Temat: Porównanie języka Java i Python w użyciu baz SQL i NoSQL.

Anna Żurczak 127227

Ocena efektywności systemów komputerowych – laboratoria

Projekt I

1. Wstęp

Zadanie dotyczyło porównania języków Java i Python w odniesieniu do obsługi baz SQL i NoSQL, a właściwie bibliotek które pozwalają na wykonywanie operacji na tych bazach. Jako baza SQL zostało wykorzystane SQLite, a jako baza NoSQL użyto MongoDB. Celem zadania było sprawdzenie, która para narzędzi najbardziej efektywnie – w najkrótszym czasie przy najniższym stopniu wykorzystania zasobów - wykona załadowanie bazy, danymi z pliku tekstowego.

2. Metodyka badania

Pomiary zostały wykonane dla 4 par : Java + SQLite, Python + SQLite, Java + MongoDB, Python + MongoDB.

Dane do załadowania bazy pochodzą z http://millionsongdataset.com/. Pomiary wykonano dla 5 rozmiarów plików – 50MB, 100MB, 150MB, 250MB, 500MB. Każdy plik ma następującą strukturę: user_id<SEP>song_id<SEP>date(timestamp).

W SQL stworzono dwie tabele: Samples (user_id, song_id, foreign key) i Dates(id, day, month, year). Plik czytano linia po linii, każdą krotkę wstawiano pojedynczo.

W MongoDB stworzono dwie kolekcje Samples (_id, user_id, song_id, key to date) i dates (_id, id, day, month, year). Plik czytano linia po linii. W Pythonie tworzony był json zawierający 1000 linii, a w Javie kolekcja zawierająca 1000 obiektów BasicDBObject.

Użyte biblioteki		
Język	SQLite	MongoDB
Java	sqlite-jdbc w wersji 3.30.1	mongo-java-driver w wersji 3.9.1
Python	sqlite3 w wersji 2.6.0	pymongo w wersji 3.9.0

Użyte bazy danych : MongoDB w wersji 4.2.2 i SQLite w wersji 3.30.1

Parametry komputera na którym wykonano pomiary :

System operacyjny	Windows 10
Procesor	Intel Core i5-7300HQ (4 rdzenie)
RAM	8.00 GB

Do pomiarów w pythonie użyto bibliotek: time, psutil i os, a z nich funkcji:

```
current_process = psutil.Process(os.getpid())
current_process.cpu_percent()
current_process.memory_info()[0] # [0] - rss(resident set size)
t_start = time.time()
```

Do pomiarów w Javie użyto klas OperatingSystemMXBean z pakietu com.sun.management, Runtime i System z java.lang.

```
long startTime = System.currentTimeMillis();
(com.sun.management.OperatingSystemMXBean) bean).getProcessCpuLoad()
Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory()
```

3. Wyniki

Jako miarę tendencji centralnej wybrano średnią, ponieważ suma wartości pomiarów ma sens i ich rozkład nie jest skośny. Jako miarę rozrzutu wybrano odchylenie standardowe – zostanie zaznaczone w postaci słupków błędu na wykresach poniżej.

3.1. Java

a. SQLite

Rozmiar pliku [MB]	Wykorzystanie procesora[%]	Pamięć [MB]	Czas[s]
50	13,76	69,68	5,44
100	19,68	80,49	7,26
150	17,90	93,23	12,95
250	19,05	66,11	19,66
500	17,57	81,81	46,46

Tabela 1 Java + SQLite - średnia

b. MongoDB

Rozmiar pliku [MB]	Wykorzystanie procesora[%]	Pamięć [MB]	Czas[s]
50	8,96	60,55	15,20
100	7,15	63,12	31,00
150	7,81	52,46	45,40
250	6,85	59,52	77,40
500	6,89	64,81	155,40

