Evaluating the Efficiency of a Hybrid HPC Cluster by Using a Parallel Algorithm in Documents Clustering

Ioan UNGUREAN, Ştefan Gheorghe PENTIUC, Vasile Gheorghiță GĂITAN, Ovidiu GHERMAN

***Abstract*—The analysis of huge volumes of data has always required high computational ability. Implementing the clustering and classification algorithms upon an ordinary computer will not be efficient, since this shows limitations as regards the computational ability and available memory. Within this paper, an algorithm of clustering the huge volumes of data on a hybrid HPC cluster, is proposed. This cluster is based upon the RoadRunner architecture. It includes 48 QS22 blade servers, each of them having 2 PowerXCell 8i processors. The algorithm proposed within the present paper will carry out the clustering of a number of documents, by using the k-means algorithm, implemented upon the RoadRunner hybrid architecture.**

***Index Terms*—k-means, parallel algorithm, hybrid cluster architecture, QS22, MPI**

I. INTRODUCERE

Volumul de date pus la dispoziție de rețeaua Internet este foarte mare şi într-o creştere continuă. Internetul conține în acest moment cel puțin 22.25 miliarde de pagini. Algoritmii de recunoaştere a formelor şi data-mining joacă un rol foarte important în analiza informațiilor conținute în aceste documente. Deoarece setul de date de intrare pentru aceşti algoritmi sunt de nivelul gigabiților, algoritmii secvențiali nu pot realiza eficient gruparea documentelor folosind un singur procesor cu memorie limitată. Algoritmi paraleli [1] pot rezolva această problemă prin folosirea unui număr mare de procesoare, fiecare procesor având propria sa memorie.

Este foarte important de definit diferența dintre grupare şi clasificare [2]. În cazul clasificării este furnizat un set de date pre-clasificat pentru antrenare, şi problema constă în stabilirea claselor din care fac parte datele neetichetate. În cazul procesului de grupare, problema o reprezintă gruparea unui set de date neetichetate intr-un număr de clase.

Gruparea documentelor este un proces foarte important în data-mining. În general, procesul de grupare a documentelor respectă următorii paşi [3]:

Ioan UNGUREAN – Universitatea Ştefan Cel Mare Suceava, str.Universitatii nr.13, RO-720229 Suceava (e-mail: ioanu@eed.usv.ro)

Ştefan Gheorghe PENTIUC – Universitatea Ştefan Cel Mare Suceava, str.Universitatii nr.13, RO-720229 Suceava (e-mail: pentiuc@eed.usv.ro)

Vasile Gheorghiță GĂITAN– Universitatea Ştefan Cel Mare Suceava, str.Universitatii nr.13, RO-720229 Suceava (e-mail: gaitan@eed.usv.ro)

Ovidiu GHERMAN– Universitatea Ştefan Cel Mare Suceava, str.Universitatii nr.13, RO-720229 Suceava (e-mail: ovidiuh@stud.usv.ro)

- reprezentarea din punct de vedere matematic a documentelor;
- definirea unei funcții de similaritate;
- gruparea sau clustering-ul.

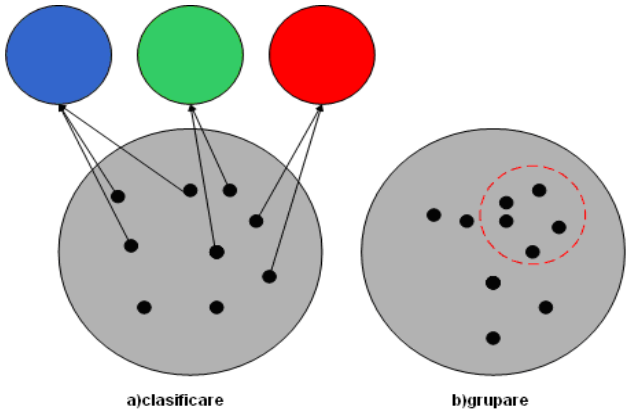


Fig. 1. Diferența dintre clasificare şi grupare.

Reprezentarea documentelor din punct de vedere matematic se referă la numărul de documente, numărul de clase şi la numărul de caracteristici care sunt specifice algoritmului de grupare. Extragerea acestor caracteristici reprezintă procesul de reprezentarea din punct de vedere matematic a setul de documente care vor fi grupate.

