UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA:Calculatoare

Tema de casă

MapReduce

Ciubuc Remus-Mihail
Grupa 1407 A

Cuprins

Enunțul temei de casă:

În cadrul oricărui sistem de regăsire a informațiilor, colecția de date țintă este re-organizată (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 urmatorii:

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 lista de forma < docIDx,{term1 : count1,term2 : count2,...,termn : countn} > (**index direct** – countk înseamna numărul de apariții al termenului k);
2. fiecare lista obținută 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 obținem: < termk ,{docIDx : countk} >;
3. perechile obținute în pasul anterior sunt sortate după termk (cheie primară), docIDx (cheie secundară);
4. 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 implemenatrea unei soluţii MPI de tip **MapReduce** pentru problema construirii unui index invers pentru o colecţie de documente text.

Aplicaţia de test va primi ca **parametrii 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.

Noțiuni introductive:

MapReduce este o paradigmă de programare aplicată la scară largă, pentru procesarea și generarea unor seturi mari de date, folosind un algoritm paralel și distribuit, pe un cluster Hadoop.

În general, se consideră că acest model implică existenţa unui nod de procesare cu rol de coordonator (sau master sau iniţiator) şi mai multe noduri de procesare cu rol de worker. Ideea principală din spatele MapReduce este maparea setului de date într-o colecție de perechi <cheie,valoare> și apoi reducerea perechilor cu aceeași cheie.

Etapele paradigmei MapReduce:

Termenul MapReduce se referă la două sarcini separate și distincte pe care le efectuează programele Hadoop.

Un program MapReduce este compus dintr-o procedură *Map()* care efectuează filtrarea și sortarea (cum e sortarea elevilor după prenume în cozi, o coada pentru fiecare) si o metodă *Reduce()* care efectuează însumarea rezultatelor.

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).
    - nodul cu rol de coordonator împarte problema „originală” în sub probleme şi le distribuie către workeri pentru procesare.
    - 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 ; rezultă, în acest caz 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 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.
    - nodul cu rol de coordonator (sau un set de noduri cu rol de woker „desemnat” de coordonator) colectează soluţiile sub-problemelor şi le combină pentru a obţine rezultatul final al procesării dorite.

Sistemul ***MapReduce*** administrează serverele distribuite, rulând diferite sarcini în paralel, gestionează comunicațiile și transferul de date între diferitele componente ale sistemului, precum și asigurarea redundanței și a toleranței erorilor.

Schematic, cele două etape sunt prezentate în figura 1.[5]

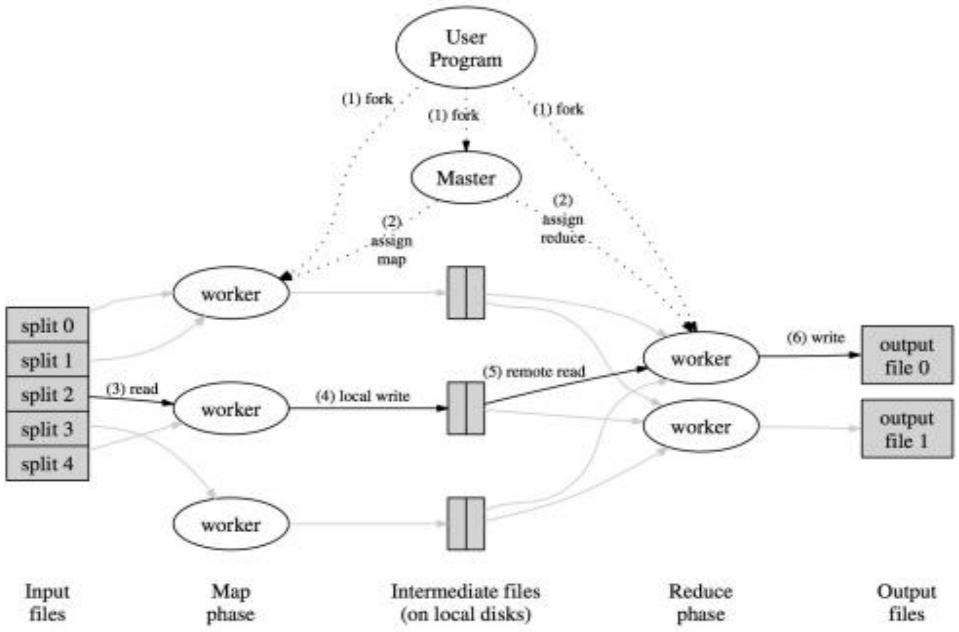

Figura 1: Paradigma MapReduce[5]

Michael Kleber (Google Inc.) descrie următoarele etapele implicate în paradigma MapReduce:

1. **<u>Preprocesare</u>:**

Datele sunt pregătite pentru etapa de mapare. Dimensiunea lor se alege astfel încât procesul sa fie eficient (vor fi distribuite în mod egal, pentru ca numărul acestora să nu fie prea mare). Diviziunile vor fi asignate in mod continuu. În cazul în care o mașină se va defecta, diviziunea acesteia se va asigna unei alteia. [1][2]

2. **<u>Mapare</u>:**
   În această etapă are loc stabilirea datelor de interes.
   Etapa de mapare constă în maparea diviziunilor datelor de intrare în perechi cheie valoare. O pereche cheie-valoare are forma „(termID, docID)”.[1][2]

3. **Amestecare și sortare:**

   Datele vor fi organizate astfel încât să fie optimizată etapa de reducere.[1][2]

4. **Reducere:**

   În această etapă, valorile obținute din etapa anterioară sunt combinate și returnate ca o singură ieșire. Reducerea poate returna zero sau mai multe perechi cheie-valoare.[1][2]

5. **Stocare rezultat**

Avantajele algoritmului MapReduce:

Cel mai mare avantaj al algoritmului MapReduce este reprezentat de uşurinţa scalării procesărilor de date peste mai multe noduri computaţionale. Odată ce scriem o aplicaţie în forma MapReduce scalarea acelei aplicaţii pentru a se putea executa peste sute, mii, sau chiar zeci de mii de maşini într-un cluster reprezintă doar o chestiune de modificarea unei configurări. De asemenea, MapReduce oferă flexibilitate în procesarea structurii sau a datelor nestructurate de către diverse companii care pot folosi datele și pot opera pe diferite tipuri de date.

