Initiation à Apache Spark avec Java

Hamzaoui Mohamed Ali

2016

Faculté des sciences mathématiques, physiques et naturelles de Tunis

Plan

1. Intoduction
2. Familiarisation avec les concepts de Spark
3. Mise en oeuvre des RDDs

Intoduction

Intoduction

Apache Spark se présente comme la nouvelle génération de moteur de calcul distribué qui remplace progressivement Hadoop/MapReduce.

Spark est écrit en Scala et s'exécute sur la machine virtuelle Java (JVM).

Les langages supportés actuellement pour le développement d'applications sont : Scala , Python , Clojure R ET JAVA .

Mise en place de l'environement

Pour bien commencer, il vous faut installer sur la machine :

- Un JDK $\succcurlyeq$1.8.x.
- Un IDE ou éditeur de texte : Sublime Text, Eclipse, netbeans.
- Le binaire pré-compilé de Spark.

Sommaire

Intoduction

Familiarisation avec les concepts de Spark

Mise en oeuvre des RDDs

Spark Context

SparkContext est la couche d'abstraction qui permet à Spark de savoir où il va s'exécuter.

Un SparkContext standard sans paramètres correspond à l'exécution en local sur 1 CPU du code Spark qui va l'utiliser.

```
public class FirstRDD {
public static void main(String[] args) {
JavaSparkContext sc = new JavaSparkContext();
JavaRDD<String> lines = sc.textFile("/path/fst.
    txt");}}
```

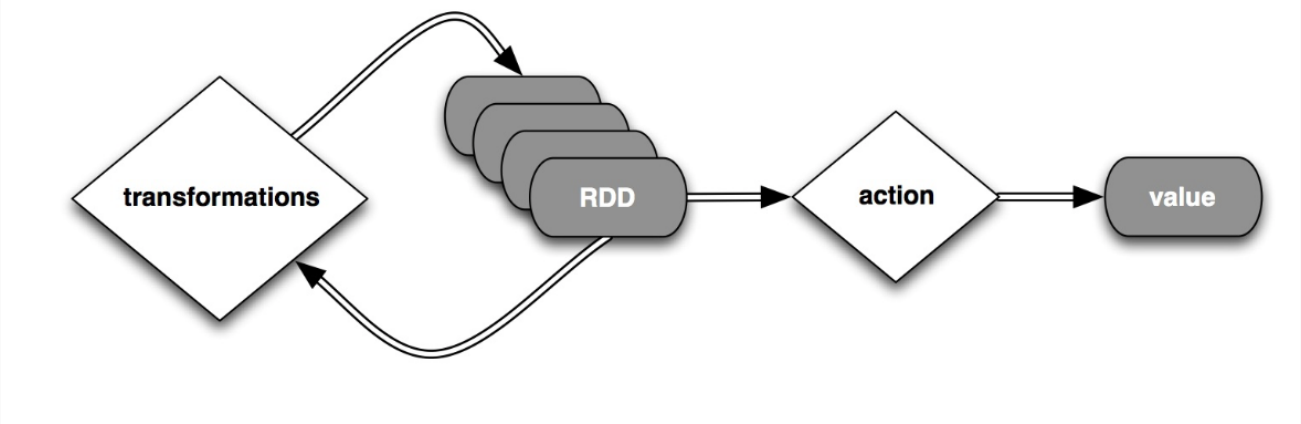
RDD

L'abstraction de base de Spark est le RDD pour Resilient Distributed Dataset, c'est une structure de donnée immutable, nous pouvons voir un RDD comme une table dans une base de données.

Un calcul distribué avec Spark commence toujours par un chargement de données via un Base RDD.

Transformations et Actions

2 concepts de base s'appuient et s'appliquent sur le RDD.