Algoritmul k-means [4] este unul din cei mai reprezentativi algoritmi de grupare a datelor şi poate fi folosit pentru gruparea documentelor în funcție de modul de reprezentare a acestora.

Pentru gruparea documentelor, în acest articol se propune un algoritm paralel k-means care rulează pe mai multe servere blade QS22. Aceste servere blade conțin două procesoare PowerXCell 8i, noua generație de procesoare bazate pe Cell Broadband Engine (Cell/B.E.) Architecture[5].

II. MODUL DE REPETENTARE A DOCUMENTELOR

Primul pas in realizarea procesului de grupare a documentelor constă în reprezentarea documentelor din setul de intrare sub o formă numerică, pentru a se putea aplica algoritmii de data clustering, în cazul de fată algoritmul k-means.

O metodă de reprezentare a documentelor o reprezintă

folosirea unei matrici de temeni şi documente. Acesta este o matrice matematică care conține frecvența de apariție a termenilor într-o colecție de documente. Pe lângă această matrice mai este nevoie şi de definirea unui dicționar de termeni sau a unui spațiu de termeni. Astfel fiecare linie din matrice corespunde unui document iar fiecare coloană corespunde unui termen din dicționar. Fiecare document din colecție reprezintă un punct sau un vector din cadrul spațiului de termeni. Următorul pas în realizarea procesului de grupare a documentelor o reprezintă definirea unei funcții de similitudine dintre 2 documente folosind reprezentarea numerică a documentelor.

III. SIMILITUDINEA DINTRE DOUĂ DOCUMENTE

Cea mai folosită metodă de măsurare a similarității în data clustering o reprezintă distanța Euclidiană. În cazul unui spațiu cu două dimensiuni aceasta reprezintă distanță dintre două puncte. În coordonate carteziene, distanța dintre 2 puncte $A(a_1, a_2, ...a_n)$ şi $B(b_1, b_2, ...b_n)$ este dată de formula :

$$d(A,B) = \sqrt{(a_1^2 + b_1^2) + (a_2^2 + b_2^2) + ... + (a_n^2 + b_n^2)} =$$

$$= \sqrt{\sum_i^n (a_i^2 + b_i^2)} \quad (1)$$

Dacă cele două puncte reprezintă doi vectori, se poate spune că cu cât unghiul dintre cei doi vectori este mai mic (cu cât cei doi vectori sunt mai apropiați), cosinusul acestui unghi se apropie de 1, cea ce înseamnă ca similaritatea dintre doi vectori nu depinde de mărimea lor. În continuare este dată formula pentru un spațiu cu două dimensiuni.

$$Sim(A,B) = \cos ine\theta = \frac{A \bullet B}{|A||B|} =$$

$$= \frac{a_1 * a_2 + b_1 * b_2}{(a_1^2 + b_1^2)^{1/2} (a_2^2 + b_2^2)^{1/2}} \quad (2)$$

În cazul clasificări documentelor este definit termenul weights, notat cu w. Acesta depinde de frecvența de apariție a termenilor în cadrul unui document, lungimea documentului şi/sau lungimea medie a documentelor din colecția de documente. Astfel similaritatea dintre două documente este dată de relația[7]:

$$Sim(A,B) = \frac{\sum_{i=1}^{1} w_{ai} * w_{bi}}{\sqrt{\sum_{i=1}^{1} w_{ai}^2} * \sqrt{\sum_{i=1}^{1} w_{bi}^2}} \quad (3)$$

Unde *w* poate fi definit ca:

- w= tf/tfmax (4)
- w = IDF = log(N/n) (5)
- w = tf*IDF=tf* log(N/n) (6)

tf - frecvența termenului, *tfmax* frecvența maxima a unui termen, *n*-numărul de termeni, *N* numărul de documente din colecția de documente. În cazul de față s-a folosit varianta *w= tf./tfmax*.

Această metodă poate fi folosită pentru clasificarea documentelor, prin măsurarea similarității documentelor față de un vector de referință. Această funcție de similaritate este denumită similaritate cosinus.

