Traitements des données structurées en Spark

Master DAC – Bases de Données Large Echelle
Mohamed-Amine Baazizi
baazizi@ia.lip6.fr
2019-2020

Plan

- Récap. API Spark RDD
- Chargement de données
- Opérations de base
- Opérations complexes
  - Agrégations
  - Jointures
  - Fonctions prédéfinies
  - Fonctions utilisateur

Constat RDD

- Pas de schéma
  - code peu lisible, programmation fastidieuse

Ex. Interrogation Films (Id, Title, Genres)
accès au film identifié par 2

```
spark> val films = sc.textFile().map(_.split(‘ .. )).map(…)
spark> films.filter(x=>x._1==2)
```

1	Toy Story (1995)	Animation|Children
2	Jumanji (1995)	Adventure|Children
	..	

Possibilité 1 : RDD de Tuples

Constat RDD

- Pas de schéma
  - code peu lisible, programmation fastidieuse

Ex. Interrogation Films (Id, Title, Genres)
accès au film identifié par 2

```
spark> case Class Film(Id:Str, Title:Str, Genres:Str)
spark> val films = sc.textFile().map(_.split( … )).map(…)
spark> films.filter(x=>x.MovieID==2)
```

Film(1, Toy Story (1995), Animation|Children)
Film(2,Jumanji (1995),Adventure|Children)
…

Possibilité 2 : RDD d’objets

Constat RDD

- Pas de schéma
  - Code peu lisible, programmation fastidieuse
  - Encapsuler dans des objets reflétant la structure
    - Performances dégradées (sérialisation d'objets, GC)
  - Dans tous les case : absence d'optimisation logique
    - Analyse statique de la requête (vérification des attributs, projection sur attributs non pertinents, …)

Quelle API pour données structurées?

- API Datafame
  - Collection distribuée de tuples (row) avec un même schéma
  - Inspirée du langage R
- API Dataset
  - Collection distribuée d'objets conformes à un type T stockés de manière optimisée
  - Spécifique à Scala
  - Dataframe en est un cas spécifique
    - Dataframe = Dataset[Row]

Avantage des Datasets

- Typage statique : détecter erreurs avant exéc.
  - Ex. méthode utilisant attribut absent
- Haut niveau d'abstraction
  - Pas besoin de connaître l'organisation physique
    - Appel des attributs avec leur nom
- Optimisation logique
  - Ex. Projection sur attributs inutiles

Typage statique

	SQL	DataFrames	Datasets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time

Haut niveau d'abstraction

Films (Id, Title, Genres)

```
spark> val films = sc.textFile().map(_.split(' .. )).map(…)
spark> films.filter(x=>x._1==2)
```

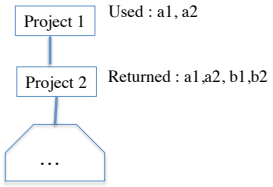
En RDD

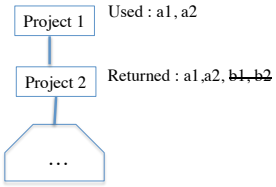
```
spark> val films = spark.load …
spark> films.where(``id=2``)
```

En Dataset

Optimisation logique

- Projection d'attributs inutiles
  - Ex. Project1 n'utilise pas b1 et b2. Les projeter dès Project2





Création de Dataset

- Par conversion de RDD
  - Méthode *toDS()* invoquée depuis une Seq
  - Importer spark.implicits._
- Par chargement de nouvelles données
  - Formats : CSV, JSON, Parquet, texte
  - Schéma utilisateur fourni
  - Schéma automatiquement inféré

Illustration

```
MovieID,Title,Genres
1,Toy Story (1995),Animation|Children…
2,Jumanji (1995),Adventure|Children….
3,Grumpier Old Men (1995),Comedy….
…
```

movies.csv

```
scala> case class Movie(MovieID:String,Title:String,Genres:String)

scala> val films = spark.read.format("csv").
            option("header",true).
            load(path+ "movies.csv").as[Movie]
films: org.apache.spark.sql.Dataset[Movie] = [MovieID: string, Title: string ... 1 more field]
```

