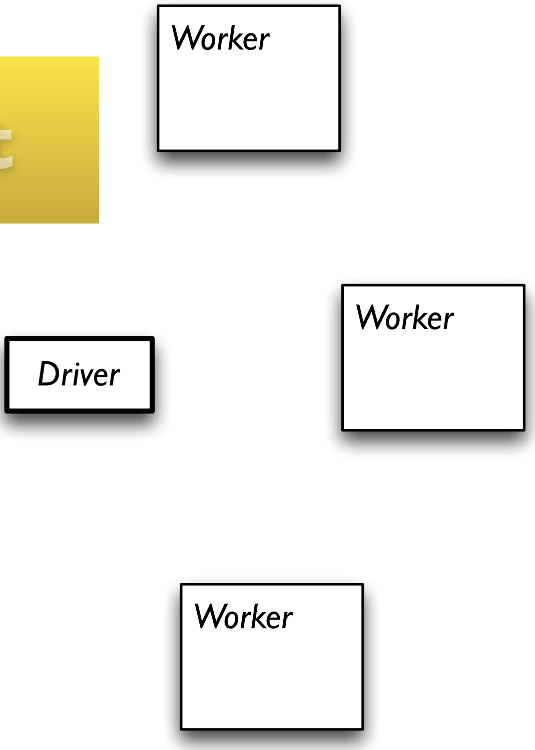
At this point, take a look at the transformed RDD *operator graph*:

```
scala> messages.toDebugString
res5: String =
MappedRDD[4] at map at <console>:16 (3 partitions)
  MappedRDD[3] at map at <console>:16 (3 partitions)
    FilteredRDD[2] at filter at <console>:14 (3 partitions)
      MappedRDD[1] at textFile at <console>:12 (3 partitions)
        HadoopRDD[0] at textFile at <console>:12 (3 partitions)
```

Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

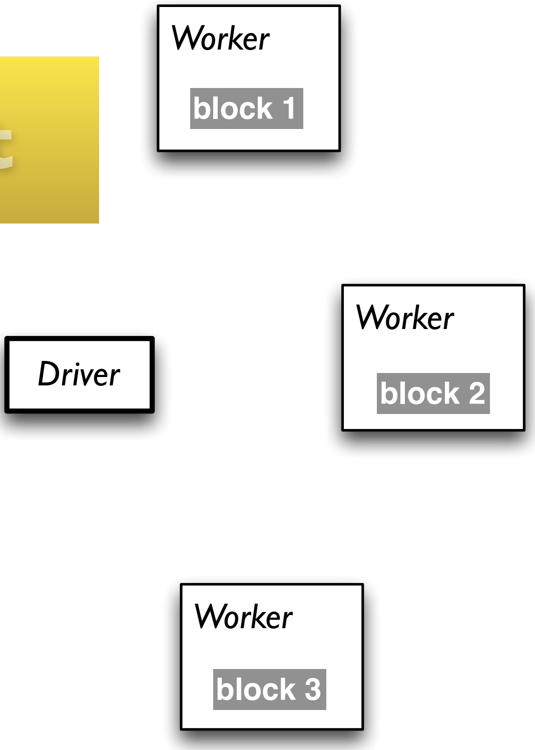
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

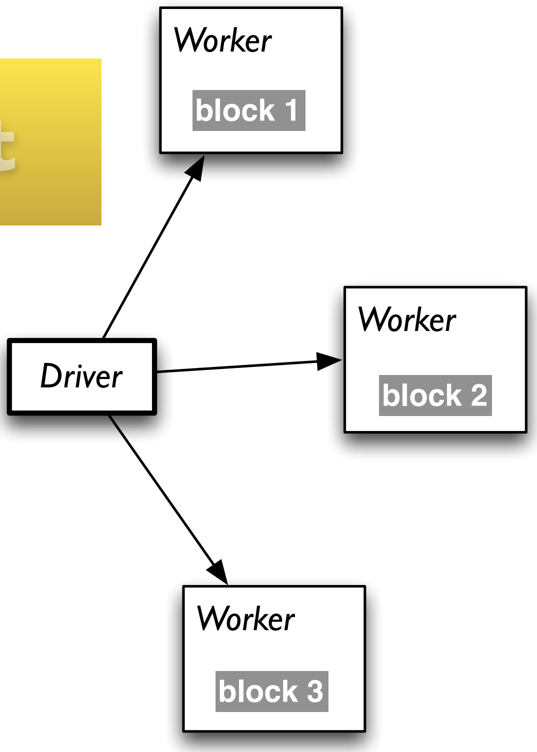
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

