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
3. Miara rozrzutu
4. Podsumowanie