

pythonモジュールLT会 【pyspark】

データ戦略G 橋口友哉

- Sparkについて
- pysparkの基本操作
- サンプルコード使った簡単な説明
- Spark関連の技術ブログなどの紹介

Sparkって知ってますか？

- ・**分散処理**を用いて**大容量のデータを分析**するツール
→似たツールのHadoopに比べ、最大100倍以上の処理パフォーマンスがある(らしい)
- ・SparkはJava、Python、R、Scala用のAPIが提供されている
→**Python APIのpyspark**を紹介するが、どれも似た記法で実装可能(っぽい)
- ・やりたいデータ分析内容に応じて、ライブラリを変更することができる
→業務上は**SparkSQL**のみ利用(他のライブラリは環境に入っていない?)



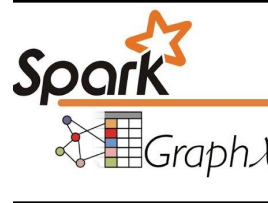
SQLでデータ操作



ストリーミング
データ処理



機械学習の前処理・モデル学習

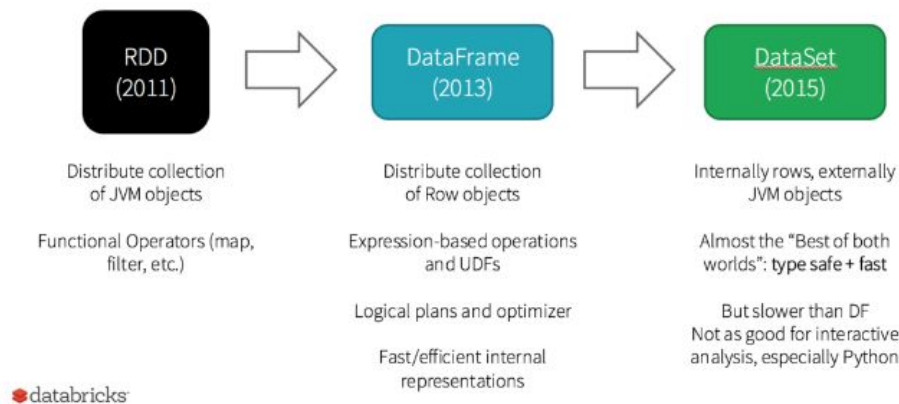


グラフ作りとグラフ並列
計算処理

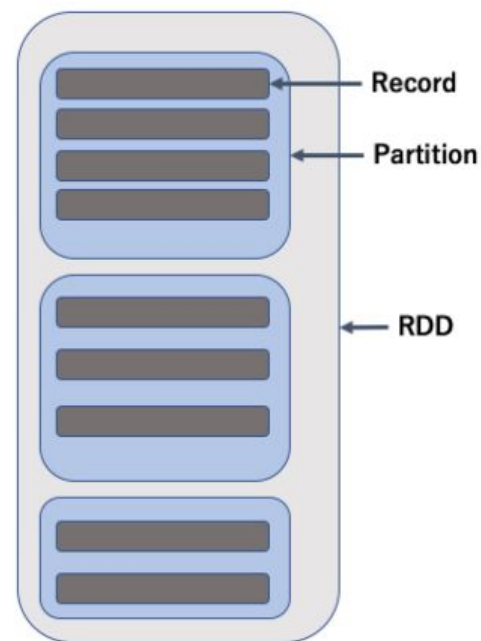
- RDD (Resilient Distributed Datasets) を基本に今のAPIができています
- 1つ1つのデータはレコード
- 同一のExecutor内のレコード集合はパーティション
- パーティション全体でRDD (DataFrame)

