Jaki format pliku jest najlepszy do obliczeń w PySparku

Sprawozdanie z projektu numer 1 z przedmiotu OESK

Abstrakt

Celem badań jest sprawdzenie który format plików csv, csv.gz czy snappy.parquet najlepiej współpracuje z PySparkiem w języku Python. Dane wejściowe przygotowane zostały uwzględniając liczbę wierszy, różnorodność danych oraz liczbę kolumn. Obliczenia wykonywane przy użyciu Sparka to operacja połączenia (join) danych samych ze sobą. Mierzone parametry to czas obliczeń, obciążenie procesora oraz zużycie pamięci operacyjnej.

Przygotowanie danych

Zdecydowano się na trzy rozszerzenia plików: csv jako najprostszy I najczęstszy tekstowy format danych, csv.gz jako skompresowany plik csv w formacie gzip, snappy.parquet jako kolumnowy binarny format danych z najczęściej używaną kompresją snappy.

Do testów przygotowano dane uwzględniając popularne właściwości, czyli liczbę wierszy oraz liczbę kolumn. Dodano także trzecią, która odnosi się do różnorodności danych. Pierwsze dwie bezpośrednio wpływają na rozmiar pliku. Trzecia ukazuje nam naturę danych, czyli to czy wartości są powtarzalne czy losowe. W każdej cesze zdecydowano się na dwie skrajne wartości. Liczba wierszy może być mała (500 000) albo duża (3 000 000). Liczba kolumn może być mała (9 kolumn) albo duża (27 kolumn). Liczba kolumn jest podzielna przez 3, ponieważ występują tylko trzy typy danych. Liczba całkowita z zakresu od 1 do 10 000 000. Tekst składający się zawsze z 20 znaków spośród małych i wielkich liter alfabetu łacińskiego. Data zapisana jako tekst w formacie „YYYY-MM-DD HH:MM:SS”. Różnorodność danych może być ustalona (5 możliwych wartości) albo dowolna (losowe wartości). Co w rezultacie daje 24 różne kombinacje (3x format pliku, 2x liczba wierszy, 2x liczba kolumn, 2x różnorodność danych) a tym samym liczbę przygotowanych plików wejściowych. Format csv zajmuje najwięcej miejsca ze względu na brak kompresji. Pozostałe dwa formaty posiadają kompresję, która ma różną efektywność w zależności od różnorodności danych.

Metodologia

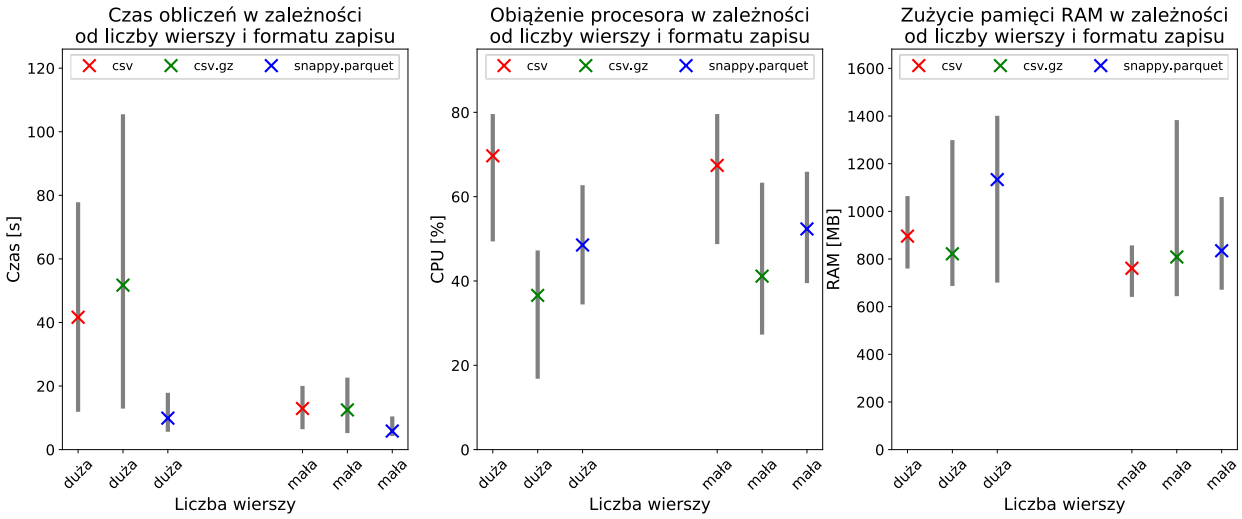
Do przeprowadzenia testów użyto Pythona (v.3.7.4) oraz pakietu PySpark (v.2.4.4). Do utworzenia plików w formacie csv oraz csv.gz użyto pakietu Pandas (v.0.25.3). Dla formatu snappy.parquet skorzystano dodatkowo z pakietu PyArrow (v.0.15.1).

