Metody Programowania Równoległego
Map Reduce

Maciej Trątnowiecki

AGH, Semestr Letni, 2022

1 Sformułowanie problemu

W ramach zadania przygotowałem sekwencyjną implementację algorytmu zliczającego wystąpienia słów w tekście. Następnie wykonałem pomiary czasu wykonania zarówno wspominanej wersji sekwencyjnej jaki i przygotowanej w ramach laboratorium implementacji dla frameworka hadoop.

2 Środowisko uruchomieniowe

Wersję równoległą uruchamiałem w środowisku *EMR* z chmury *AWS*. Wersję sekwencyjną na maszynie wirtualnej z usługi *EC2* w tej samej chmurze.

Usługa	Nazwa instancji	Ilość Węzłów	Ilość rdzeni na węzeł	Sumaryczna ilość rdzeni
EC2	m4.large	1	2	2
EMR	m4.large	12	2	24
EMR	m4.xlarge	6	4	24

Tabela 1: Konfiguracja instancji.

Zauważyć należy, że *m4.large* i *m4.xlarge* oferują identyczne procesory wirtualne, lecz różnią się ich ilością i konfiguracją.

3 Implementacja sekwencyjna

Implementację sekwencyjną algorytmu wykonałem w języku *C++*. Wczytuje ona dane wyjściowe z jednego pliku, a wyniki zapisuje w drugim. Kod źródłowy załączam poniżej.

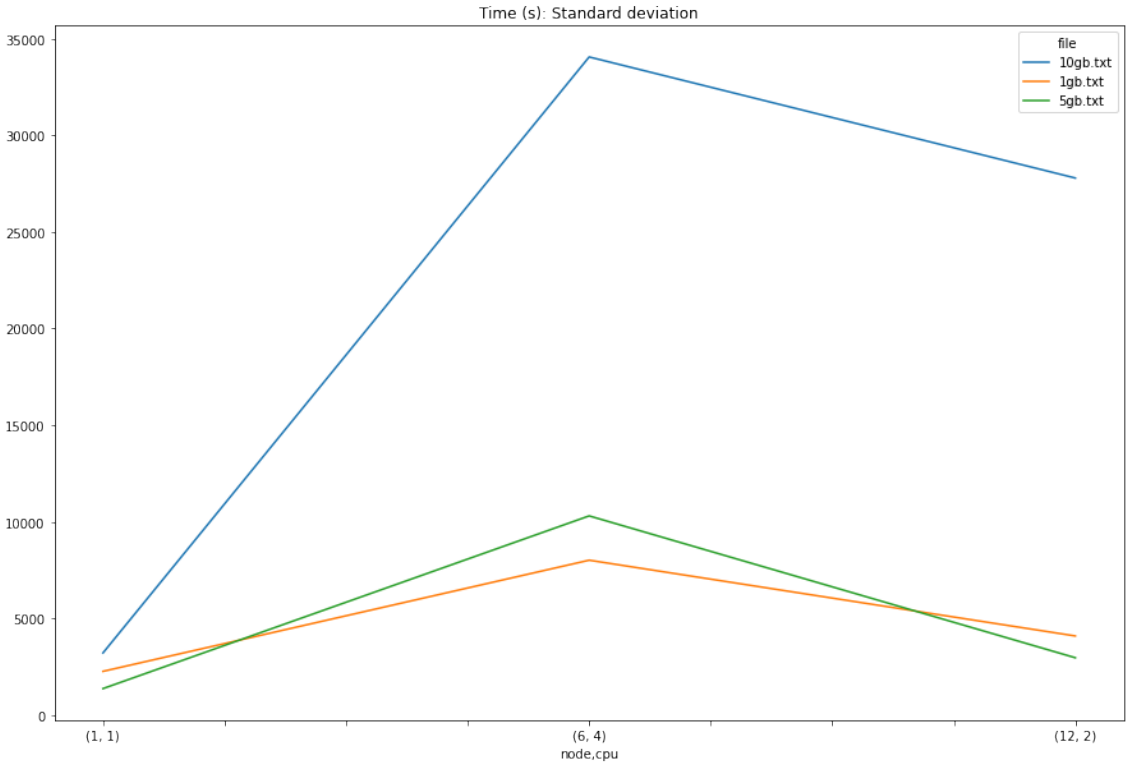
```cpp
#include<bits/stdc++.h>
using namespace std;

int main(int argc, char** argv){
    const string result_filename = "output.txt";
    std::ifstream file(argv[1]);
    unordered_map<string, int> counter;
    if ( file.is_open() ) {
        string token;
        while ( file.good() ) {
            file >> token;
            auto insert_result = counter.insert({token, 1});
            if (!insert_result.second)
                insert_result.first->second = (insert_result.first->second)+1;
        }
    }
    ofstream result_file(result_filename);
    for (auto const& entry : counter)
        result_file<<entry.first<<':'<<entry.second<<endl;
    result_file.close();
}
```