IV. ALGORITMUL DE GRUPARE

În cazul de față se intenționează folosirea algoritmului de grupare k-means pentru un set de date foarte mare. Acest algoritm trebuie să ruleze în paralel pe mai multe noduri QS22. În cazul de față se vor distribui, în mod egal, la noduri datele din setul de intrare. K-means este un algoritm foarte eficient pentru gruparea unui set de date în k clase. Dacă setul de date conține n puncte $X_1, X_2, ...X_n$ de dimensiune m, atunci procesul de clustering reprezintă procesul de grupare a n puncte de date în k clase, astfel încât să se maximizeze sau minimizeze funcția de similaritate (în funcție de tipul de similaritate care este folosit):

$$\sum_{j=1}^{k} \sum_{i}^{n} f(X_i, c_j), \quad (7)$$

unde c_j reprezintă centrul clasei C_j.

În cazul de față foloseşte similaritatea cosinus, deci cu cât funcția f se aproprie de 1 cu atât cele două documente sunt mai apropriate.

Algoritmul secvențial de grupare a *n* documente în *k* clase este:

```
Citeşte setul de date de intrare
Stabileşte aleatoriu k documente ca
centrele inițiale a claselor

repetă
    Pentru fiecare document caută cea mai
    apropriată clasă
    Pentru fiecare document incrementează
    cu 1 valoarea delta dacă şi-a schimbat
    clasa din care face parte.
    Recalculează centrele claselor în
    funcție de documentele care fac parte
    din fiecare clasă
Până când procentul documentelor care şi-
au schimbat clasa este mai mic decât o
valoare prestabilită

Salvează rezultatul procesului de grupare.
```

Pentru execuția în paralel a acestui algoritm pe mai multe noduri QSS22 trebuie luată în considerare arhitectura acestor noduri de calcul. Procesorul PowerXCell 8i de pe QSS22 are 9 nuclee eterogene conectate printr-o magistrală de 288GB/s, şi a fost proiectat pentru a avea o eficiență forte mare în cea ce priveşte consumul de energie. Procesorul PowerXCell 8i este un procesor multicore asimetric care este optimizat pentru procesarea paralelă şi aplicații de streaming. Procesorul PowerXCell 8i include un procesor Power Processor Element (PPE) şi opt procesoare SIMD optimizate pentru execuția în dublă precizie numite Synergistic Processor Elements (SPE).

De obicei PPE-ul este folosit pentru execuția sistemului de operare și coordonarea execuției pe cele opt procesoare SPE.

În Fig2. este prezentată arhitectura unui procesor PowerXCell 8i din punctul de vedere al programatorului.

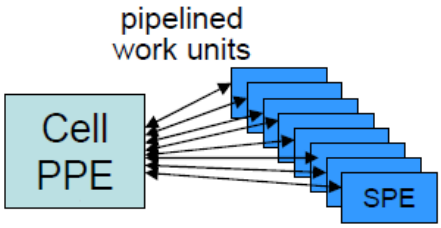


Fig. 2. Arhitectura unui procesor PowerXCell 8i.

Din punctul de vedere al programatorului, un nod QS22 are două procesoare PowerXCell 8i. El are acces doar la un PPE, de unde poate crea 16 theaduri care rulează în paralel pe cele 8 SPE de pe cele 2 procesoare PowerXCell 8i. Astfel aplicația care rulează pe mai multe noduri QS22 are două niveluri de paralelizare: primul nivel se realizară cu ajutorul librăriilor MPI pentru realiza execuția în paralel pe nodurile QS22 și al doilea nivel se realizează cu ajutorul librăriilor DaCS pentru a realiza execuția în paralele pe fiecare nod QS22 folosind cele 16 nuclee SPE. Procesoarele PPE sunt folosite doar pentru distribuirea și preluarea datelor de/la cele 16 procesoare SPE de pe fiecare nod.

Algoritmul k-means propus să lucreze pe *np* noduri QS22 este:

```
Citește setul de date de intrare (pe primul nod QS22)
Împarte setul de intrare la cele np noduri(pe primul nod QS22 la celelalte noduri)
Stabilește aleatoriu k documente ca centrele inițiale a claselor(pe primul nod QS22)
Repetă
    Transmite la cele np noduri centrele celor k clase(pe primul nod QS22 la celelalte noduri)
    Pentru fiecare document caută cel mai apropriat cluster (pe fiecare nod QS22 in paralel)
    Pentru fiecare document incrementez cu 1 valoarea delta dacă și-a schimbat clasa din care face parte(pe fiecare nod QS22 in paralel).
    Recalculează centrele claselor în funcție de documentele care fac parte din fiecare clasă (centralizează datele pe primul nod QS22)
Până când delta < n*0.1
Salvează rezultatul procesului de grupare.
```

Trebuie observat ca documentele sunt împărțite în mod egal la cele *np* noduri. De asemenea pe fiecare nod QS22 date sunt împărțite din nou la cele 16 procesoare SPE. Procesorul PPE este folosit doar pentru comunicația cu celelalte noduri QS22 și pentru transmiterea la procesoarele SPE a informațiilor necesare pentru a realiza execuția în paralel a algoritmului.