Aby wyeliminować różnego rodzaju zapamiętane wartości przez PySparka zdecydowano się na uruchamianie testu dla każdego pliku osobno jako osobny proces. Na początku uruchamiany jest trzykrotnie proces, który tylko inicjuje sesję PySpark. To pozwala na uzyskanie średniego czasu konstrukcji takiego obiektu i pominięcie tego czasu w fazie pomiarów. W pojedynczym teście pomiarów dokonywano około 3 razy na sekundę. Dla obciążenia procesora obliczano średnią arytmetyczną, natomiast dla zużycia pamięci operacyjnej wybierano najwyższą wartość. Każdy test przeprowadzono 10 razy, aby wyeliminować błędy pomiarowe. Testy powtarzano całą grupą a nie pojedynczo, z czego wynika, że ten sam plik testowany był co 24 test.

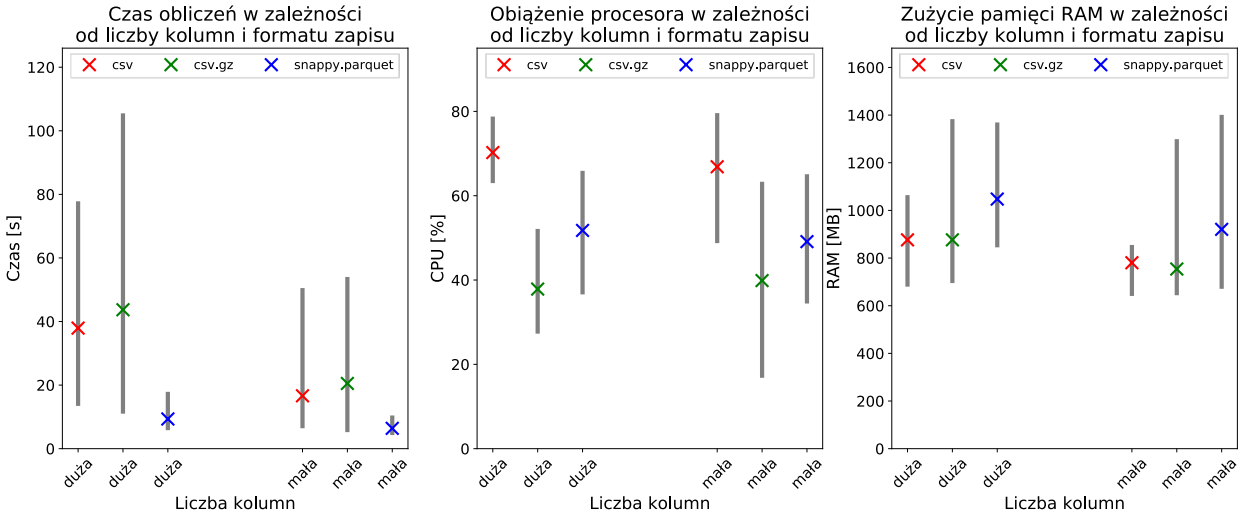
Sercem przeprowadzonego testu jest operacja połączenia (join) danego pliku samego ze sobą. Kluczem połączenia jest zawsze pierwsza kolumna. Lewy i prawy zbiór uprzednio są redukowane do jednej trzeciej swej szerokości poprzez selekcję losowych kolumn.

Rezultaty

Na każdym rysunku poniżej pokazane będą trzy wykresy. Miejsce zaznaczone znakiem „x” oznacza średnią wartość dla danej miary. Szary pasek ukazuje rozrzut danych, który rozpoczyna się w wartości minimalnej a kończy w wartości maksymalnej dla danej miary.



