Porovnanie exekučnej efektívnosti programovacích jazykov Python a Scala používaných v aplikáciách pre dátovú vedu

Comparison of the code execution efficiency of programming languages Python and Scala used in data science applications

Pavol Sojka[1]

Abstrakt
Náš článok sa zameriava na exekučnú efektívnosť programovacích jazykov Python a Scala z hľadiska časovej náročnosti. Oba jazyky sa používajú v dátovej vede. Dátová veda sa v súčasnosti rozrástla vo veľkom meradle a tento článok by mal priniesť lepší prehľad o tom, ktorý z vybraných jazykov má lepší výkon. Pripravili sme experiment a scenár bežne používaný v praxi v oboch jazykoch, ktorý zahŕňa prípravu dát, ich spracovanie a výpočet a následne interpretáciu vrátených výsledkov.

Kľúčové slová
Python, Scala, množiny údajov, spracovanie údajov

Abstract
Our paper aims on using programming languages like Python and Scala to compare their efficiency in the meaning of the code execution times. Both languages are being used in data science. Data science has grown at large scale nowadays and this paper should bring a better insight into which of the chosen languages have better performance. We prepared an experiment and scenario commonly used in practice in both languages which comprises data preparation, data processing and calculation and consequently interpretation of returned results.

Key words
Python, Scala, datasets, data processing

JEL classification
C8

1 Úvod

V súčasnosti je veľký dopyt po dátových analytikoch a programátoroch, pretože dátová veda a priemysel v posledných rokoch výrazne narástli. S týmto trendom sa objavujú otázky, ako napríklad, ktorý programovací jazyk by sme mali použiť na splnenie požiadaviek na spracovanie dát. Mnohé odpovede možno nájsť na internete alebo v literatúre. Môžeme zvážiť tieto odporúčania na základe rozšírenosti jazyka, krivky učenia, dostupnosti zdrojov poznatkov a podobne. V našom článku sme si vybrali jedno kritérium na porovnanie a toto kritérium je založené striktne na meraní efektívnosti na základe hardvérových parametrov s prednastaveným softvérovým scenárom.

[1] Ing. Pavol Sojka, PhD., Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1/b, 852 35 Bratislava 5, Slovakia, pavol.sojka@euba.sk

2 Príprava prostredia

Ako primárny operačný systém, na ktorom budú vykonávané experimenty, sme si vybrali Linux Ubuntu 22.04. Naše prostredie bolo umiestnené na platforme Google Cloud na virtuálnom stroji s hardvérovými parametrami začínajúcimi na ôsmich CPU (centrálna procesorová jednotka) a šestnástich gigabajtoch RAM (operačná pamäť). Počet CPU sa postupne menil a merali sme dopad na efektívnosť vykonávania kódu z hľadiska času. Môže sa zdať, že výsledky sú ľahko predvídateľné, ale naším cieľom nebolo len dokázať, že zníženie počtu CPU má negatívny vplyv na výkon, ale tiež presne merať hodnoty vo zvolenom prostredí a skúmať, či tento vplyv je rovnaký alebo podobný u oboch programovacích jazykov.

Po úspešnom nasadení operačného systému sme nainštalovali jazyk Python a jazyk Scala z repozitárov operačného systému. Spolu s Pythonom sme nainštalovali aj knižnicu Pandas. Táto knižnica je navrhnutá na efektívnu prácu s dátovými sadami uloženými vo formátoch CSV, XLSX a ďalších. Na úspešnú inštaláciu kompilátora Scala je potrebné nainštalovať Java framework a vývojové nástroje.

Naším cieľom bolo otestovať rýchlosť nášho naprogramovaného kódu v týchto krokoch:

- Otvoriť súbor CSV
- Načítať stĺpce dátovej sady do pamäte
- Vypočítať súčet celého stĺpca „Tržby“ a „Zisk“
- Vytlačiť názov súboru a výsledok na obrazovku
- Vytlačiť výsledný čas vykonávania a ukončiť aplikáciu

Rozhodli sme sa generovať syntetické dátové sady vo formáte CSV (hodnoty oddelené čiarkami). Na tento účel sme použili aplikáciu naprogramovanú v jazyku Java pôvodne pre iné experimenty vykonané v minulosti. Vygenerovali sme 500 testovacích súborov s údajmi o 20 riadkoch a 15 stĺpcoch. Náš program môže generovať dátové sady so stovkami tisíc riadkov, ale pre naše experimenty by malo byť 500 súborov x 20 riadkov dostatočných. Vygenerované súbory boli skopírované do adresárov Python a Scala.