Spark APIの歴史

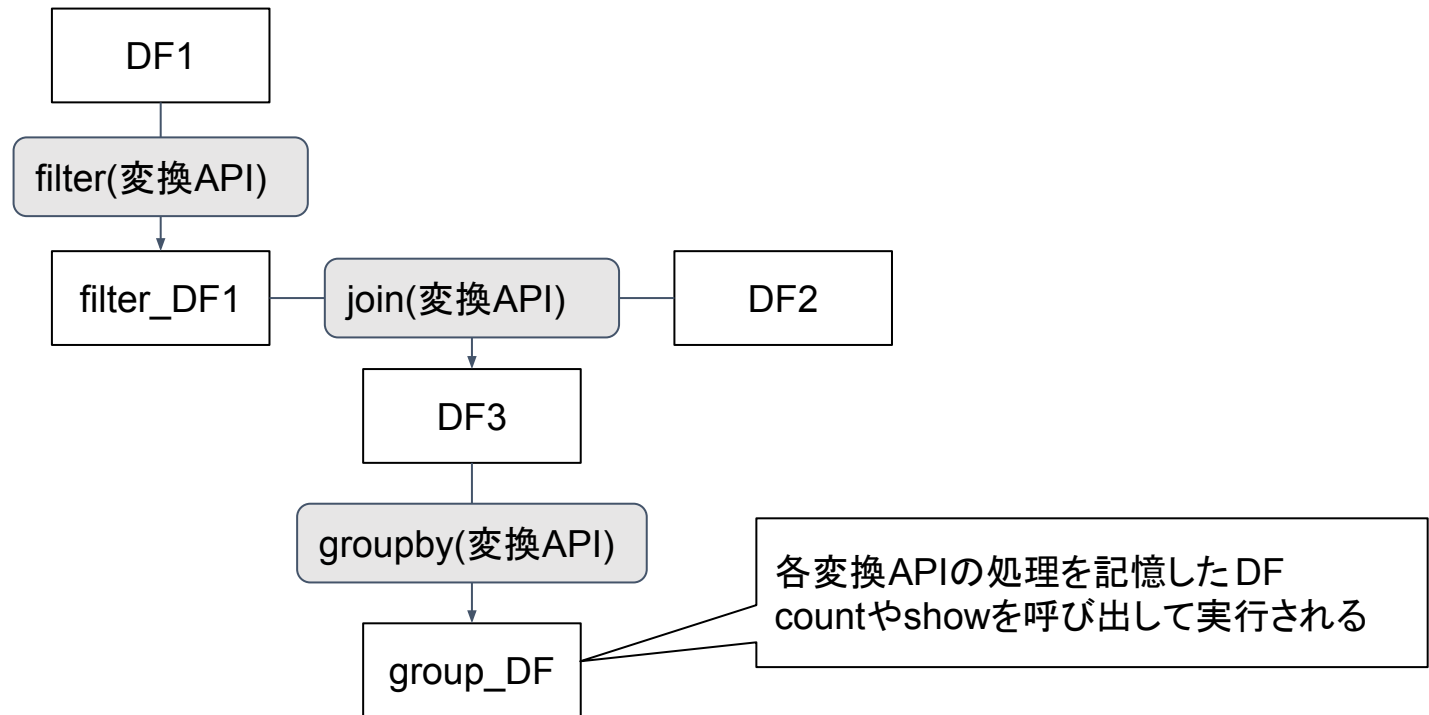
History of Spark APIs



データ構造と各種名称



filterやgroupbyなどの変換API(Transformations)では処理が実行されない。
表示などの実行API(Actions)を呼び出したときに初めて処理が実行される。



・変換API、実行API

<https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations>

<https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions>

座学はよく分からん...