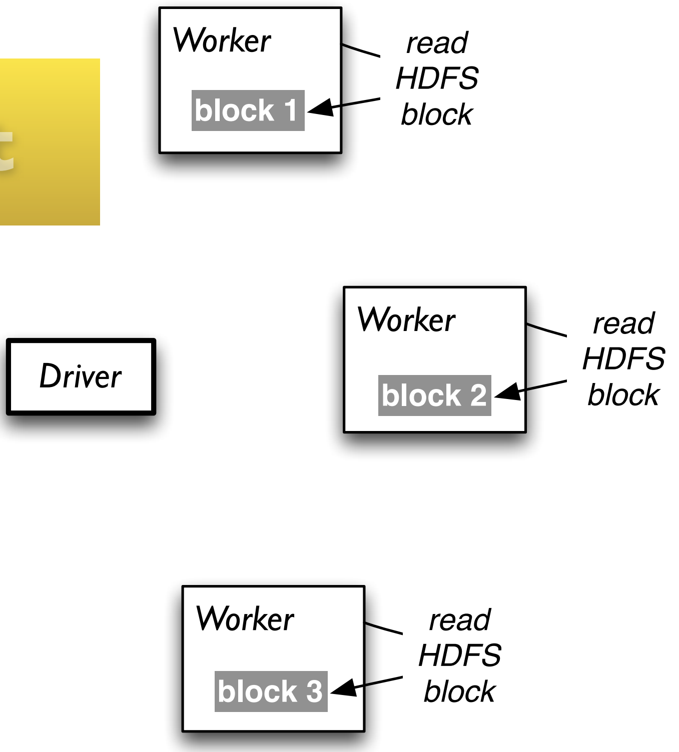
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

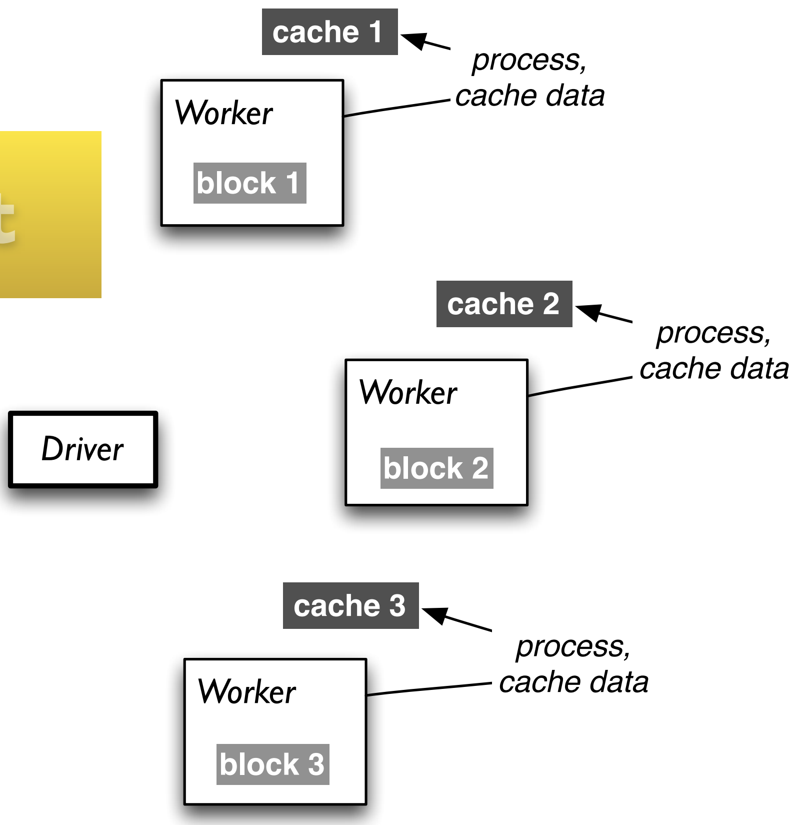
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