MapReduce are o toleranţă mare la defecte. Ca un sistem de tip MapReduce să pice, trebuie să „pice” toţi workerii la o anumită fază a algoritmului.În momentul în care un worker pică, task-ul asignat acelui worker, este asignat altui worker.

Pseudocodul algoritmului MapReduce:[2]

```
function map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
    emit (w, 1)

function reduce(String word, Iterator partialCounts):
  // word: a word
  // partialCounts: a list of aggregated partial counts
  sum = 0
  for each pc in partialCounts:
    sum += pc
  emit (word, sum)
```

Exemple:

Putem observa cele două etape (Map și Reduce) precum și modul de funcționare al paradigmei MapReduce în cele două figuri de mai jos:

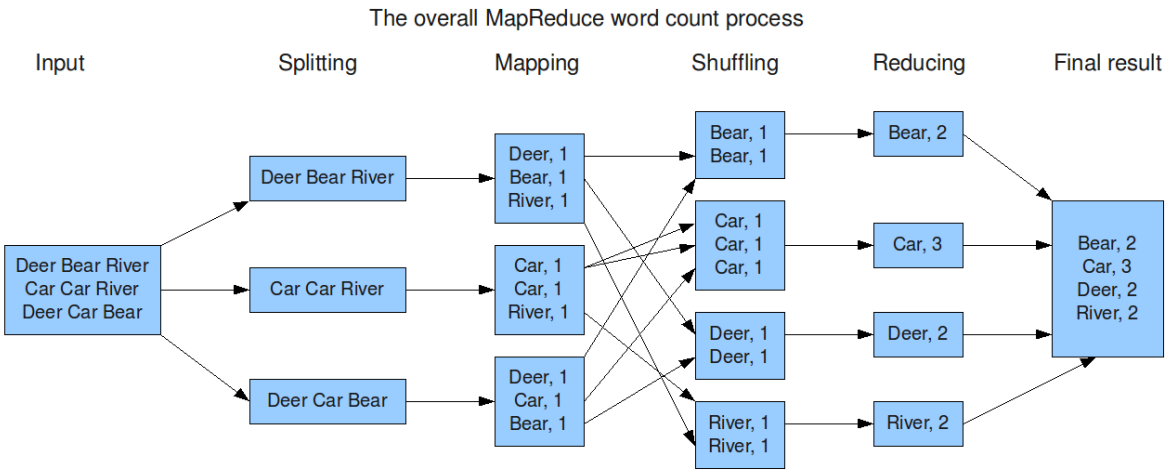

Figura 2[4]

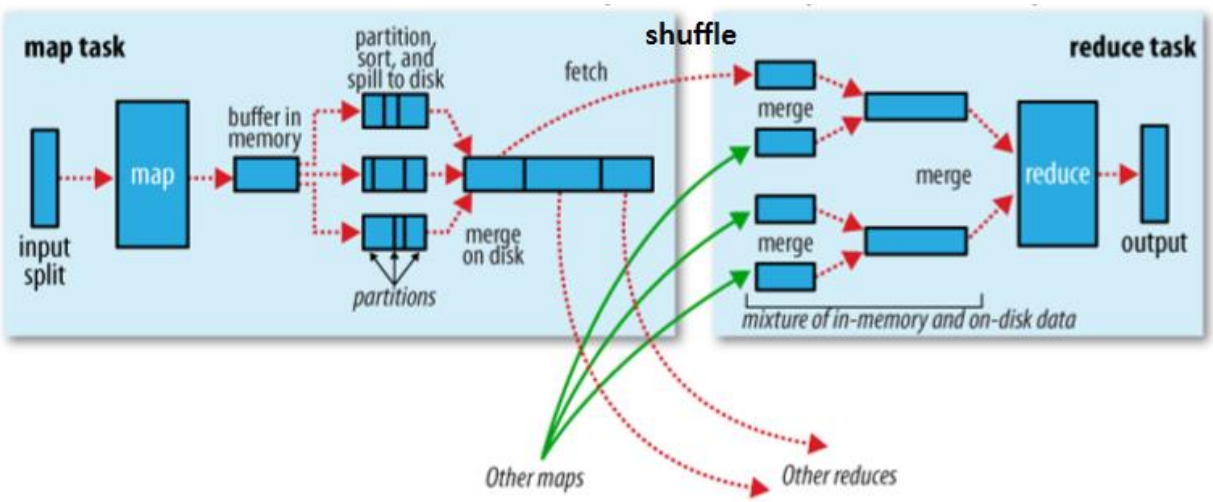

Figura 3[4]

Bibliografie:

[1].Michael Kleber. The MapReduce paradigm.
https://sites.google.com/site/mriap2008/lectures, January 2008.
[2].Wikipedia. MapReduce. http://en.wikipedia.org/wiki/MapReduce, November 2013.
[3].Laborator ALPD
[4]. https://www.todaysoftmag.com/article/1358/hadoop-mapreduce-deep-diving-and-tuning