Tabela 2 Java + MongoDB - średnia

3.2. Python

a. SQLite

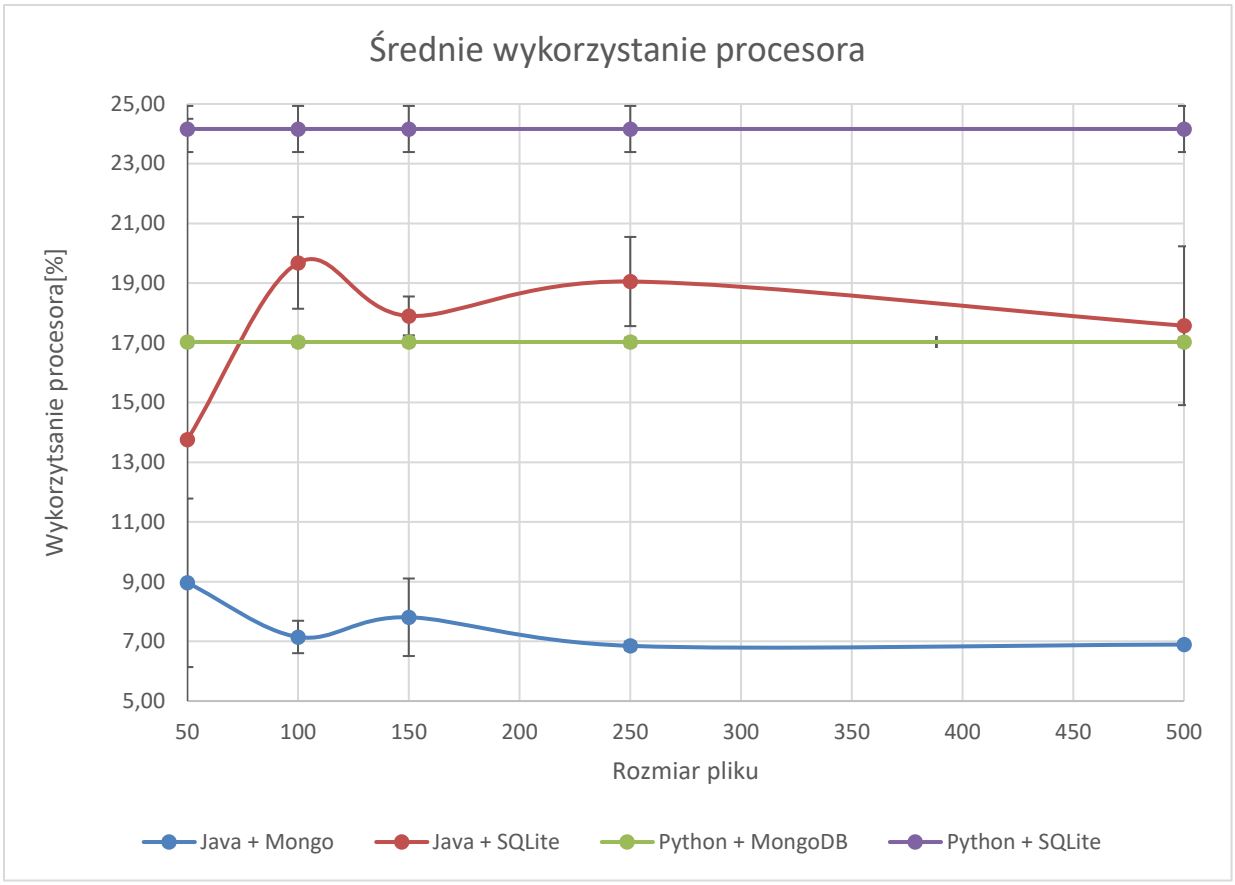
Rozmiar pliku [MB]	Wykorzystanie procesora[%]	Pamięć [MB]	Czas[s]
50	24,16	17,03	27,76
100	24,16	17,03	36,42
150	24,16	17,03	53,98
250	24,16	17,03	92,91
500	24,16	17,03	181,01

