Многоликая Pandas

Павел Филонов

Идея доклада

Павел Филонов
Data Scientist
t.me/pavel_filonov

- разнообразие DS проектов
- воспроизводимость
- быстрая проверка гипотез
- переиспользование кода

This is the way

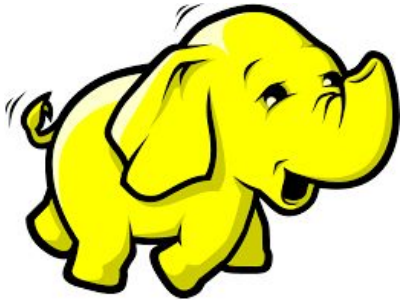
Small Data

EDA

Enough Data

fit_predict

Big Data



batch

Наши милые API

Pandas

Dask

Spark

AMPLab Big Data benchmark

Rankins

```
pageURL       VARCHAR(300)
pageRank      INT
avgDuration   INT
```

UserVisits

```
sourceIP      VARCHAR(116)
destURL       VARCHAR(100)
visitDate     DATE
adRevenue     FLOAT
userAgent     VARCHAR(256)
countryCode   CHAR(3)
languageCode  CHAR(6)
searchWord    VARCHAR(32)
duration      INT
```

Documents

Unstructured HTML documents

Size	Rankins (rows)	Rankings (bytes)	UserVisits (rows)	UserVisits (bytes)	Documents (bytes)
tiny	1200	77.6 KB	10000	1.7 MB	6.8MB
1node	18 Million	1.28 Gb	155 Million	25.4GB	29.0 GB
5node	90 Million	6.38 Gb	775 Million	126.8 GB	136.9 GB

Round 1

Pandas

Tiny

Simple queries

```
rankings
     .loc[rankings.pageRank > 100]
```

Win

```
uservisits
   .groupby(uservisits.sourceIP.str.slice(0, 7))
   .adRevenue
   .sum()
```

Win

UDF

```
url_regex = re.compile("(?P<url>https?://[^\s]+)")


crawl
    .pageSource
    .apply(extract_url)
    .value_counts()
```

Win

Join

```
rankings
    .merge(uservisits, left_on="pageURL", right_on="destinationURL", how="left")
    .query("visitDate > '1980-01-01' and visitDate < '1980-04-01'")
    .groupby("sourceIP")
    .agg({
            "pageRank": "mean",
            "adRevenue": "sum",
    })
    .sort_values("adRevenue", ascending=False)
    .head(10)
```

Flawless Victory

Round 2

Pandas

1 Node

Simple queries

```
rankings
     .loc[rankings.pageRank > 100]
```

Win

```
uservisits
   .groupby(uservisits.sourceIP.str.slice(0, 7))
   .adRevenue
   .sum()
```

Win

UDF

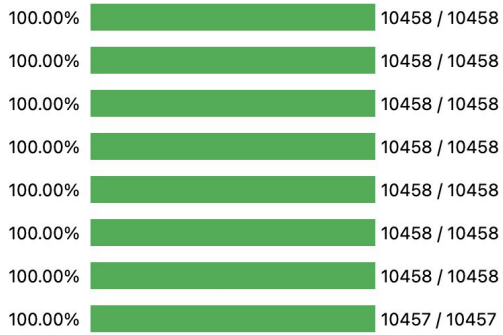
```
url_regex = re.compile("(?P<url>https?://[^\s]+)")


crawl
    .pageSource
    .apply(extract_url)
    .value_counts()
```

Too Long

Pandarallel

```
pandarallel.initialize(progress_bar=True)
```

```
crawl
    .pageSource
    .parallel_apply(extract_url)
    .value_counts()
```



Win

Join

```
rankings
    .merge(uservisits, left_on="pageURL", right_on="destinationURL", how="left")
    .query("visitDate > '1980-01-01' and visitDate < '1981-01-01'")
    .groupby("sourceIP")
    .agg({
            "pageRank": "mean",
            "adRevenue": "sum",
    })
    .sort_values("adRevenue", ascending=False)
    .head(10)
```