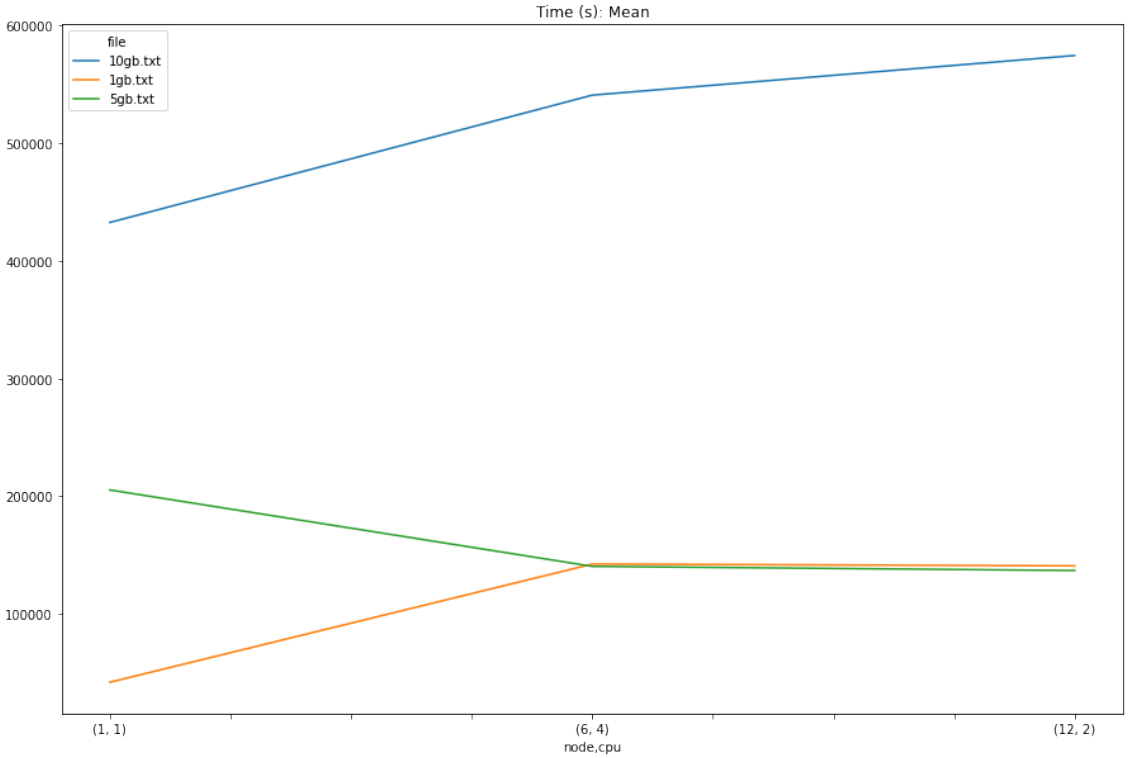
../counter.cpp

4 Pomiary

Przygotowałem pomiary czasu wykonania algorytmu dla danych wejściowych rozmiaru kolejno *1GB*, *5GB* i *10GB*. Każdy z pomiarów uruchomiłem dziesięciokrotnie dla każdej z konfiguracji sprzętowych opisanych w tabeli powyżej. Wyniki przedstawiłem na wykresach.