Tabela 3 Python + SQLite - średnia

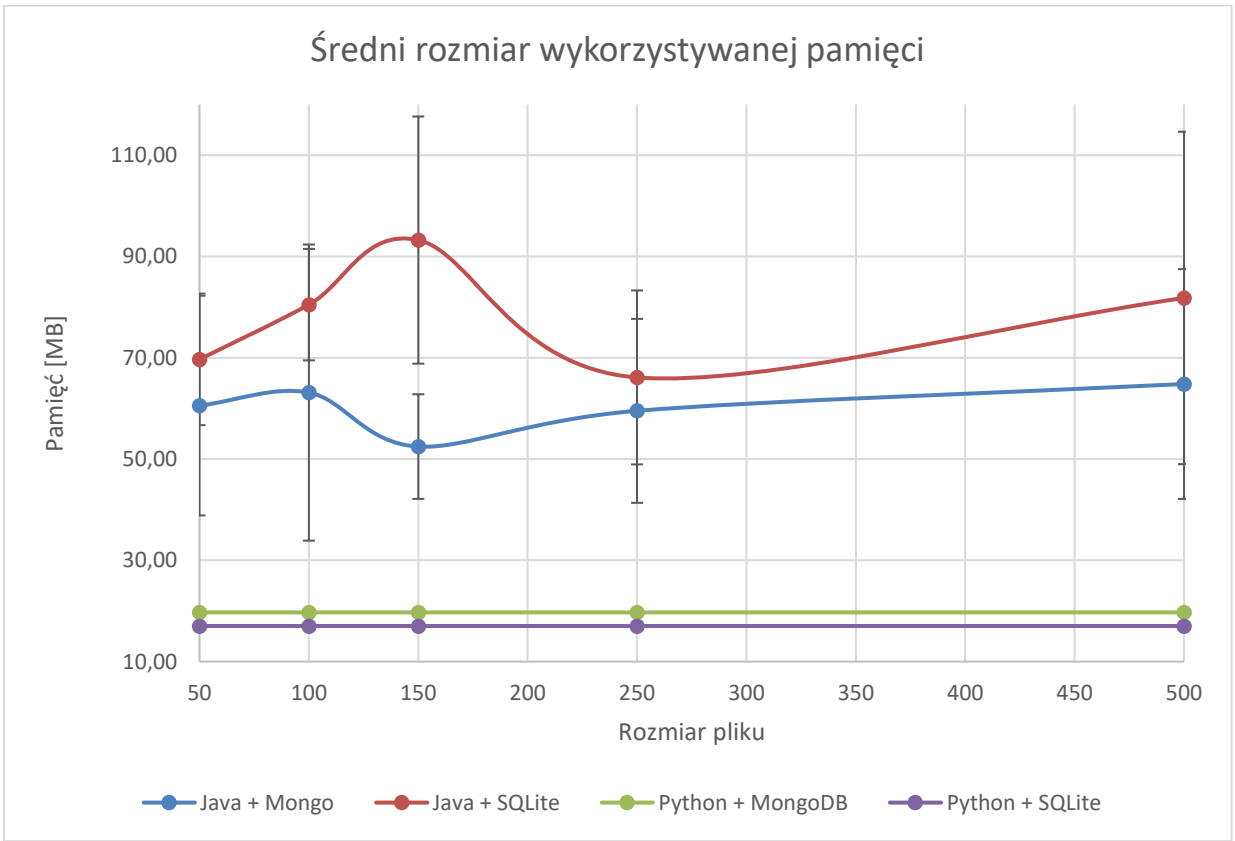
b. MongoDB

Rozmiar pliku [MB]	Wykorzystanie procesora[%]	Pamięć [MB]	Czas[s]
50	17,03	19,72	38,78
100	17,03	19,72	72,97
150	17,03	19,72	110,95
250	17,03	19,72	181,66
500	17,03	19,72	364,47

