Universitatea Tehnică ”Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
Domeniul Calculatoare și Tehnologia Informației
Specializarea Tehnologia Informației

Algoritmul MapReduce

TEMĂ DE CASĂ LA DISCIPLINA ALGORITMI PARALELI ȘI DISTRIBUIȚI

Profesor îndrumător,
Adrian Alexandrescu

Student,
Pristanda Amalia-Maria, Grupa 1409B

An universitar 2022-2023

CUPRINS

Enunțul temei de casă

În cadrul oricărui sistem de regăsire a informaţiilor, colecţia de date ţintă este reorganizată pentru a optimiza funcţia de căutare. Un exemplu în acest sens este dat chiar de motoarele de căutare a informaţiilor pe Web: colecţia de documente este stocată sub forma unui index invers.

Paşii implicaţi în construirea unui astfel de index invers sunt următorii:

- ➔ fiecare document din cadrul colecţiei ţintă (identificat printr-un docID) va fi parsat şi spart în cuvinte unice (sau termeni unici); se obţine în finalul acestui pas o listă de forma < docIDx , {term1 : count1, term2 : count2 , . . . , termn : countn} > (***index direct*** – countk înseamnă numărul de apariţii al termenului k);
- ➔ fiecare listă obţinută în pasul anterior este spartă în perechi de forma: < docIDx , {termk : countk} >; pentru fiecare astfel de pereche, se realizează o inversare de valori, astfel încât să obţinem: < termk , {docIDx : countk} >;
- ➔ perechile obţinute în pasul anterior sunt sortate după *termk* (cheie primară), *docIDx* (cheie secundară);
- ➔ pentru fiecare termk se reunesc < termk , {docIDx : countk} >, astfel încât să obţinem: < termk , {docIDk1 : countk1, docIDk2 : countk2, . . . , docIDkm : countkm} > (***indexul invers***).

Tema de casă constă în implementarea unei soluţii MPI de tip MapReduce pentru problema construirii unui index invers pentru o colecţie de documente text.

Aplicaţia de test va primi ca parametri de intrare numele unui director ce conţine fişiere text (cu extensia ”.txt”) şi un nume de director pentru stocarea datelor de ieşire şi va genera pe post de răspuns un set de fişiere text ce conţin indexul invers corespunzător colecţiei de documente de intrare.

Capitolul 1. MapReduce - Noțiuni teoretice

MapReduce este o paradigmă de programare care permite scalabilitate masivă pe sute sau mii de servere dintr-un cluster Hadoop.
În general, se consideră că acest model implică existența unui nod de procesare cu rol de coordonator și mai multe noduri de procesare cu rol de worker.

1.1. Etapele MapReduce

Termenul „MapReduce” se referă la două sarcini separate şi distincte pe care le efectuează programele Hadoop [2]:

- Etapa de mapare:
  - nodul cu rol de coordonator împarte problema „originală” în sub probleme şi le distribuie către *workeri* pentru procesare.
  - divizarea caracteristică acestei etape nu trebuie să coreleze efectiv dimensiunea datelor de intrare cu numărul de *workeri* din sistem; un *worker* poate primi mai multe sub-probleme de rezolvat;
- Etapa de reducere:
  - nodurile cu rol de worker determina solutiile sub-problemelor identificate în faza de mapare/selecție

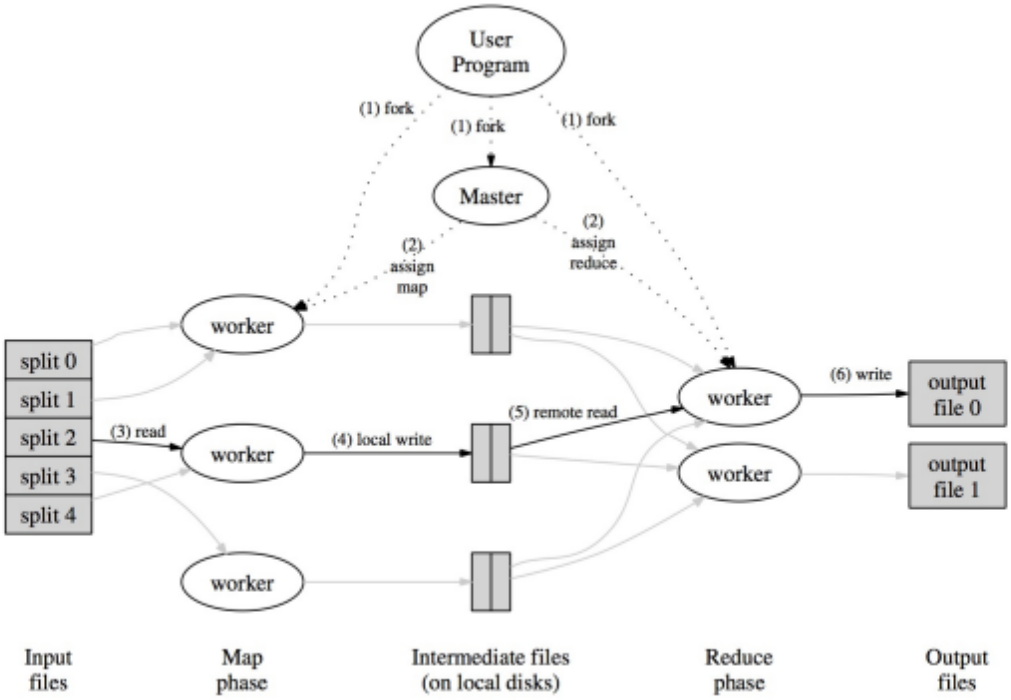

Figura 1 (preluare din [4])