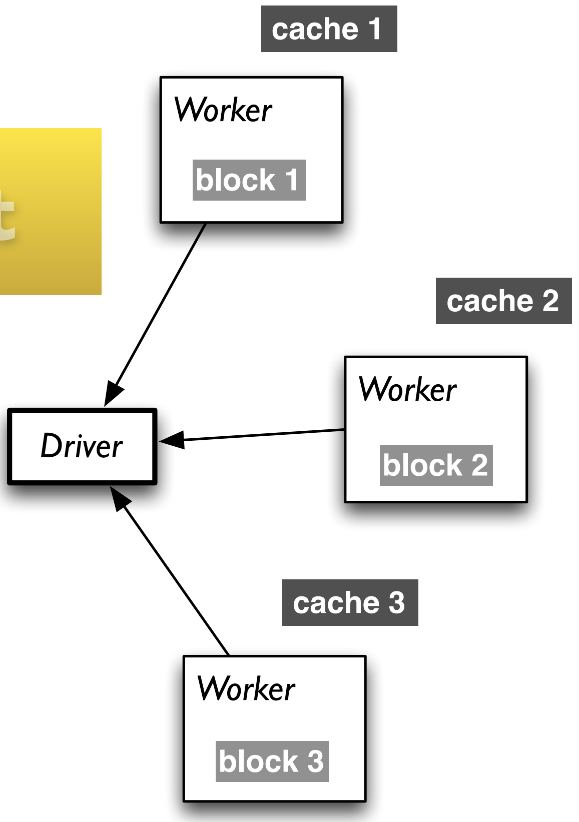
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

discussing the other part

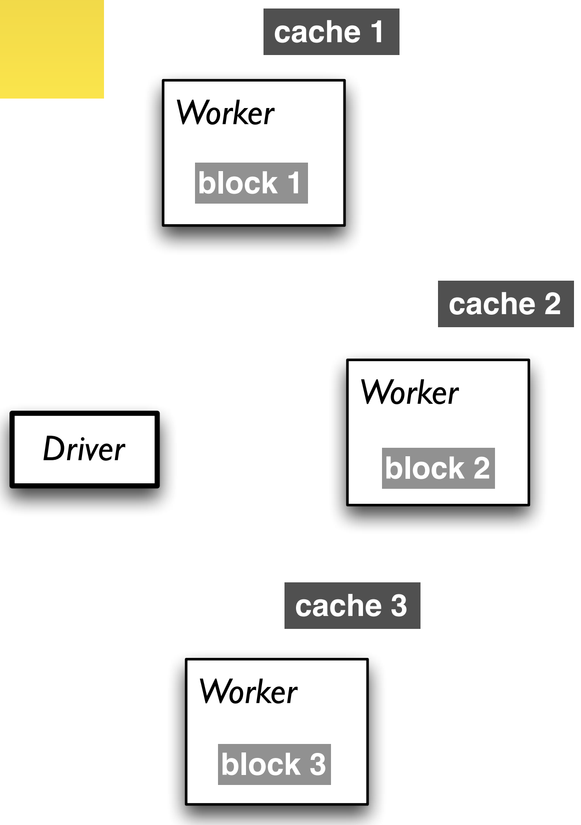


Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()
```

discussing the other part

```
// action 2
messages.filter(_.contains("php")).count()
```

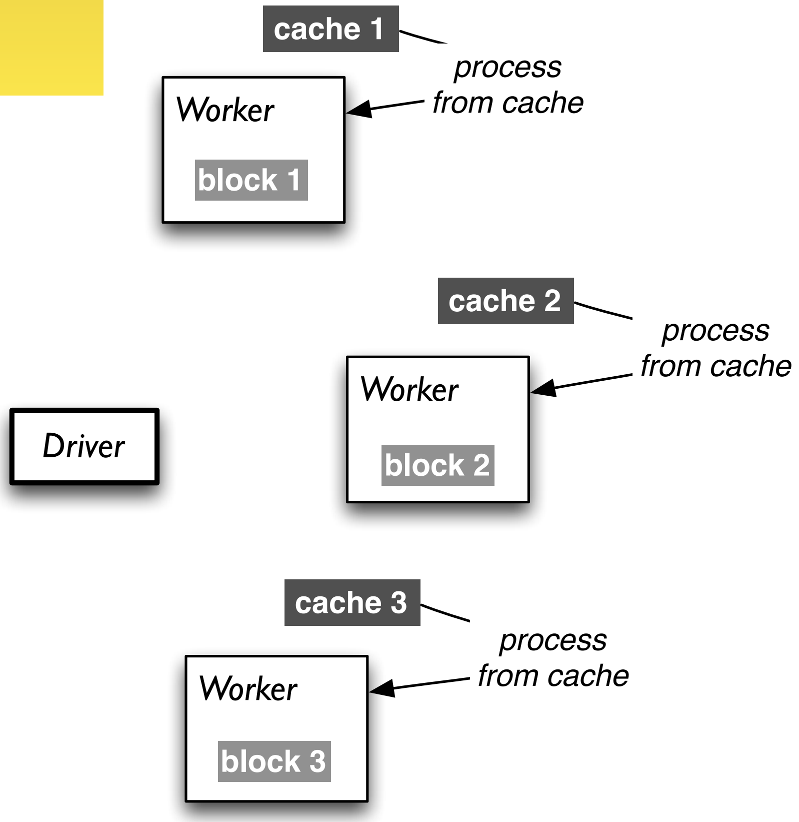


Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()
```