Sommaire

Intoduction

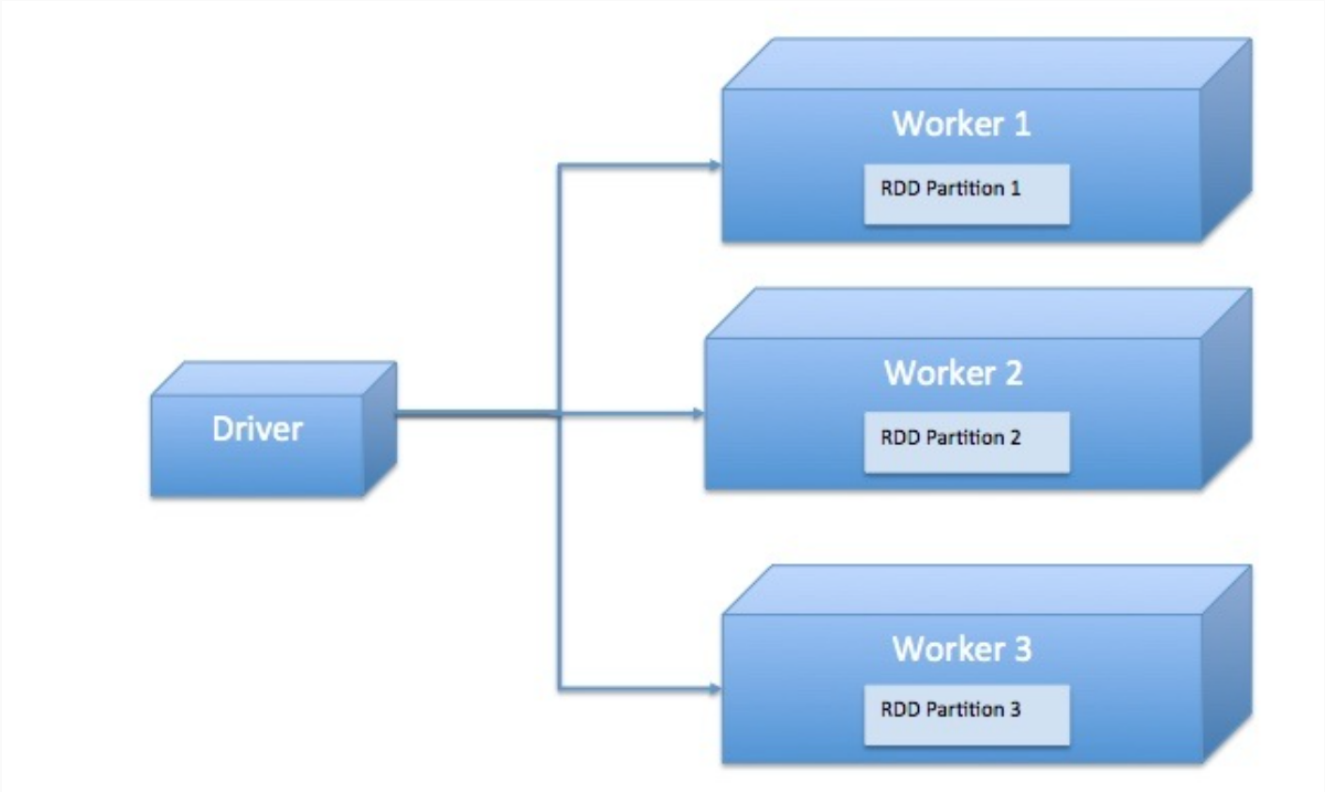
Familiarisation avec les concepts de Spark

Mise en oeuvre des RDDs

Mise en oeuvre des RDDs

Les RDDs sont une collection d'objets immuables répartis sur plusieurs noeuds d'un cluster. Un RDD est créé à partir d'un source de données ou d'une collection d'objets Scala, Python ou Java.

Mise en oeuvre des RDDs



Mise en oeuvre des RDDs

Les opérations disponibles sur un RDD sont :

La création :

```
public class FirstRDD {
public static void main(String[] args) {
JavaRDD<String> lines = sc.textFile("data.txt");
List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
JavaRDD<Integer> distData = sc.parallelize(data);}}
```

Mise en oeuvre des RDDs

Les transformations :

Les transformations ne retournent pas de valeur seule, elles retournent un nouveau RDD. Par exemple : map, filter, flatMap, groupByKey, reduceByKey.

```
public class FirstRDD {
public static void main(String[] args) {
JavaRDD<String> lines = sc.textFile("data.txt");
List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
JavaRDD<Integer> distData = sc.parallelize(data);
JavaRDD<Integer> 2distData = distData.map(( i)->{
return i*2;});}}
```

Mise en oeuvre des RDDs

L'action :

Les actions évaluent et retournent une nouvelle valeur. Au moment où une fonction d'action est appelée sur un objet RDD, toutes les requêtes de traitement des données sont calculées et le résultat est retourné. Les actions sont par exemple *reduce*, *collect*, *count*, *first*, *take*, *countByKey* et *foreach*.

```
public class FirstRDD {
public static void main(String[] args) {
JavaRDD<String> lines = sc.textFile("data.txt");
List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
JavaRDD<Integer> distData = sc.parallelize(data);
JavaRDD<Integer> = distData.reduce((j,i) -> {
return i + j;});}}
```

Les Pairs RDD

Spark offre des fonctionnalités spécifiques aux RDD clef-valeur, RDD[(K,V)] . Il s'agit notamment des fonctions *groupByKey, reduceByKey, mapValues, countByKey, cogroup.*

```
JavaRDD<String> lines = sc.textFile("data.txt");
JavaPairRDD<String, Integer> pairs;
pairs = lines.mapToPair(s -> new Tuple2(s, 1));
JavaPairRDD<String, Integer> counts;
counts = pairs.reduceByKey((a, b) -> a + b);
```