UNIVERSITATEA TEHNICĂ "GHEORGHE ASACHI" DIN IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SPECIALIZAREA TEHNOLOGIA INFORMAȚIEI

DISCIPLINA ALGORITMI PARALELI ȘI DISTRIBUIȚI

TEMĂ DE CASĂ

Algoritmul MapReduce

Coordonator,

Ș.l.dr.ing. Adrian Alexandrescu

Studentă,

Nicoleta-Ecaterina Mihalache

Iași, 2022

Cuprins

Capitolul 1. Noțiuni teoretice MapReduce

MapReduce este o tehnică de procesare şi un model de programare pentru calculul distribuit. Termenul „MapReduce” se referă la un tipar de dezvoltare a aplicaţiilor paralele/distribuite ce procesează volume mari de date. În general, se consideră că acest model implică existenţa unui nod de procesare cu rol de coordinator şi mai multe noduri de procesare cu rol de workeri.

Etapele algoritmului MapReduce

Algoritmul MapReduce conţine 2 etape de lucru importante: „Map” (Maparea) şi „Reduce” (Reducerea):

1. Etapa de mapare – preia un set de date şi îl converteşte într-un alt set de date, unde elementele individuale sunt „sparte” în tuple (adică perechi cheie/valoare).

a) nodul cu rol de coordonator împarte problema „originală” în sub probleme şi le distribuie către workeri pentru procesare.

b) trebuie reţinut faptul că această divizare a problemei de lucru (a datelor de procesat) se realizează într-o manieră similară divide-et-impera – în unele cazuri nodurile worker pot divide la rândul lor sub-problema primită şi pot trimite aceste subdiviziuni către alţi worker, rezultând o arhitectură arborescentă;

c) 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;

2. Etapa de reducere – preia ieşirea de la etapa de mapare ca fiind datele de intrare şi combină aceste tuple, rezultând un alt set de tuple, dar de dimensiuni mai mici.

a) nodul cu rol de coordinator colectează soluţiile sub-problemelor şi le combină pentru a obţine rezultatul final al procesării dorite.

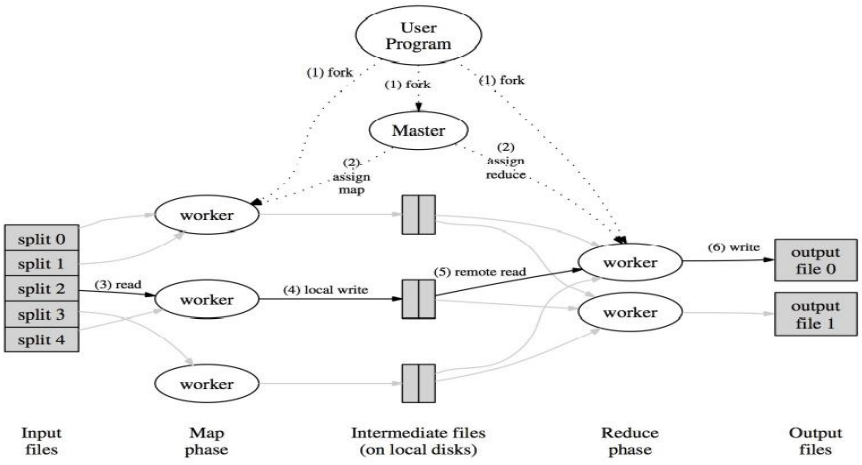

Figura 1: Paradigma MapReduce (preluare din [2])

Capitolul 2. Pseudocod

Pentru implementarea algoritmului MapReduce am folosit limbajul de programare Java la care am importat biblioteca specifică pentru programarea paralelă mpiJava (https://sourceforge.net/projects/mpjexpress/).

În prima etapă, etapa de mapare, se citesc numele tuturor fișierelor text ce vor fi prelucrare din folderul pus la dispoziție pentru testarea programului. Algoritmul începe cu un fir de execuție care împarte sarcinile de lucru și trimite mai departe lista cu numele fișierelor ce vor fi mapate ulterior de mai multe fire de execuție în parte. Maparea efectivă a fișierelor se face de către fiecare worker în parte. Acesta primește numele fișierului, îl deschide, citește textul și elimină semnele de punctuație rezultând o lista de cuvinte. Acestea vor fi filtrare după inițială și puse într-un nou fișier temporar numit după inițială, împreună cu numele fișierului din care a fost citit.