実際に動かそう！



手元で実行(サンプルコード)

9

- ・Google Colabを使った処理内容を紹介
→実行環境をPC環境に依存しないでノートブックのみで構築可能

- ・データサイエンス100本ノック(構造化データ加工編)を実装

https://github.com/t-hashiguchi1995/100knock_pyspark/blob/main/preprocess_knock_Python_Spark.ipynb

main 100knock_pyspark / preprocess_knock_Python_Spark.ipynb Go to file ...

t-hashiguchi1995 最終版 Latest commit 00dadf7 3 days ago History

1 contributor

6083 lines (6083 sloc) 285 KB

業務PC以外でクリック
結果を表示してるので結果のみ確認も可

Open in Colab

データサイエンス100本ノック (構造化データ加工編) - Python
for Google Colab

※業務に一部利用できないコードあるかも？(pyspark.pandasは利用不可)

手元で実行(サンプルコード)

10

The screenshot shows the Google Colab interface. At the top, the file name is 'preprocess_knock_Python_Spark.ipynb'. Below it are tabs for 'ファイル', '編集', '表示', '挿入', 'ランタイム', 'ツール', and 'ヘルプ'. The 'ランタイム' (Runtime) menu is open, displaying various options with their corresponding keyboard shortcuts. The first option, 'すべてのセルを実行' (Run all cells), is highlighted. Below it are 'より前のセルを実行' (Run previous cells), '現在のセルを実行' (Run current cell), '選択範囲を実行' (Run selection), and '以降のセルを実行' (Run remaining cells). Further down are '実行を中断' (Interrupt), 'ランタイムを再起動' (Restart), '再起動してすべてのセルを実行' (Restart and run all cells), 'ランタイムを接続解除して削除' (Disconnect and delete), 'ランタイムのタイプを変更' (Change runtime type), 'セッションの管理' (Manage session), and 'ランタイムログの表示' (Show runtime logs). In the background, a code cell is visible with the text 'データサイエンス for Google Colab' and a code block starting with '!git clone https://git'. Below the code block, the output of the command is shown: 'Cloning into '100knock...', 'remote: Enumerating objects 100%', 'remote: Counting objects 100%', 'remote: Compressing objects 100%', 'remote: Total 1448 (delta 1000)', 'Receiving objects: 100%'.

preprocess_knock_Python_Spark.ipynb
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

+ コード + テキスト

データサイエンス
for Google Colab

処理対象のデータ

```
[ ] !git clone https://git
```

Cloning into '100knock':
remote: Enumerating objects 100%
remote: Counting objects 100%
remote: Compressing objects 100%
remote: Total 1448 (delta 1000)
Receiving objects: 100%

ランタイム > すべてのセルを実行

で動くはず...

P-086でたまにエラーがでるが、
ランタイム > 再起動してすべてのセルを実行
で動く

※業務に一部利用できないコードあるかも？(pyspark.pandasは利用不可)

- ・読み込み/書き込み: `spark.read()`、`df.write()`
- ・表示: `df.show`、`df.head`、`df.tail...`
- ・カラム追加※¹: `df.withColumn(col, val)`
- ・結合: `df1.join(df2, col, 結合方法)`
- ・集約: `df.groupBy(col).agg(functions)`
- ・ユニーク化※²: `df.drop_duplicates()`、`df.distinct()`
- ・条件抽出: `df.filter(条件)`、`df.where(条件)`
- ・ソート(昇順、降順)※³: `df.orderBy(col.asc)`、`df.orderBy(col.desc)`
- ・値格納※⁴: `df.cache()`、`df.persist()`
- ・パーティション操作※⁵: `df.coalesce(N)`、`df.repartition(N)`
- ・ウィンドウ: `Window.partitionBy(col).orderBy(col)`
- ・カラムの値を分解※⁶: `explode(col)`、`split(col, 条件).getItem(N)`
- ・正規表現: `like("**")`、`rlike("**")`
- ・自作関数: `udf`