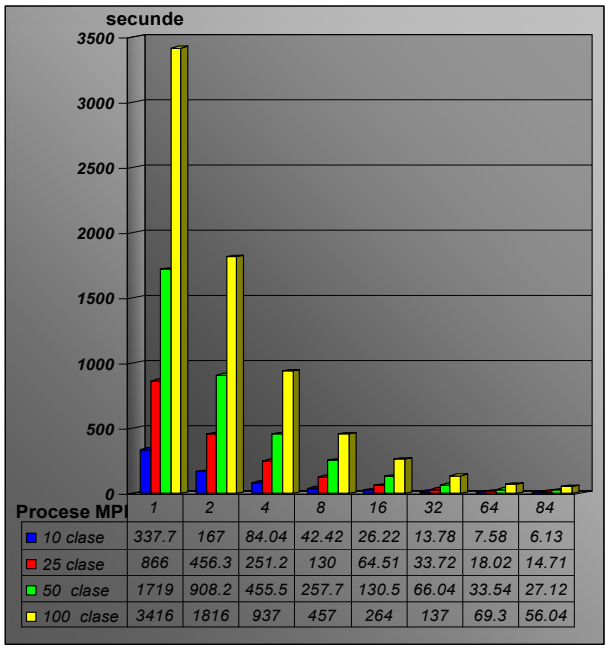
V. REZULTATE EXPERIMENTALE

Pentru testarea algoritmului k-means pe mai multe noduri QS22 s-au folosit seturile de date de intrare puse la dispoziție de UCI Machine Learning Repository[8]. S-au folosit două seturi de intrare: NIPS full papers care conține 1500 documente și mărimea dicționarului este de 12419 termeni și KOS blog entries care conține 3430 documente și mărimea dicționarului este de 6906 termeni.

Testele au fost efectuate pe un cluster cu 48 de noduri QS22. În cazul acestor teste s-a măsurat timpul efectiv de calcul, nu s-a luat in considerare timpul de citire a setului de date de intrare din fișier și distribuția acestui set de date la nodurile de calcul. În timpul testelor pe fiecare nod QS22 au rulat două procese MPI (câte un proces pe fiecare procesor PowerXCell 8i de pe QS22). De exemplu, dacă pentru un test se folosesc 2 noduri, atunci aplicația va executa 4 procese MPI. Setul de date de intrare este împărțit în mod egal la fiecare proces MPI. Pentru că fiecare proces MPI corespunde unui procesor PowerXCell 8i, în cadrul unui proces MPI se face distribuția setului de intrare și a calculelor la cele 8 procesoare SPE corespunzătoare procesorului PowerXCell atașat procesului MPI curent.

Pentru fiecare set de intrare s-a realizat gruparea documentelor în 10, 25, 50, 100 clase. Pentru fiecare grupare aplicația a fost executată folosind 1 proces MPI (s-a folosit un singur procesor de pe un nod QS22), 2 procese MPI (un nod QS22), 4, 8, 16, 32, 64, 84 procese MPI. Rezultatele testelor sunt date pentru 26 de iterații k-means [4].

În fig.3 și fig 4 sunt prezentate rezultatele experimentale pentru setul de intrare KOS.



Procese MPI	1	2	4	8	16	32	64	84
10 clase	337.7	167	84.04	42.42	26.22	13.78	7.58	6.13
25 clase	866	456.3	251.2	130	64.51	33.72	18.02	14.71
50 clase	1719	908.2	455.5	257.7	130.5	66.04	33.54	27.12
100 clase	3416	1816	937	457	264	137	69.3	56.04

Fig. 3. Rezultatele experimentale pentru setul de date de intrare KOS.