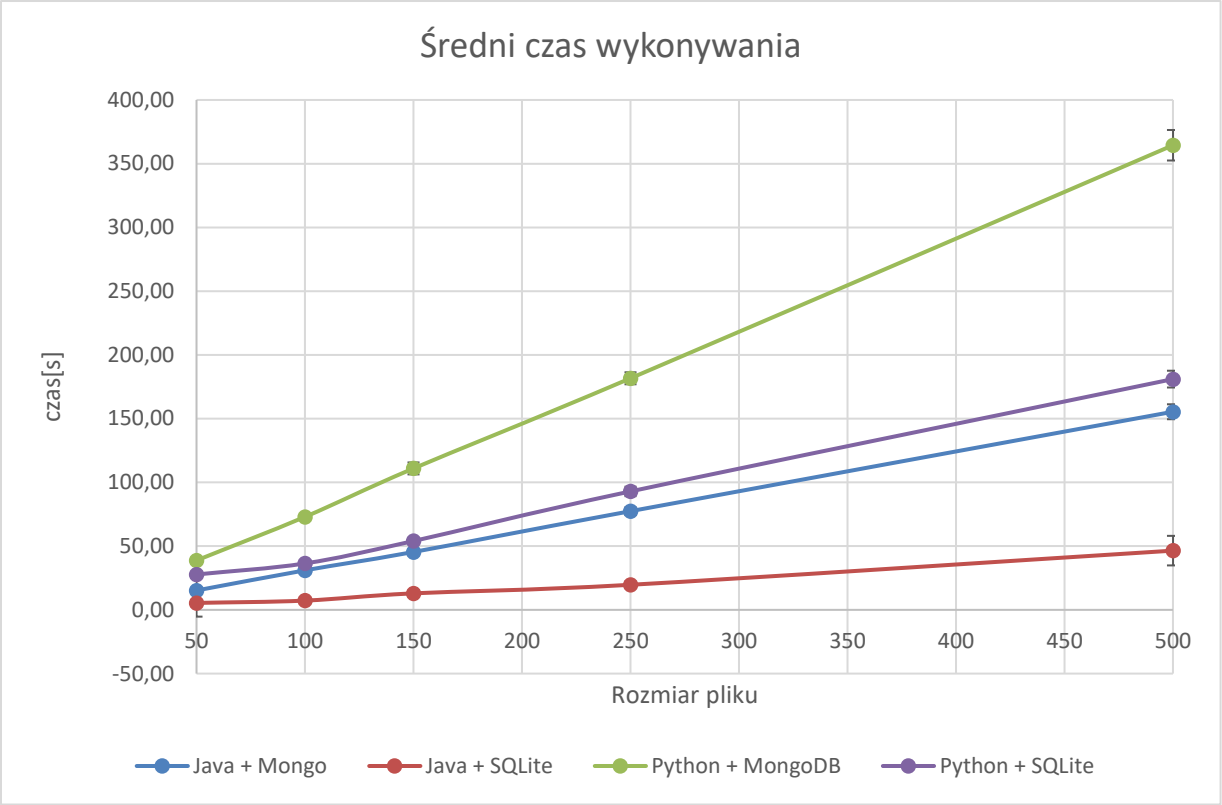
Tabela 4 Python + MongoDB - średnia



Wykres 1 Średnie wykorzystanie procesora



Wykres 2 Średni rozmiar wykorzystywanej pamięci



Wykres 3 Średni czas wykonywania

4. Podsumowanie

Java zużywa dużo pamięci, ale w krótkim czasie i przy niskim procencie wykorzystania procesora[Tabela 1]. Python ma zużycie pamięci mniejsze o prawie $^2/_3$ od Javy, ale czas wykonywania jest dłuższy o prawie $^3/_4$[Tabela 3].

Najbardziej efektywną parą okazała się para Java + MongoDB, wykorzystuje dużo więcej pamięci niż Python, jednakże robi to w krótkim czasie i przy 10-procentowym użyciu procesora.

Trzeba zwrócić uwagę na różnicę w pobieraniu danych, w Javie wartość wykorzystywanej pamięci w danej chwili to cała pamięć jaką wykorzystuje maszyna Javy, a w Pythonie to pamięć która zajmują zmienne i program – stąd linia prosta na Wykres 2.

Java jest szybsza, ponieważ jest językiem kompilowanym, Python jest językiem interpretowalnym, który określa rodzaj danych w czasie wykonania co czyni go wolniejszym. Połączenie tych języków z SQLite okazało się szybsze niż z MongoDB [Wykres 3].

Dla każdego przypadku zostało wykonane pięć pomiarów, a z nich obliczono odchylenie standardowe. Słupki błędu najbardziej widoczne są na Wykres 2, czyli średnim wykorzystaniu pamięci dla Javy. Dla czasu [Wykres 3] odchylenie jest bliskie zeru, a dla wykorzystania procesora [Wykres 1] największy rozrzut wystąpił dla SQLite.

Warto również porównać dla którego języka napisanie testów okazało się prostsze, można jednoznacznie stwierdzić, że w przypadku bazy SQLite i MongoDB biblioteki są prostsze w użyciu dla Pythona.