※¹ `val`には`case`文(`when`)を使って、条件ごとに値を入れることもできる

※² 特定の条件下(最新日付など)でユニーク化したい場合は挙動に要注意

※³ パフォーマンス上あまり使わないほうがいい

※⁴ 実行APIまでの遅延評価を1度だけにする

※⁵ `N=1`は動作が遅くなるので、書き込みのタイミング以外では使わない

※⁶ `explode`: カラムのリストを縦に分解、`split`: カラムの文字列を条件で区切る、`getItem`: リストの`N`番目取得

- ・Sparkは分散処理をしてくれるフレームワーク
- ・pysparkはSparkを呼び出すAPIでいろいろライブラリがある
- ・データ処理はpythonのインタプリタ方式とは異なる
- ・変換APIと実行APIがあり、実行APIを動かして、初めて処理される
 - まとめて変換APIを書くとデバッグが...
 - 変換結果を確認するために都度実行すると処理が...
- ・pysparkのコードについて業務で扱うものは大体網羅できているはず
 - コードの不明点については橋口に聞いてもらえれば、ある程度は回答可能
 - いただいた不明点はコメントつけてアップデートしていきたい

あー
そーゆーことね
完全に
理解した

おかてない →



ARISEが公開している技術ブログより

- ・Scala sparkのコードを確認→大体理解できるようになってる....はず！！

<https://www.ariseanalytics.com/activities/report/20201111/>

- ・Sparkのパラメータチューニング(EMRのチケット発行時のパラメータ)

<https://www.ariseanalytics.com/activities/report/20201030-2/>

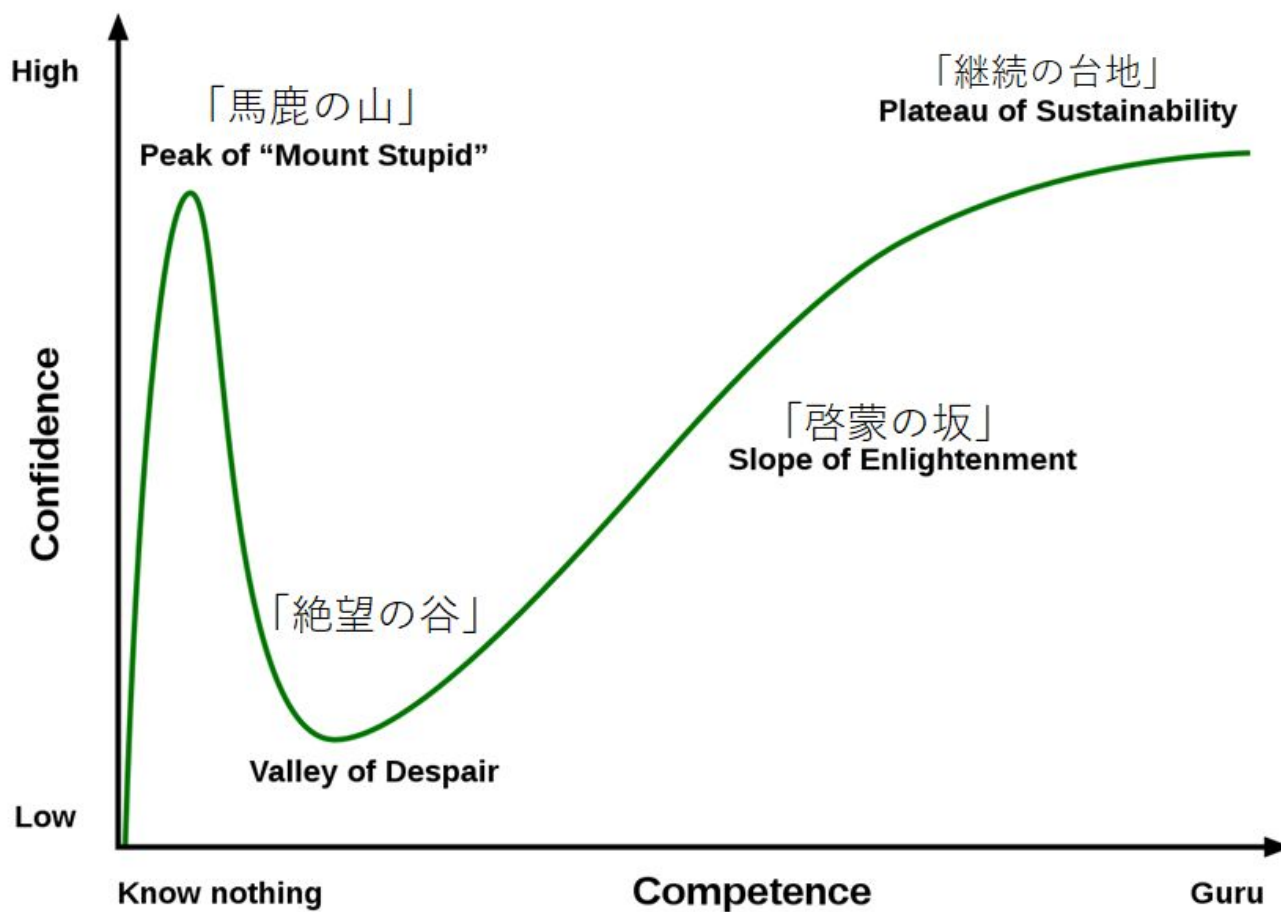
パラメータチューニングは結構重要らしい(正直よーわからん)

