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.

1. 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.

Biblioteki użyte dla Javy : mongo-java-driver w wersji 3.9.1, sqlite-jdbc w wersji 3.30.1

Biblioteki użyte dla Pythona : pymongo w wersji 3.9.0 , sqlite3 w wersji 2.6.0

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

Parametry komputera na którym wykonano pomiary :

|  |  |
| --- | --- |
| System | 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()

current\_process.cpu\_percent()

current\_process.memory\_info()[0] # [0] – rss(resident set size)

t\_start = time.time()

t\_stop = time.time()

t\_stop – t\_start

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

long startTime = System.*currentTimeMillis*();

long stopTime = System.*currentTimeMillis*();

long elapsedTime = (stopTime - startTime)/1000;

(com.sun.management.OperatingSystemMXBean) bean).getProcessCpuLoad()

Runtime.*getRuntime*().totalMemory() - Runtime.*getRuntime*().freeMemory()

1. Wyniki
2. Miara tendencji centralnej – średnia

Jako miarę tendencji centralnej wybrano średnią, ponieważ suma wartości pomiarów ma sens i ich rozkład nie jest skośny.

* 1. Java

1. SQLite

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania procesora[%] | Pamięć [MB] | Czas[s] |
| 50 | 13,76 | 69,68 | 21,44 |
| 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 |

1. MongoDB

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania 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 |

* 1. Python

1. SQLite

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania 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 |

1. MongoDB

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania 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 |

1. Miara rozrzutu – odchylenie standardowe
   1. Java
2. SQLite

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania procesora[%] | Pamięć [MB] | Czas[s] |
| 50 | 10,74 | 12,99 | 10,75 |
| 100 | 1,54 | 11,00 | 0,62 |
| 150 | 0,65 | 24,40 | 0,70 |
| 250 | 1,49 | 17,18 | 2,49 |
| 500 | 2,66 | 32,82 | 11,59 |

1. MongoDB

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania procesora[%] | Pamięć [MB] | Czas[s] |
| 50 | 2,82 | 21,69 | 1,10 |
| 100 | 0,54 | 29,23 | 1,58 |
| 150 | 1,30 | 10,33 | 2,07 |
| 250 | 0,15 | 18,16 | 3,05 |
| 500 | 0,10 | 22,69 | 5,86 |

* 1. Python

1. SQLite

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania procesora[%] | Pamięć [MB] | Czas[s] |
| 50 | 0,77 | 0,07 | 11,77 |
| 100 | 0,77 | 0,07 | 1,44 |
| 150 | 0,77 | 0,07 | 2,31 |
| 250 | 0,77 | 0,07 | 3,72 |
| 500 | 0,77 | 0,07 | 6,58 |

1. MongoDB

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar pliku [MB] | Procent wykorzystania procesora[%] | Pamięć [MB] | Czas[s] |
| 50 | 0,15 | 0,03 | 1,54 |
| 100 | 0,15 | 0,03 | 1,85 |
| 150 | 0,15 | 0,03 | 4,68 |
| 250 | 0,15 | 0,03 | 4,70 |
| 500 | 0,15 | 0,03 | 11,97 |

1. Podsumowanie