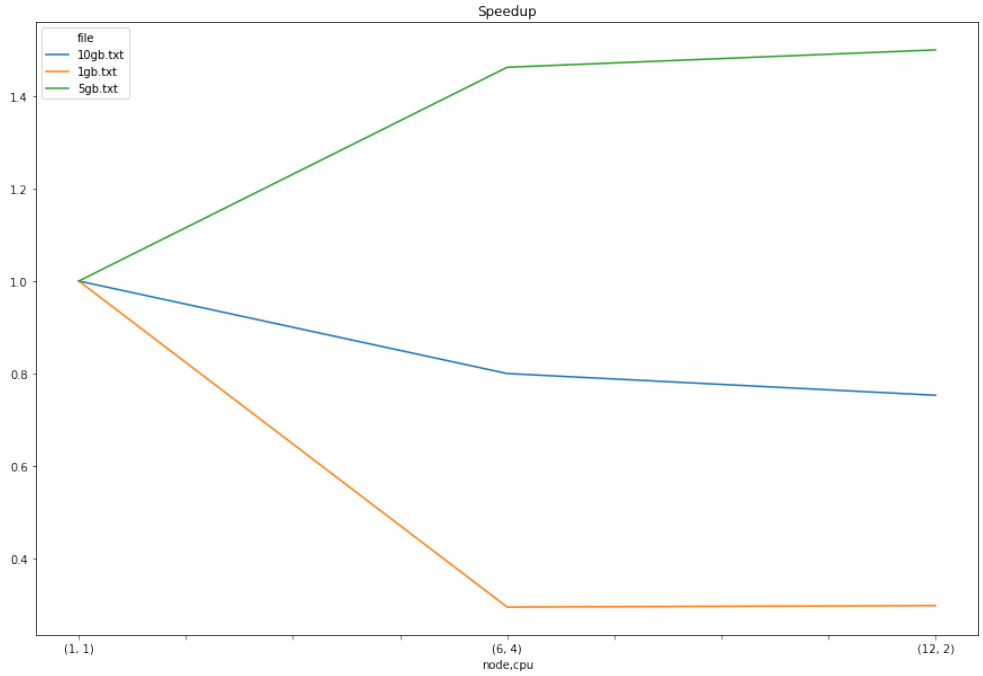


Rysunek 1: Odchylenie standardowe pomiarów



Rysunek 2: Średni czas wykonania programu w *ms*

Następnie obliczyłem przyśpieszenie programu dla kolejnych konfiguracji sprzętowych, w oparciu o średni czas wykonania programu.



Rysunek 3: Przyśpieszenie

5 Wnioski

Łatwo zauważyć, że implementacja sekwencyjna paradoksalnie okazała się szybsza od implementacji równoległej dla wszystkich konfiguracji i rozmiarów plików, za wyjątkiem pliku o rozmiarze *5GB*.

Potencjalnie wynika to z faktu, że wykorzystałem dobrze zoptymalizowaną, kompilowaną implementację sekwencyjną, która naturalnie lepiej radzi sobie z przetwarzaniem dużych rozmiarów danych niż te wykonane w językach interpretowanych. W przypadku plików odpowiednio małych (w moim przypadku *1GB*), ta specyfika jest wystarczająca do otrzymania lepszych wyników. W przypadku plików odpowiednio dużych (w moim przypadku *10GB*), występujące obciążenia komunikacyjne mogą potencjalnie zajmować więcej czasu niż samo wykonanie obliczeń. W przypadku średnim natomiast lepsza skalowalność rozwiązania równoległego pozwoliła na otrzymanie lepszych wyników i krótszych czasów wykonania niż sekwencyjna. Potencjalnie moglibyśmy otrzymać lepsze wyniki, dla największego pliku, mając dostęp do mocniejszych węzłów obliczeniowych, a przez to obniżając narzuty wynikające z komunikacji pomiędzy węzłami.

Warto zauważyć, że generalnie konfiguracja sprzętowa o mniejszej ilości mocniejszych węzłów obliczeniowych skutkuje niższymi czasami wykonania niż konfiguracje o większej liczbie słabszych węzłów, pomimo że suma rdzeni obliczeniowych jest w obu przypadkach taka sama. Jest to efekt oczekiwany, ze względu na występujące w nich mniejsze obciążenie wynikające z komunikacji pomiędzy węzłami.

Dla tak zdefiniowanego problemu obliczeniowego nie da się wskazać wartości metryki *COST* na podstawie danych zebranych w ramach zadania. Dla dwóch z rozmiarów plików wejściowych implementacja równoległa osiągnęła gorsze niż sekwencyjna czasy wykonania, co wyklucza wskazanie konkretnej wartości metryki. W przypadku średniego rozmiaru pliku możemy wskazać, że wartość metryki jest mniejsza niż *24* rdzenie, choć jej konkretna wartość nie wynika z wykresu (potencjalnie mniejsza ilość rdzeni pozwoliłaby na przecięcie progu w postaci czasu sekwencyjnego wykonania obliczeń.