- ・<https://qiita.com/uryyyyyyyy/items/ba2dceb709f8701715f7>

- ・https://qiita.com/taka_yayoi/items/8d964a4949f41e2bf6bd

『完全に理解した』曲線は本当だった！！

Dunning-Kruger Effect



AI Programmer: 日本語⇔コードの相互変換をしてくれる補助ツール

- ↳ Github Copilotの日本語版みたいな使い方ができる(β版で無料)
- ↳ pysparkで〇〇を処理したい! の大枠は書いてくれる(微修正必要)

Python

pysparkでレシート明細データフレーム(df_receipt)に対し、店舗コード(store_cd)ごとに売上金額(amount)と売上数量(quantity)を合計せよ。

```
df_receipt.groupby(store_cd).agg(sum(amount), sum(quantity))
```



日本語→コード

コードを日本語で解説

```
df_receipt.groupby("store_cd").sum("amount", "quantity")\
.withColumnRenamed("sum(amount)", "amount")\
.withColumnRenamed("sum(quantity)", "quantity")\
.show()
```

このコードは、「店舗コード」をグループ化し、「金額」と「数量」の合計を計算しています。
また、「sum(amount)」と「sum(quantity)」の列名を「amount」と「quantity」に変更しています。
最後に、計算結果を表示しています。



コード→日本語

Python

SQL

Python

JavaScript

HTML & CSS

Solidity

C++

PHP

Go

R

Perl

Ruby

正規表現

Haskell

Lisp

C#

コードを日本語で解説

Lisp

日本語→コード
の対応言語

・AI Programmer

<https://aiprogrammer.hashlab.jp/>

- ・公式ドキュメント

<https://spark.apache.org/docs/3.2.0/api/python/reference/index.html>

<https://spark.apache.org/docs/3.2.0/api/python/reference/pyspark.sql.html>

- ・変換API、実行API

<https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations>

<https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions>

- ・Sparkで参考になりそうな記事たち

<https://qiita.com/miyamotok0105/items/bf3638607ef6cb95f01b>

<https://qiita.com/uryyyyyyyy/items/ba2dceb709f8701715f7>

https://toeming.hatenablog.com/entry/2021/04/04/Spark_LazyEval_and_Persist

<https://www.databricks.com/jp/glossary/spark-api>

- ・サンプルコード

<https://qiita.com/tchih11/items/90f4b87de65464fe2881>

<https://qiita.com/taka4sato/items/4ab2cf9e941599f1c0ca>

- ・Colabでpyspark

https://techblog.gmo-ap.jp/2021/06/07/colab_pyspark/

- ・データサイエンス 100本ノック(構造化データ加工編)

<https://github.com/The-Japan-DataScientist-Society/100knocks-preprocess>

- ・AI Programmer

<https://aiprogrammer.hashlab.jp/>