Création de Dataset

- Cas particulier : Dataframe
  - Pas besoin de spécifier le Type (ex. classe Movie)
  - Type Row générique
  - Row = séquence de colonnes ayant un type prédéfini Spark SQL

Création d'un Dataframe : illustration

```
MovieID,Title,Genres
1,Toy Story (1995),Animation|Children…
2,Jumanji (1995),Adventure|Children….
3,Grumpier Old Men (1995),Comedy….
…
```

movies.csv

```
scala> val films = spark.read.format("csv").
            option("header",true).
            load(path+ "movies.csv").as[Movie]
films: org.apache.spark.sql.Dataset[Row]
```

Schéma non fourni : inférence automatique

Types Spark SQL

- Types de base
  - boolean, numeric (integer, decimal, …), String, null, timestamp
- Tableau
  - ArrayType(type, containsNull)
- Enregistrement
  - StructType(List [StructField])
  - StructField(name, type, nullable)
- Tableau associatif
  - MapType (keyType, valueType, valueContainsNull)

Illustration

Films (Id: num, Title: text, Genres: text)

```
StructType(List(StructField('Id', int, true),
                StructField('Title', String, true),
                StructField('Genres', String, true)
               )
          )
```

Quelques opérations Dataset

- Actions
  - count
  - describe
  - reduce
  - show
- Fonctions
  - rdd
  - dtypes
  - printSchema
- Transformations
  - distinct
  - except
  - filter
  - flatMap
  - groupByKey
  - map
  - orderBy
  - agg
  - groupBy
  - join
  - select

Dataset par l'exmple

- Actions et fonctions de base

```
scala> films.show
+-------+--------------------+--------------------+
|MovieID|               Title|              Genres|
+-------+--------------------+--------------------+
|      1|    Toy Story (1995)|Animation|Childre...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|        Comedy|Drama|
|      5|Father of the Bri...|              Comedy|
```

```
scala> films.printSchema
root
 |-- MovieID: string (nullable = true)
 |-- Title: string (nullable = true)
 |-- Genres: string (nullable = true)
```


Dataset par l'exemple

- Transformations

```
scala> films.map(x=>x.Genres.split("\\|")).show
+--------------------+
|               value|
+--------------------+
|[Animation, Child...|
|[Adventure, Child...|
|   [Comedy, Romance]|
|     [Comedy, Drama]|
+--------------------+
```

```
scala> films.orderBy("Title").show
+-------+--------------------+--------------------+
|MovieID|               Title|              Genres|
+-------+--------------------+--------------------+
|      5|Father of the Bri...|              Comedy|
|     10|    GoldenEye (1995)|Action|Adventure|...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      6|         Heat (1995)|Action|Crime|Thri...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      7|      Sabrina (1995)|      Comedy|Romance|
|      9| Sudden Death (1995)|              Action|
```


Dataset par l'exemple

- Agrégation simple – agrégation avec groupement

```
scala>  case class Note(UserID: Int, MovieID: Int, Rating: Int, Timestamp: Int)

scala> val notes = sc. read.format("csv").load(…)

scala> notes.agg(min("Rating"), max("Rating"), avg("Rating"))

scala> notes.describe("Rating").show //montre count, stddev en plus

scala> notes.groupBy("MovieID").agg(count("*")).sort("count(1)").show
```

Dataset par l'exmple

Sélection par critère

```
scala> films.where("MovieID=1")

scala> films.where("MovieID=1 or Title='Toy Story (1995)'")
```

Projection

```
scala> films.select("MovieID","Genres")

scala> films("MovieID") // une colonne à la fois
```

Equi-jointure

```
scala> films.join(notes,"MovieID")
```


Demo