discussing the other part

```
// action 2
messages.filter(_.contains("php")).count()
```

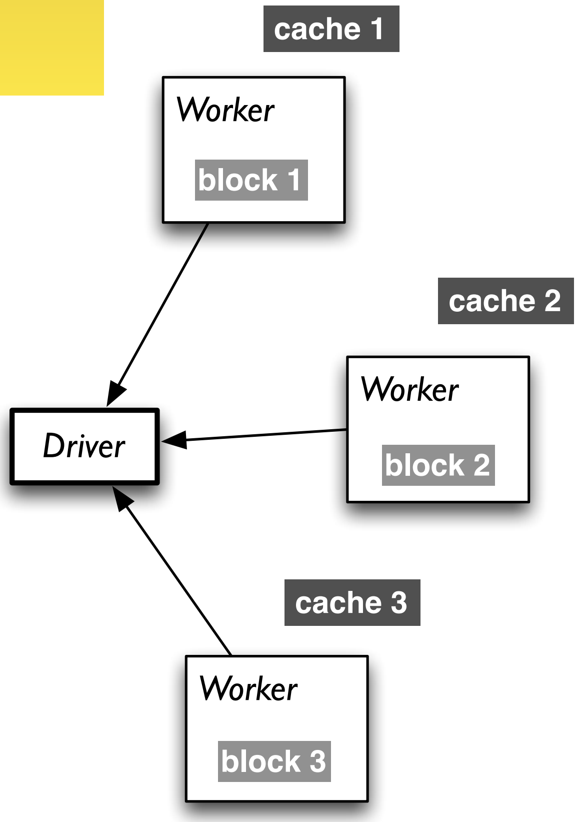


Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()
```

discussing the other part

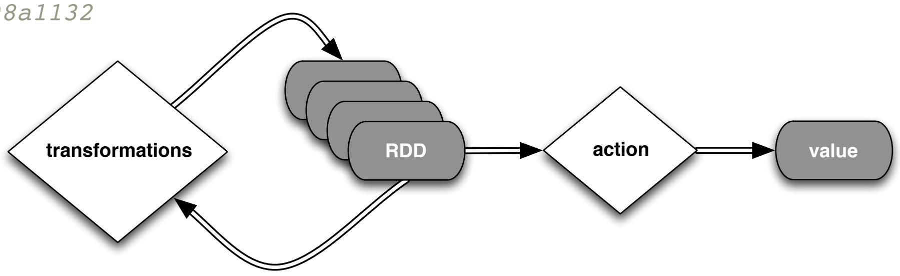
```
// action 2
messages.filter(_.contains("php")).count()
```



Spark Deconstructed:

Looking at the RDD transformations and actions from another perspective…

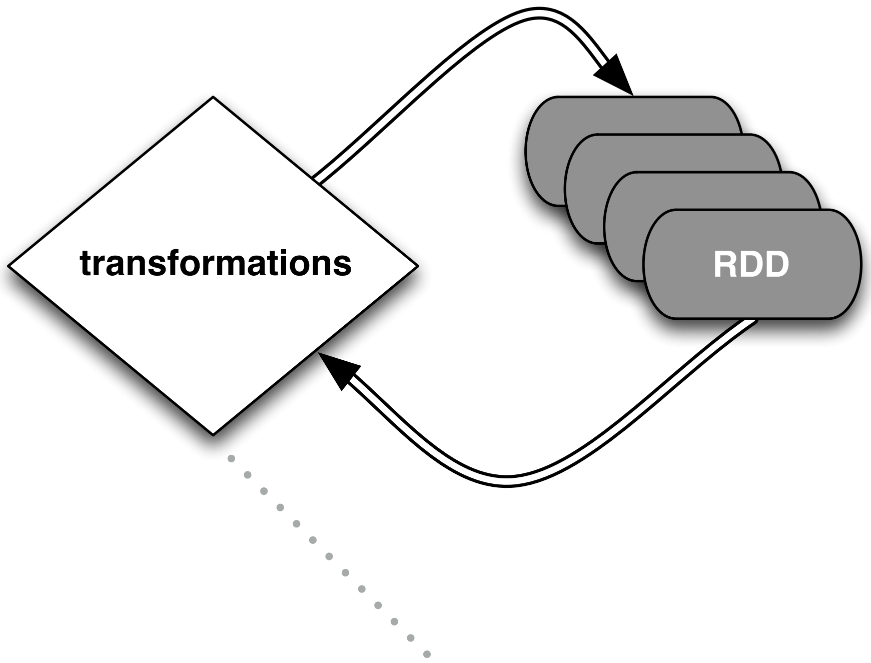
```
// load error messages from a log into memory
// then interactively search for various patterns
// https://gist.github.com/ceteri/8ae5b9509a08c08a1132

// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```



Spark Deconstructed:

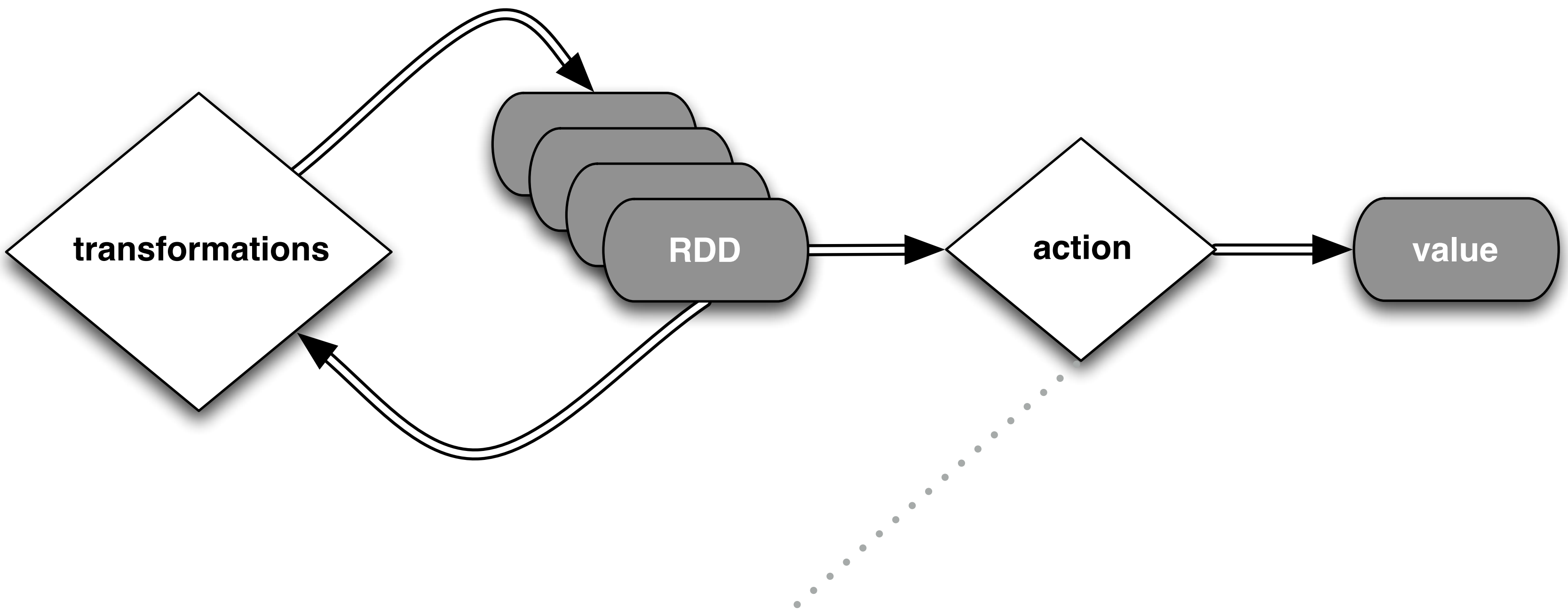
```
// base RDD
val lines = sc.textFile("hdfs://...")
```

Spark Deconstructed:



```
// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()
```

Spark Deconstructed:



```
// action 1
messages.filter(_.contains("mysql")).count()
```