3 Výsledky

Každé meranie sme vykonali trikrát za sebou v oboch jazykoch a zaznamenané časy sme uložili do tabuľky. Po troch cykloch sme vypli server a postupne menili počet používaných CPU z ôsmich na dve. Pozorovali sme malé zmeny v časoch vykonávania kódu, ale vyskytli sa len menšie rozdiely, až pri dvoch CPU bola zmena času významná. Ako vidno z tabuliek nižšie, môžeme povedať, že program v Scale bol efektívnejší ako v jazyku Python, pretože Scala je kompilovaná do bajtkódu spusteného na Java Virtual Machine, ktorý je vždy rýchlejší ako interpretovaný jazyk – Python. Oba programy bežali v konzolovom okne Linux bash. Prvý meraný cyklus v oboch jazykoch bol vždy pomalší ako ďalšie dva. Tento efekt možno vysvetliť prednačítaním systémových knižníc do pamäti po prvom behu.

Tab. 1: Časy vykonávania (8 CPU)

	CPU: 8, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel (python/scala)**
1. meranie	1,50	0,51	2,92
2. meranie	1,08	0,22	4,84
3. meranie	1,08	0,24	4,45
priemer	**1,22**	**0,33**	3,74

Zdroj: vlastné spracovanie

Tab. 2: Časy vykonávania (6 CPU)

	CPU: 6, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel**
1. meranie	1,44	0,56	2,55
2. meranie	1,07	0,25	4,22
3. meranie	1,09	0,26	4,14
priemer	**1,20**	**0,36**	3,33

Zdroj: vlastné spracovanie

Tab. 3: Časy vykonávania (4 CPU)

	CPU: 4, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel**
1. meranie	1,45	0,60	2,42
2. meranie	1,07	0,31	3,49
3. meranie	1,08	0,32	3,37
priemer	**1,20**	**0,41**	2,94

Zdroj: vlastné spracovanie

Tab. 4: Časy vykonávania (2 CPU)

	CPU: 2, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel**
1. meranie	1,76	0,84	2,08
2. meranie	1,22	0,52	2,35
3. meranie	1,25	0,49	2,57
priemer	**1,41**	**0,62**	2,28

Zdroj: vlastné spracovanie

Ako môžeme vidieť z údajov v tabuľkách, jazyk Scala je vždy efektívnejší ako jazyk Python. Efektívnosť oboch jazykov mierne klesá vždy, keď sa zníži počet jadier (CPU). Tabuľky 1–3 ukazujú, že efektívnosť Pythonu je podobná a rýchlo klesá v poslednom scenári iba s dvoma CPU. Efektívnosť jazyka Scala významne klesá v každom experimente. Ako bolo spomenuté vyššie, prvé meranie je pomalšie ako ďalšie dve a tento efekt ovplyvňujú interné procesy riadené operačným systémom. Rozdiely v jazykoch Python a Scala sú na prvý pohľad zrejmé. V porovnaní s Pythonom a knižnicou Pandas bude Scala vždy efektívnejšia, pretože Scala je staticky typovaný programovací jazyk a kompilátor pozná každú premennú a výraz pri behu (Jancauskas, 2016). Na druhej strane Python je dynamicky typovaný programovací jazyk a preto sa typ premennej odvodzuje na základe jej použitia (Kohli, 2021). Ďalší rozdiel je, že Scala je kompilovaná do bajtkódu, ktorý sa následne vykonáva na Java Virtual Machine (Jancauskas, 2016). Python s jeho najbežnejším použitím je interpretovaný vždy zo zdrojového kódu, čo je jasne viditeľné z časov vykonávania (nižší čas spracovania = vyššia efektívnosť aplikácie pri ostatných nezmenených podmienkach). Na optimalizáciu procesov nášho konceptu by sme mali zahrnúť špecifické algoritmy vhodné pre daný typ úlohy, hlbší vhľad do konkrétneho programovacieho jazyka, výber vhodnejších programovacích jazykov pre strojové učenie alebo výber výkonnejších knižníc jazyka Python použitých pri strojovom učení (Esposito & Esposito, 2020). Náš článok má za cieľ základné predstavenie často používaných

jazykov strojového učenia, ktoré zahŕňajú jazyky ako Scala, Python a R (Farth, 2018). Často používaným nástrojom v praxi je tiež rámec Spark (Mehrotra & Grade, 2019), ktorý využíva efektívnosť Scala/Java a knižnice Spark používaných na manipuláciu s obrovským množstvom dát, ako aj platformy Hadoop (Nordeen, 2020) používané na udržiavanie takýchto veľkých dátových sád. Nasledujú časti kódu, ktoré sme použili v našom testovacom prostredí.