1.2. Etapele “rafinate” conform Michael Kleber

Michael Kleber (Google Inc.) stabileşte in [3] etapele implicate de paradigma MapReduce după cum urmează:

1. pre-procesare – datele sunt pregătite pentru funcţia de mapare;
2. mapare – stabilirea datelor de interes;
3. amestecare şi sortare – datele pot fi organizate astfel încât să fie optimizată etapa de reducere;
4. reducere – determinarea rezultatului;
5. stocare rezultat.

Capitolul 2. Prezentarea soluției

Algoritmului în limbajul Python datorită bibliotecilor care ușurează munca procesării datelor. Este necesară instalarea bibliotecilor mpi4py (*pip install mpi4py*) și filelock (*pip install filelock*). Rularea programului se face cu ajutorul comenzii *mpiexec -n noOfProcess python main.py* în directorul unde se află fișierul main.py, unde noOfProcess reprezintă numărul de procese disponibile programului.

Soluția propusă respectă pașii prezentați în subcapitolul 1.2. Procesul 0 are rolul de coordonator în timp ce celelalte procese au rolul de workeri. Se consideră că prima jumătate din numărul de procese se ocupă de mapare în timp ce restul de reducere.

Rezultatele algoritmului se vor afla in directorul dat ca parametru de intrare, fiecare cuvânt găsindu-se în ordine alfabetică în fișierul corespunzător primei sale litere.

Capitolul 3. Pseudocod

În acest capitol este prezentat pseudocodul funcțiilor principale din soluția implementată.

Funcția principală a programului. Din aceasta se apelează celelalte metode.

```
Function main()
    citeste parametrii de intrare
    If rank = 0 Then
      trimite catre procesele care se ocupa de mapare fisierele corespunzatoare fiecaruia
      primeste confirmarea terminarii etapei de mapare

      trimite catre procesele care se ocupa de reducere fiesierele corespunzatoare fiecaruia
      primeste confirmarea terminarii etapei de reducere
    Else
        If procesul se ocupa de mapare Then
            primeste de la procesul 0 fisierele files de care se va ocupa
            pre_processing(files)
            mapping(files)
            trimite catre procesul 0 mesaj de finalizare
        Else
            primeste de la procesul 0 fisierele files de care se va ocupa
            reducing(files)
            trimite catre procesul 0 mesaj de finalizare
        Endif
    Endif
Endfunction
```

Funcția care pregătește datele pentru mapare. Fiecare proces elimină caracterele nedorite din fișierele atribuite. Aceasta primește un parametru, ‘files’, care reprezintă fișierele de care se ocupă un proces.

```
Function pre_processing(files):
    For file in files Do
        elimina caracterele nedorite din fisierul file
    Endfor
Endfunction
```

Funcția care se ocupă de stabilirea datelor de interes. Aceasta primește un parametru, ‘files’, care reprezintă fișierele de care se ocupă un proces.

```
Function mapping(files):
    For file in files Do
        se iau toate cuvintele din fisierul file
        se scrie intr-un fisier intermediar rezultatul <cuvant fileID 1>
    Endfor
Endfunction
```

Funcția care calculează numărul de apariții al fiecărui cuvânt din fiecare fișier, sortându-le ulterior în ordine alfabetică și după fileID. Aceasta primește un parametru, ‘files’, care reprezintă fișierele de care se ocupă un proces.

```
Function reducing(files):
    For file in files Do
        se iau toate cuvintele din fisier si se numara aparitiile fiecaruia
        rezultatul este pus intr-un dictionar
        se ordoneaza dictionarul dupa cuvant si fileID, obtinandu-se sorted_dictionary
        se apeleaza final_operation(sorted_dictionary)
    Endfor
Endfunction
```

Funcția care obține indexul invers și scrie rezultatul final în fișierele finale. Aceasta primește un parametru ‘sorted_dictionary’, care reprezintă dicționarul obținut în urma apelului funcției ‘reducing’.

```
Function final_operation(sorted_dictionary):
    For key in sorted_dictionary Do
        se creeaza indexul invers de forma <cuvant, {fileID:count}>
        rezultatul este salvat intr-un dictionar, final_dictionary
    Endfor

    For key in final_dictionary Do
        rezultatul este scris in fisierul final, aflat in directorul dat ca argument de intrare
    Endfor
Endfunction
```

Capitolul 4. Bibliografie

[1] IBM Corp. What is MapReduce? https://www.ibm.com/analytics/hadoop/mapreduce.
[2] Wikipedia. MapReduce. http://en.wikipedia.org/wiki/MapReduce, November 2013.
[3] Michael Kleber. The MapReduce paradigm.
https://sites.google.com/site/mriap2008/lectures, January 2008.
[4] Michael Kleber. What is MapReduce?
https://sites.google.com/site/mriap2008/lectures, January 2008.