Pandas loses

Разбор полетов

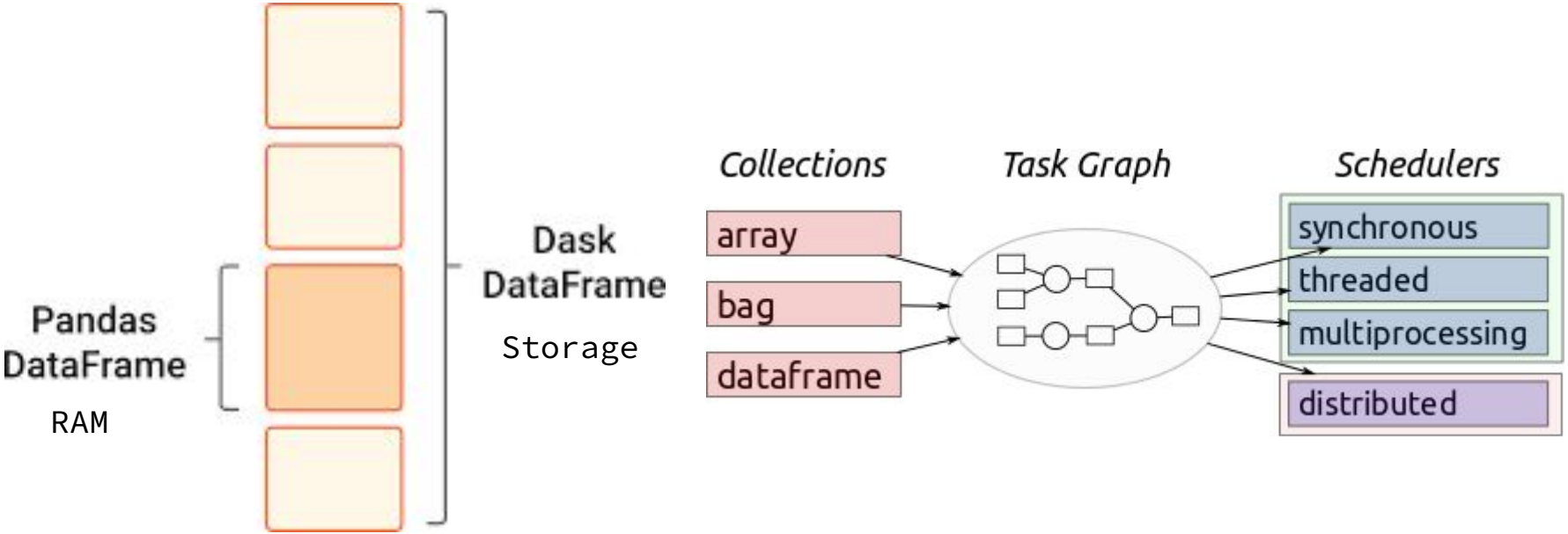
- идеально для маленьких объемов
- можно ускорить с помощью pandarallel
- нужно много ~~бамбука~~ RAM

Round 3

Dask

1 Node

Dask


Pandas
DataFrame
RAM
Dask
DataFrame
Storage
Collections
array
bag
dataframe
Task Graph
Schedulers
synchronous
threaded
multiprocessing
distributed

Simple queries

```
rankings
     .loc[rankings.pageRank > 100]


uservisits
   .groupby(uservisits.sourceIP.str.slice(0, 7))
   .adRevenue
   .sum()
```

UDF

```
url_regex = re.compile("(?P<url>https?://[^\s]+)")


crawl
    .pageSource
    .apply(extract_url)
    .value_counts()
```

Join

```
rankings
    .merge(uservisits, left_on="pageURL", right_on="destinationURL", how="left")
    .query("visitDate > '1980-01-01' and visitDate < '1980-04-01'")
    .groupby("sourceIP")
    .agg({
            "pageRank": "mean",
            "adRevenue": "sum",
    })
    .sort_values("adRevenue", ascending=False)
    .head(10)
```

Dask Wins

Round 4

Dask

5Node

Хорошее время для вопросов

Round 5

Spark

5Node

Simple queries

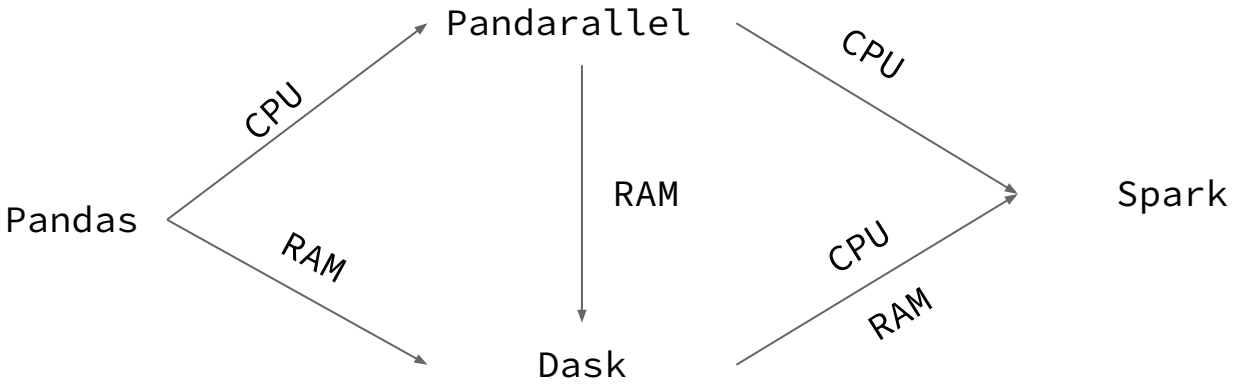
```
rankings
    .filter(rankings["pageRank"] > 100)
```

```
uservisits
    .withColumn("sourceIPpart", substring("sourceIP", 1, 8))
    .groupBy("sourceIPpart")
    .sum("adRevenue")
```

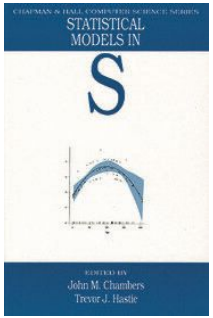
Pandas API on Spark

```
import pyspark.pandas as ps

rankings
     .loc[rankings.pageRank > 100]

uservisits
   .groupby(uservisits.sourceIP.str.slice(0, 7))
   .adRevenue
   .sum()
```

Как же выбрать?



К истокам DataFrame API



1990

2000

2009

2015

2021
(Pandas API)

Источники

- AMPLabs Big Data Benchmark
- Pandarallel
- Dask
- Pandas API on Spark
- Preventing the Death of Dataframes
- Polars
- examples source codes


So long, and thanks
for all the bamboo!