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

Algoritmi paraleli și Distribuiți
Implementarea unei soluții MPI de tip MapReduce

Coordonator,
Ș.l.dr. Alexandru Archip

Student,
Modiga Camelia-Maria, 1410B

An universitar 2021-2022

Cuprins

Enunțarea temei **3**

Considerente teoretice ale algoritmului **3**

Exemplu de aplicare a Map-Reduce pentru Word Counter **4**

Etapele MapReduce **4**

Prezentarea soluției **5**

Implementare și pseudocod **6**

Bibliografie **8**

1. Enunțarea temei

În cadrul oricărui sistem de regăsire a informațiilor, colecția de date țintă este reorganizată (sau re-modelată) pentru a optimiza funcția de căutare. Un exemplu în acest sens este dat chiar de motoarele de căutare a informațiilor pe Web: colecția de documente este stocată sub forma unui **index invers**. Pașii implicați în construirea unui astfel de **index invers** sunt următorii:

1. fiecare document din cadrul colecției țintă (identificat printr-un *docID*) va fi parsat și spart în cuvinte unice (sau termeni unici); se obține în finalul acestui pas o listă de forma <$docID_x$,{$term_k$: $count_1$, $term_2$: $count_2$, …,$term_n$:$count_n$}> (**index direct** – $count_k$ înseamnă numărul de apariții al termenului *k*);
2. fiecare listă obținută în pasul anterior este spartă în perechi de forma: <$docID_x$,{ $term_k$:$count_k$ }>; pentru fiecare astfel de pereche, se realizează o inversare de valori astfel încât să obținem: <$term_k$,{$docID_x$: $count_k$ }>;
3. perechile obținute în pasul anterior sunt sortate după $term_k$ (cheie primă), $docID_x$(cheie secundară);
4. pentru fiecare $term_k$ se reunesc <$term_k$,{$docID_x$: $count_k$ }> astfel încât să obținem: <$term_k$,{$docID_{k1}$: $count_{k1}$,$docID_{k2}$:$count_{k2}$,…,$docID_{km}$ $count_{km}$ }> (**indexul invers**)

Tema de casă constă în implementarea unei soluții MPI de tip **MapReduce** pentru problema construirii unui index invers pentru o colecție de documente text. Aplicația de test va primi ca **parametrii de intrare numele unui director ce conține fișiere text** (cu extensia ".txt") și un **nume de director pentru stocarea datelor de ieșire** și va genera pe post de răspuns un **set de fișiere text** ce conțin **indexul invers** corespunzător colecției de documente de intrare.

2. Considerente teoretice ale algoritmului

MapReduce este o paradigmă de programare pentru calculul distribuit. Acest model implică, în general, existența unui nod de procesare cu rol de coordonator (sau master sau inițiator) și mai multe noduri de procesare cu rol de worker.

MapReduce simplifică procesul complex de procesare masivă paralelă a seturilor de date prin furnizarea unui model de design care instruiește algoritmii să exprime informații sub formă de hartă și să reducă fazele. Maparea poate fi utilizată pentru a efectua transformări simple pe date, iar reducerea este utilizată pentru a grupa datele împreună și pentru a efectua agregări.

3. Exemplu de aplicare a Map-Reduce pentru Word Counter

Pentru simplitate, să luăm în considerare câteva cuvinte dintr-un document text. Vrem să aflăm numărul de apariție a fiecărui cuvânt.

În prima etapa cuvintele se spară în vederea mapării, apoi se mapează cuvintele sub forma perechilor cheie-valoare < $term_k$: $count_k$ >. Astfel sunt create perechi numite tuple. În primul nod din etapa de mapare ajung 3 cuvinte: Deer, Bear și River iar ieșirea nodului este reprezentată de 3 chei distincte care au valoarea unu. Procesul de mapare rămâne același pentru toate nodurile iar tuplele rezultate din acest proces sunt trimise în etapa de reducere. Pentru ca etapa de reducere să fie mai eficientă toate tuplele cu aceeași cheie sunt trimise aceluiași nod. Funcția de reducere adună toate valorile pentru o anumită cheie și astfel este calculat numărul de apariții al cuvântului. În exemplu există două perechi cu cheia ”Bear” care sunt reduse la o singură tuplă care are drept cheie cuvântul iar ca valoare numărul de apariții ale acestuia.Toate tuplele sunt apoi colectate și scrise în fișierul de ieșire.

