Temă de casă
MapReduce

1 Prezentare generală și model

Termenul **MapReduce** referă, în prezent, un tipar de dezvoltare a aplicațiilor paralele/distribuite ce procesează volume mari de date [2, 6], [5, Cap. 4.4: Distributed indexing]. În general, se consideră că acest model implică existența unui nod de procesare cu rol de *coordonator* (sau *master*, sau *inițiator*) și mai multe noduri de procesare cu rol de *worker*. Modelul **MapReduce** implică două etape (după cum se poate intui din acronim) [6]:

o etapă de *mapare* nodul cu rol de *coordonator* împarte problema „originală” în sub-probleme și le distribuie către *workeri* pentru procesare;

trebuie reținut 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 și pot trimite aceste subdiviziuni către alți *worker*-i; rezultă în acest caz o arhitectură arborescentă;

divizarea caracteristică acestei etape nu trebuie să coreleze efectiv *dimensiunea datelor de intrare* cu *numărul de worker-i* din sistem; un *worker* poate primi mai multe sub-probleme de rezolvat;

o etapă de *reducere* nodurile cu rol de *worker* determină soluțiile sub-problemelor identificate în faza de mapare/selecție;

nodul cu rol de *coordonator* (sau un set de noduri cu rol de *woker* „desemnat” de *coordonator*) colectează soluțiile sub-problemelor și le combină pentru a obține rezultatul final al procesării dorite.

Schematic, aceste două etape sunt prezentate în Figura 1.

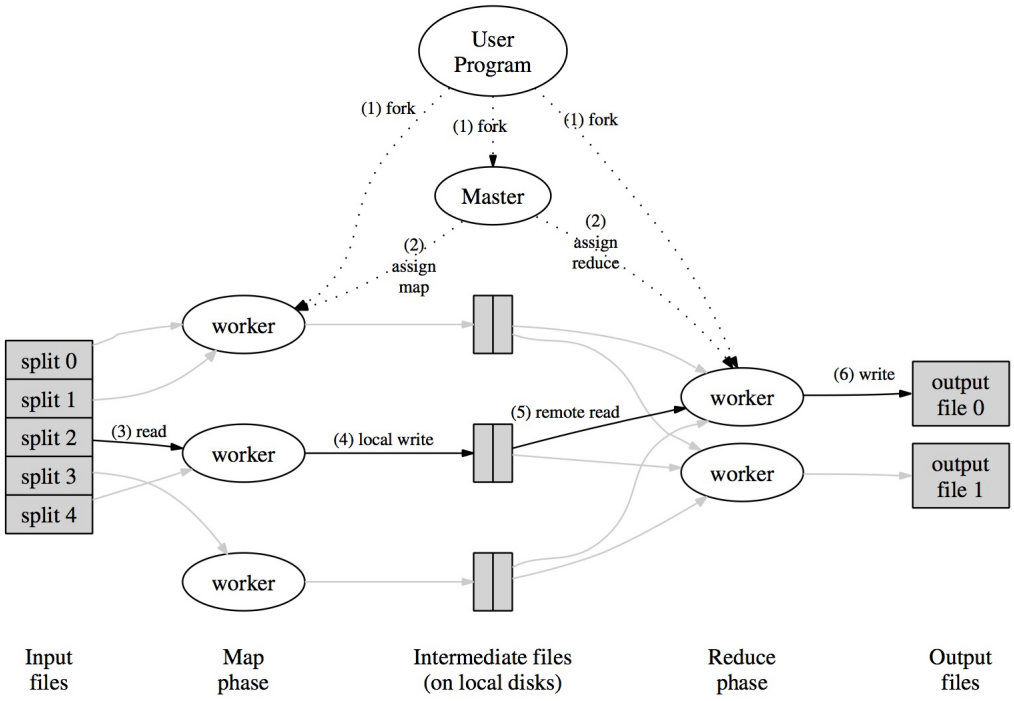

Figura 1: Paradigma MapReduce (preluare din [2])

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

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

Practic, Michael Kleber indică în [3] faptul că o problemă ce poate fi soluționată prin utilizarea modelului **MapReduce** va respecta întotdeauna cele 5 etape enumerate anterior. Singurele diferențe ce apar sunt legate de adaptarea etapelor de **mapare** și **reducere** astfel încât acestea să corespundă problemei de rezolvat.

Modele de aplicații ce pot fi soluționate prin intermediul acestei paradigme pot fi consultate în [1, 2, 3, 4, 6]. Pentru toate exemplele indicate în bibliografie se poate remarca faptul că datele de lucru trebuie să permită o organizare de tip dicționar – $< cheie, valoare >$, operațiile caracteristice paradigmei **MapReduce** fiind aplicate peste colecții masive de astfel de perechi.

2 Tema 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** (pentru detalii vezi [5, Cap. 4: Index construction]). 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 $< docID_x, \{term_1 : count_1, term_2 : count_2, \ldots, term_n : count_n\} >$ (**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 primaă), $docID_x$ (cheie secundară);
4. pentru fiecare $term_k$ se reunesc $< term_k, \{docID_x : count_k\} >$ astfel încât să obținem: $< term_k, \{docID_{k1} : count_{k1}, docID_{k2} : count_{k2}, \ldots, docID_{km} : count_{km}\} >$ (**indexul invers**).

Tema de casă constă în implemenatrea unei soluții MPI de tip **MapReduce** pentru problema construirii unui index invers pentru o colecție de documente text (*!!! soluția este schițată în [5, Cap. 4.4: Distributed indexing] !!!*). 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.

PREDAREA TEMEI DE CASĂ se va face în sesiune, înainte de susținerea examenului. Fiecare student va prezenta:

1. coduri sursă;
2. o prezentare scurtă a soluției, pseudocodul algoritmilor implementați, explicații scurte asupra implementării și funcționării soluției;
3. fiecare document de prezentare va include o secțiune de *Bibliografie*, în cadrul căreia se vor include principalele materiale studiate; **NU VOR FI ACCEPTATE PENTRU CORECTARE TEMELE CARE NU INCLUD SECȚIUNEA DE BIBLIOGRAFIE!!!**.

!!! Tema de casă ESTE CONDIȚIONATĂ DE NOTA MINIMĂ 5 (cinci) !!!

Bibliografie

[1] IBM Corp. What is MapReduce? `https://www.ibm.com/analytics/hadoop/mapreduce`.

[2] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA, 2004.

[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.

[5] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England, Online © 2009 Cambridge UP edition, 2009.

[6] Wikipedia. MapReduce. `http://en.wikipedia.org/wiki/MapReduce`, November 2013.