Obr. 1: Ukážka kódu v jazyku Python

```
for f in csv_files:
        df = pd.read_csv
 (f, sep=';')
        print('Location:', f)
        print('File Name:', f.split("\\")[-1])
                print(f)
        print('Total profit:', df['profit'].sum())
        print('Total sales:', df['sale'].sum())
        print("------------------------------")
```

Zdroj: vlastné spracovanie

Obr. 2: Ukážka kódu v jazyku Scala

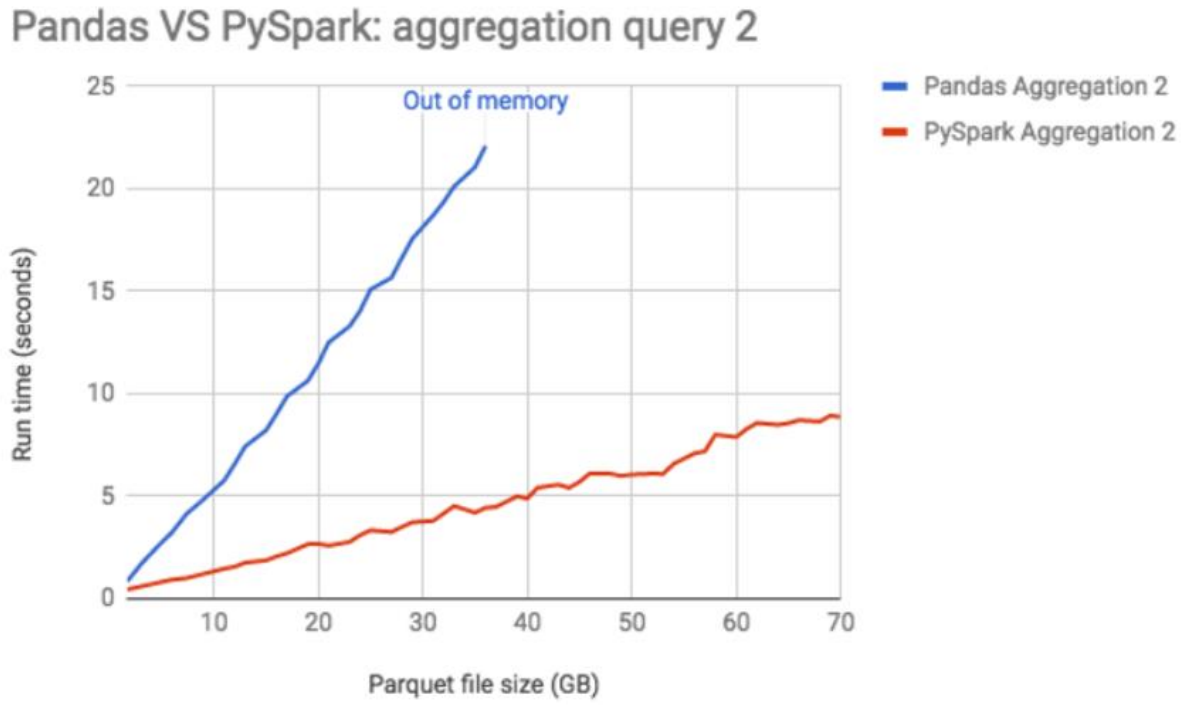
```
def compute(filename:String) {
   val bufferedSource = io.Source.fromFile(filename)
   var start = 1
   var value, value2 : Double = 0.0
   var sum : Double = 0
   var sum2 : Double = 0
   for (line <- bufferedSource.getLines) {
      val cols = line.split(";").map(_.trim)
        if (start!=1) { value = cols(4).toDouble
                        sum += value
                        value2 = cols(5).toDouble
                        sum2 += value2
                         }
        start = 0
   }
   bufferedSource.close
   println(filename)
   println("Total profit:"+sum)
   println("Total sale:"+sum2)
   println("----------------------")
 }
```

Zdroj: vlastné spracovanie

4 Diskusia

Náš prístup by mohol byť optimalizovaný pre lepšiu efektívnosť mnohými nástrojmi a knižnicami ako Hadoop, Spark a špecializovanými rámcami používanými oboma jazykmi. Ďalšie otázky, aký jazyk použiť, môžu vzniknúť, pretože v posledných rokoch dochádza k veľkému rozvoju mnohých ďalších jazykov (napr. Mojo, Julia a iné). Python môže byť tiež používaný spolu s rámcom (framework) Apache Spark. V tomto rámci je Python optimalizovaný pre špecializované knižnice, ktoré sú oveľa efektívnejšie ako samotný Python (Aven, 2018). Ako ukazuje graf nižšie, doba behu niekoľkých dopytov v PySpark (jazyk Python bežiaci spolu s Apache Spark) v porovnaní s rovnakými dopytmi v knižnici Pandas. Je zrejmé, že (Py)Spark naozaj umožňuje pracovať s väčšími dátami efektívnejším spôsobom (Element 61, 2020). Graf ukazuje (tak ako aj ako náš článok) celkovú efektívnosť funkcie „sum“ v knižnici Python Pandas oproti PySpark.