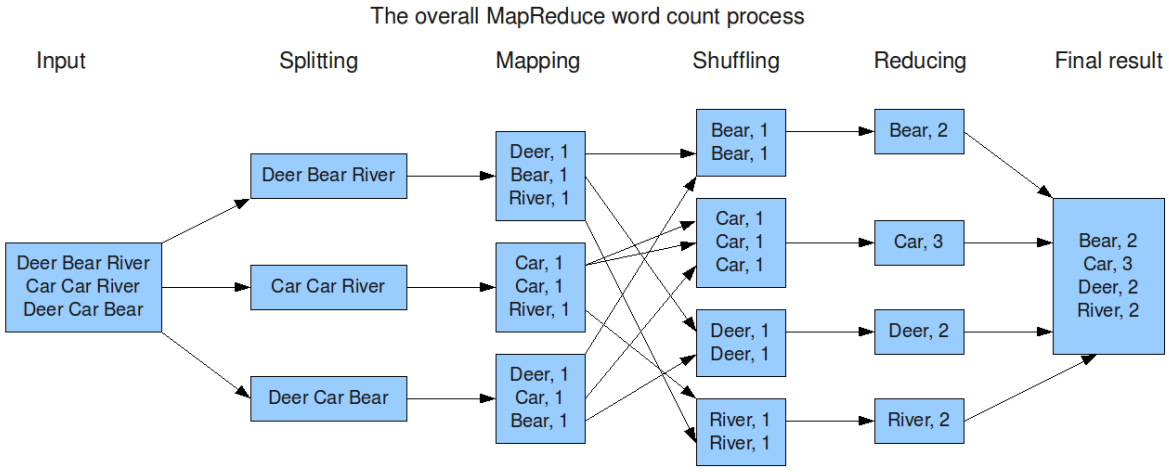

Figura 1: Exemplu de aplicare a Map-Reduce pentru Word Counter

4. Etapele MapReduce

Modelul **MapReduce** cuprinde două etape: Map (task-ul de mapare) și Reduce (task-ul de reducere):

1. Etapa de **mapare** – primește un set de date de intrare și îl convertește în alt set de date, unde elementele individuale sunt „sparte” în perechi cheie-valoare

- nodul cu rol de coordonator împarte problema „originală” în subprobleme și le distribuie către workeri pentru procesare;
- divizarea problemei de lucru (a datelor de procesat) se realizează într-o manieră similară divide-et-impera – în unele cazuri nodurile worker pot diviza la rândul lor subproblema primită şi pot trimite aceste subdiviziuni către alţi workeri, rezultând o arhitectură arborescentă;
- 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 subprobleme de rezolvat;

2. Etapa de **reducere** – primește ca date de intrare ieșirea de la etapa de mapare și combină perechile cheie-valoare astfel încât câmpul cheie va fi cuvântul, iar câmpul valoare va fi frecvența de apariție a cuvântului în text. Așadar, rezultă un set de date mai mic.
    - nodul cu rol de coordonator (sau un set de noduri cu rol de worker „desemnat” de coordonator) colectează soluțiile subproblemelor și le combină pentru a obține rezultatul final al procesării dorite.

5. Prezentarea soluției

Tema de casă constă în implementarea unei soluții MPI de tip MapReduce pentru problema construirii unui index invers pentru o colecție de documente text. Aplicația de test va primi ca parametrii de intrare numele unui director ce conține fișiere text (cu extensia ".txt") și un nume de director pentru stocarea datelor de ieșire și va genera pe post de răspuns un set de fișiere text ce conțin indexul invers corespunzător colecției de documente de intrare.

La rularea aplicației, un proces prestabilit, creat de MPI, își asumă rol de coordonator (se considera nodul master cel de rang 0), iar celelalte sunt workeri.

Inițial se verifică dacă există deja directorul care conține datele de ieșire. Dacă acesta există, va fi șters și creat din nou. La fel se procedează și cu directorul care conține datele temporare. Apoi procesul master, dacă se află în faza de mapare, atribuie fiecărui worker liber un fișier din directorul care conține datele de intrare, iar dacă se află în faza de reducere îi atribuie un fișier din directorul *temp*.

