ダウンロード・データあります

第4章 大規模データの処理では効果絶大

200万件のデータで並列処理の有効性を試す

佐藤 聖

```
1  df.head(3)
```

	Regio	old	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit
0	Australia and Ocea	4300	421.89	364.69	1814127.00	1568167.00	245960.00
1	As	145	81.73	56.67	338770.85	234897.15	103873.70
2	Sub-Saharan Africa	07	437.20	263.33	2801140.40	1687155.31	1113985.09

```
1  def func(x):
```

図1 大規模なデータに対して並列処理の有効性を確かめる
取得したデータの一部

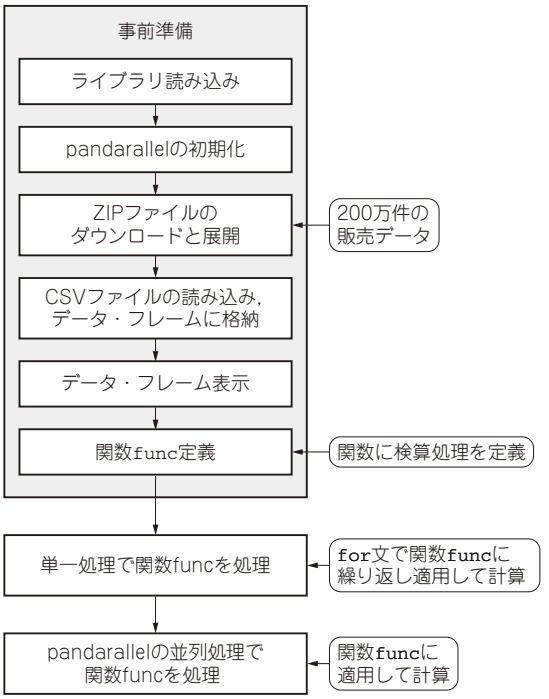

図2 大量のデータに対して単一処理と並列処理で速度を比べる

● 第3章までで本当に速いの？と思った方へ

一般的に並列処理が効果的なのは，データが大量であったり，データに対する計算や操作が複雑であったりする場合です．そのため並列化する前にシングル・スレッドで逐次処理したときに時間がかかりすぎるプログラムか否かを見定める必要があります．

第2章～第3章のプログラム例では，複雑な計算を並列処理していなかったため，Pythonの並列処理が有効なのかどうか確証が持てなかったかもしれません．

```
!pip install pandarallel

import pandas as pd
from pandarallel import pandarallel
pandarallel.initialize()  ←①

# Downloads 18 - Sample CSV Files / Data Sets for Testing(
!wget http://eforexcel.com/wp/wp-content/uploads/
!unzip 2m-Sales-Records.zip  2020/09/2m-Sales-Records.zip
                                                    ②

filename = '2m Sales Records.csv'  ←③
df = pd.read_csv(filename, header=0)
```

図3 pandarallelをインストールおよび初期化する

● より現実に近い200万件の販売データで実験

実践的なデータで並列処理を試すため，E for Excelサイト(1)から販売データ(200万件)を入手しました．ただし，データはVBAのランダム・ロジックで生成された実在しないダミーです．

これ以外にも500万件までのデータセットがあり，プログラムのテスト用データとして配布されています．

以降のプログラムでは，入手した販売データを使って実験します(**図1**)．

単一処理と並列処理の比較実験

通常のPythonプログラムによる単一処理と，pandarallelによる並列処理とを大量のデータ処理を通して比較します．処理の流れを**図2**に示します．

データ処理は関数funcに定義しておきます．このように既に作成された関数を，単一処理と並列処理とで比べます．