Obr. 3: Agregačný dopyt Pandas vs PySpark



Zdroj: Element (Element 61, 2020)

5 Záver

V našom článku sme sa zamerali na porovnanie dvoch jazykov aplikovaných na jednoduchú výpočtovú úlohu, len aby sme ukázali efektívnosť použitých jazykov. Na základe našich výsledkov sme demonštrovali, ktorý z vybraných jazykov je lepší z hľadiska výkonu. Existujú ďalšie aspekty, ktoré by sa mali zvážiť, a to je rozšírenosť konkrétneho jazyka a krivka učenia. Pri zohľadnení týchto možností je víťazom jazyk Python, kvôli jeho rozšírenosti a známosti. Jazyk Scala je pre nováčikov trochu komplikovaný na rýchle pochopenie a preto nie je veľmi známy mimo komunity vývojárov Scala. Veľkou výhodou pre nováčikov je predchádzajúca prax s jazykom Java. Tieto jazyky nie sú rovnaké, ale môžu zdieľať knižnice, sú silne typované, objektovo orientované (každá hodnota je objekt) a používajú Java Virtual Machine ako spoločné prostredie na vykonávanie bajtkódu. Na druhej strane Python je

celosvetovo uznávaný jazyk s veľkou základňou používateľov a rozsiahlou poznatkovou bázou na riešenie problémov. Učenie sa tohto jazyka nie je pre nováčikov také zložité ako Scala. Z našich kódových fragmentov možno pozorovať, že Python je ľahko čitateľný a jednoduchý. Tieto výhody a nevýhody môžu byť podporené alebo potlačené kombinovaním nástrojov ako Apache Spark, ktorý implementuje podporu pre Scala aj Python. Špeciálne pripravené knižnice by mohli zvýšiť efektívnosť Pythonu, ale stále nebude tak výkonný ako kompilované jazyky. Na záver pridávame obrázok najpoužívanejších jazykov v dátovej vede do konca roka 2020, veríme, že podobné trendy budú pokračovať aj v ďalších rokoch.

Obr. 4: Programovacie jazyky používané v dátovej vede

PROGRAMMING LANGUAGES FOR DATA SCIENCE

Platform	2019 % share	2018 % share	% change
Python	65.8%	65.6%	0.2%
R Language	46.6%	48.5%	-4.0%
SQL Language	32.8%	39.6%	-17.2%
Java	12.4%	15.1%	-17.7%
Unix shell/awk	7.9%	9.2%	-13.4%
C/C++	7.1%	6.8%	3.7%
Other programming and data languages	6.8%	6.9%	-17.1%
Scala	3.5%	5.9%	-41.0%
Julia	1.7%	0.7%	150.4%
Perl	1.3%	1.0%	25.2%
Lisp	0.4%	0.3%	46.1%
Javascript	6.8%	na	na

Zdroj: Jelvix (Naumenko, 2020)

Literatúra

1. Aven, J. (2018). Data Analytics with Spark Using Python. Pearson Education.
2. Esposito, D., & Esposito, F. (2020). Introducing Machine Learning. Pearson Education.
3. Element 61. (2020). How to use Python and Pandas while embracing the power of Spark. Element 61. https://www.element61.be/en/resource/how-use-python-and-pandas-while-embracing-power-spark.

4. Farth, T. (2018). Machine Learning Guide for Beginners with R/Python/Scala. CreateSpace Independent Publishing Platform.
5. Jancauskas, V. (2016). Scientific Computing with Scala. Packt Publishing.
6. Kohli, M. (2021). Basic Core Python Programming. BPB Publications.
7. Mehrotra, S., Grade, A. (2019). Apache Spark Quick Start Guide. Packt Publishing.
8. Naumenko, V. (2020). Top Data Science Programming Languages. Jelvix. https://jelvix.com/blog/top-data-science-programming-languages
9. Nordeen, A. (2020). Learn Hadoop in 24 Hours. Guru99.