A doua etapă, cea de reducere primește ca date de intrare fișierele temporare create în etapa de mapare pe care le comprimă. Nodul cu rol de coordinator/ master-ul primește mesaj de de la workeri că au terminat filtrarea și preia datele de la aceștia. Din toate fișierele obținute în etapa de mapare crează un singur fișier în care comprimă toate rezultatele: voi avea fiecare cuvânt cu fișierul din care a fost citit și frecvența sa pe fiecare fișier.

Funcție citire din fișier și eliminare semne de punctuație

```
public static Map<String, List<String[]>> readWordsFromAFile() throws
IOException {
    List<File> files = readFilesFromAFolder(folder);
    List<String[]> wordsList = new ArrayList<>();
    Map<String, List<String[]>> map = new HashMap<>();

    for (File file : files) {
        //System.out.println(file.getName());
        BufferedReader br;
        FileReader fr = new FileReader(file);
        try {
            br = new BufferedReader(fr);
            String line = br.readLine();
            while (line != null) {
                wordsList.add(line.split("[, =?!-`ï)%~({»|_}…-¹áàæéèïñóüž)—
§¸©’"`¢¥1234567890@\t/:;\"',*+&^$#;¿.\n@]+"));
                line = br.readLine();
            }
            map.put(file.getName(), wordsList);
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        fr.close();
    }
    return map;
}
```

Funcție de mapare

```
public static void Mapper() throws IOException {
    Map<String, List<String[]>> map = readWordsFromAFile();

    Set keys = map.keySet();
    Iterator i = keys.iterator();
    while (i.hasNext()) {
        System.out.println(i.next());
    }

    Collection<List<String[]>> getValues = map.values();

    for (List<String[]> words : getValues) {

        for (String[] word : words) {

            for (String w : word) {
                writeAWordToAFile(w);
            }
        }
    }
}
```

Capitolul 3. Exemplu

Pentru a explica în detaliu cum funcționează MapReduce, vom folosi un exemplu concret: cel de numărare a cuvintelor, frecvența cu care apar cuvintele în diferite fișiere.

MapReduce este folosit în acest caz pentru a număra aparițiile fiecărui cuvânt într-un text. La intrare sunt 3 fișiere (f1.txt, f2.txt, f3.txt), deci pentru fiecare fișier se va crea un bloc. Dacă numărul task-urilor de reducere este 1, datele intermediare generate de sarcinile de mapare sunt ordonate (sort) și toate elementele care au aceeași cheie sunt grupate (merge) și transmise mai departe către sarcina care se ocupă cu reducerea.

-worker1 se ocupă de f1.txt (Ana are mere.);

- worker2 se ocupă de f2.txt (Maria are mure mure.);

-worker3 se ocupă de f3.txt (Ioana are prune.);

La finalul etapei de mapare vom avea următoarele fișiere, puse în directorul temp:

- a.txt:
  - ana, f1.txt
  - are, f1.txt
  - are, f2.txt
  - are, f3.txt
- i.txt:
  - ioana, f3.txt

- m.txt:
  - mere, f1.txt
  - maria, f2.txt
  - mure, f2.txt
  - mure, f2.txt
- p.txt:
  - prune, f3.txt

La finalul etapei de reducere vom avea un sigur fișier cu rezultatele pus în directorul final:

- rezultate.txt:
  - ana
    - f1.txt, 1
  - are
    - f1.txt, 1
    - f2.txt, 1
    - f3.txt, 1
  - ioana,
    - f3.txt, 1
  - mere,
    - f1.txt, 1
  - maria
    - f2.txt, 1
  - mure,
    - f2.txt, 2
  - prune
    - f3.txt, 1

Capitolul 4. Bibliografie

https://ro.wikipedia.org/wiki/MapReduce

https://www.ibm.com/topics/mapreduce

https://edu.tuiasi.ro/pluginfile.php/72101/mod_resource/content/1/alpd_proiect.pdf

http://mpjexpress.org/docs/guides/linuxguide.pdf