Rysunek 1 Rezultaty ze względu na liczbę wierszy

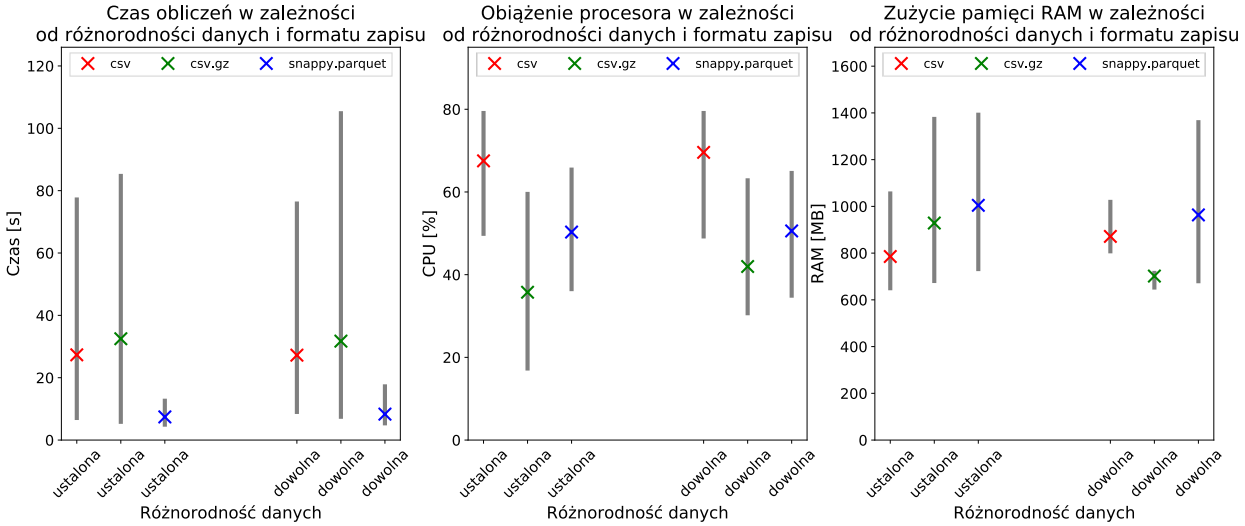


Rysunek 2 Rezultaty ze względu na liczbę kolumn

Patrząc na Rysunek 1 można zauważyć, że czas obliczeń istotnie zależy od liczby wierszy (rozmiaru pliku). Jednak format snappy.parquet jest zdecydowanym zwycięzcą. Środkowy wykres pokazuje, że nie istnieje żaden stosunek względem obciążenia procesora a liczbą wierszy. Dostrzec można natomiast korelację, że format csv generuję największe obciążenie a format csv.gz najmniejsze. Ostatni z wykresów trzeba interpretować trochę inaczej, ponieważ dla pamięci operacyjnej interesująca jest maksymalna wartość. To bezpośrednio wskazuje nam na minimalny potrzebny rozmiar pamięci operacyjnej dla danego testu. Warto tutaj zauważyć, że format csv potrzebuje

najmniej pamięci operacyjnej (prawdopodobnie przez brak kompresji, której dekompresja potrzebuje dodatkowych zasobów).

Rysunek 2 przedstawia podobne wykresy co Rysunek 1. Jest to prawdopodobnie związane z tym, że liczba kolumn oraz liczba wierszy bezpośrednio przyczyniają się do zmniejszenia/zwiększenia rozmiaru pliku.



Rysunek 3 Rezultaty ze względu na różnorodność danych

Jako ostatni Rysunek 3 obrazuje nam, że różnorodność danych nie wpływa znacząco na czas obliczeń czy obciążenie procesora. Wartym uwagi jest fakt, że losowe dane potrzebowały najmniej pamięci operacyjnej dla formatu csv.gz. Tutaj tak samo jak Rysunek 1 format snappy.parquet uzyskał najniższe czasy obliczeń a format csv.gz najniższe obciążenie procesora.

Podsumowanie

Jednoznacznie można stwierdzić, że najprostszy format csv jest najgorszym wyborem w powyższym zestawieniu ze względu na obciążenie procesora, ale stosunkowo dobrym ze względu na obciążenie pamięci operacyjnej. Najszybszym formatem okazał się snappy.parquet, który osiągał kilkukrotnie lepsze rezultaty. Zużycie procesora dla tego formatu jest mniejsze niż dla csv ale większe niż dla csv.gz. Natomiast obciążenie pamięci operacyjnej jest porównywalne z formatem csv.gz.

Wybór najlepszego formatu zależy oczywiście od indywidualnych potrzeb i możliwości. Najczęściej obliczenia w PySparku wykonuje się na osobnym komputerze przeznaczonym do obliczeń gdzie możemy sobie pozwolić na duże zużycie pamięci operacyjnej oraz obciążenie procesora, a wtedy najszybszy okazuje się format snappy.parquet. Jeżeli nie możemy wygenerować danych wejściowych w tym formacie lepszym rozwiązaniem będzie csv.gz niż csv.