Directorul *temp* conține la rândul său un set de directoare ale căror nume este dat de cuvântul extras din folderul cu datele de intrare și conține un set de fișiere text (cu extensia “.txt”) ale căror nume este dat de numele fișierului text în care se găsește cuvântul și care conțin numărul de apariții al cuvântului în fișierul respectiv.

Fiecare worker va avea fișierul propriu (care va fi de forma worker_(rankul procesului)) care va conține cuvântul, numele fișierului în care se găsește și numărul de apariții ale acestuia în fișierul respectiv.

În final conținutul fișierelor generate de workeri for fi concatenate în fișierul *output.txt* astfel încât să obținem: $<term_k,\{docID_{k1} : count_{k1} ,docID_{k2} :count_{k2} ,\ldots, docID_{km}\ count_{km}\}>$

6. Implementare și pseudocod

În continuare este prezentat pseudocodul principalelor funcții.

```
function mapStoreResultPhase(file, frequency, s)
      if fișierul care conține datele de intrare poate fi
         deschis then
            while exista cuvinte in fisier do
                  se transforma toate cuvintele în cuvinte cu
                litere mici
                  if reprezinta o valoare alfanumerica then
                        ++frequency[s]
                  end if
            end while
      end if
      for it=frequency.begin() to frequency.end() do
            wordFolder = tempFolder + "/" + it->first
            se creeaza folderul cu numele wordFolder
            în folderul care are numele cuvântului se
            creeaza cate un fișier text care
            are numele fisierului din directorul cu datele
            de intrare și conține numărul
            de apariții ale cuvantului
      end for
end function
```

Funcția calculează numărul de apariții ale unui cuvânt din fișierele din directorul care conține datele de intrare. Astfel, dacă sunt întalnite valori alfanumerice salvăm într-un map care care are drept cheie cuvantul iar ca valoare numărul de apariții ale acestuia în fișier. Se creează un folder cu numele cuvantului care conține câte un fișier text care are numele fișierului din directorul cu datele de intrare și în care se găsește numărul de apariții ale cuvântului.

```
function masterPhase(n, taskList)
      switch do
            case Map:
                  taskList preia conținutul directorului care conține
                datele de intrare
                  break
            case Reduce:
                  taskList preia conținutul directorului care conține
                datele temporare
                  break
      end switch
      for i=0 to n do
            workerState[i]= starea Free
      end for
      while currentTask < taskList.size() sau workerul este in
                                          starea Free do
            for i=1  to n do
                  if workerState[i] == starea Free && currentTask
                              < taskList.size() then
                        se trimit taskurile catre workeri
                        workerState[i]= starea Working
                        currentTask = currentTask+1
                  end if
                  if workerul nu este in starea Free then
                        se testează dacă requestul a fost finalizat
                        if a fost finalizat then
                              if workerii și-au terminat execuția then
                                    workerState[i] = starea Free
                              end if
                        end if
                  end if
            end for
end function
```

În această funcție procesul master atribuie fiecărui worker care nu se află în starea Free câte un task.

```
function reducePhase (folder, rank, output, v)
      outFileName = outputFolder + "/"  + "worker_" + to_string(rank) +
                    ".txt"
    se deschide fișierul pentru scriere și adăugare
    se citește conținutul fiecărui director din folderul temp
    rezultatul va fi de forma: {cuvant : fisiere:[ nume_fisier,
    numar_aparitii] }
    for file in v do
        se citesc valorile din fișierele din folderele din directorul temp
        rezultatele din buffer sunt scrise în fișierul output
    end for
end function
```

Funcția preia fișierele din directorul *temp* și scrie în fișierele *worker* numele fișierelor și numărul de apariții ale cuvântului.

7. Bibliografie

[1] http://mike.tuiasi.ro/labsd12.pdf
[2] https://en.wikipedia.org/wiki/MapReduce
[3] http://www.csce.uark.edu/~mqhuang/courses/5013/s2018/lecture/5_intro_to_mapreduce.pdf
[4] https://www.ibm.com/topics/mapreduce
[5] https://www.projectpro.io/hadoop-tutorial/hadoop-mapreduce-tutorial-
[6] http://www.cas.mcmaster.ca/~nedialk/COURSES/4f03/Lectures/nonblocking.pdf