Din aceste rezultate se poate observa că prin creșterea numărului de procese MPI se realizează o scădere semnificativă a timpul de execuție. De asemenea timpul de

execuţie creşte dacă creşte şi numărul de clase în care sunt grupate documentele. În fig.4 este prezentată scăderea timpului de execuţie pentru gruparea documentelor din setul de date de intrare KOS în 25 de clase, prin creşterea numărului de procese MPI (noduri QS22) pe care se execută aplicaţia

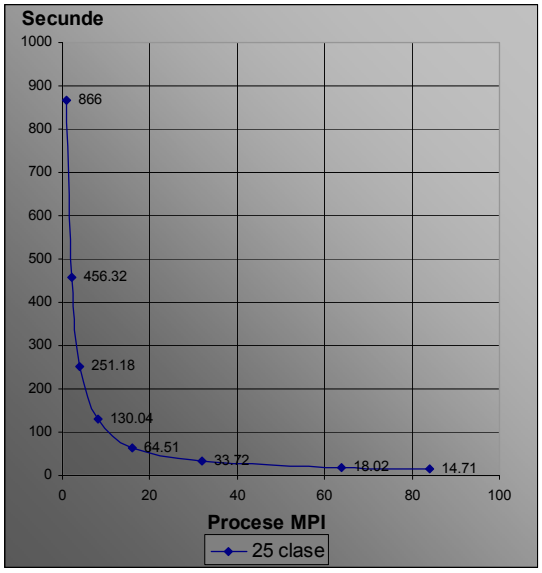
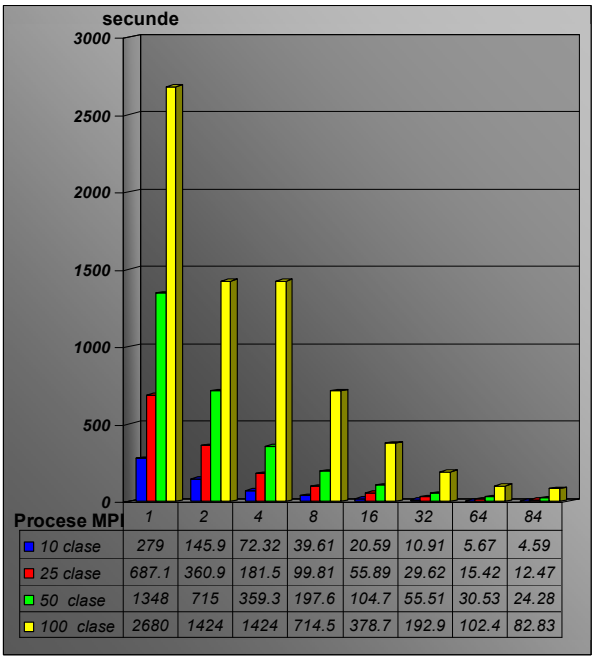


Fig. 4. Rezultatele experimentale pentru setul de date de intrare KOS şi 25 de clase.

În fig.5 sunt prezentate rezultatele pentru setul de intrare NIPS. Se poate observa aceeaşi scădere a timpului de execuţie prin creşterea numărului de procese MPI ca şi în cazul setului de date de intrare KOS.



Procese MPI	1	2	4	8	16	32	64	84
10 clase	279	145.9	72.32	39.61	20.59	10.91	5.67	4.59
25 clase	687.1	360.9	181.5	99.81	55.89	29.62	15.42	12.47
50 clase	1348	715	359.3	197.6	104.7	55.51	30.53	24.28
100 clase	2680	1424	1424	714.5	378.7	192.9	102.4	82.83

Fig. 5. Rezultatele experimentale pentru setul de date de intrare NIPS.

Scăderea timpului de execuţie se datorează modului în care a fost realizată aplicaţia. După citirea setului de date de intrare şi distribuirea lui la procesele MPI (acest timp nu este luat in considerare), la fiecare iteraţie k-means, fiecare proces caută cea mai apropriată clasă pentru fiecare document care ii aparţine. Această operaţie se realizează paralel (concurent) în toate procesele MPI. La sfârşitul iteraţiei se realizează recalcularea centrelor pentru fiecare clase, operaţie ce implică transmiterea de informaţii între procesele MPI. Pierderile datorate comunicaţiei sunt minime deoarece această operaţie implică transmiterea unui volum mic de informaţii între nodurile QS22, iar comunicaţia între noduri se realizează prin InfiniBand (20Gbps). Din acest motiv se realizează scăderea semnificativă a timpului de execuţie prin creşterea numărului de procese MPI.

VI. CONCLUZII

În cadrul acestui articol a fost prezentat un algoritm paralel pentru gruparea documentelor. Acest algoritm a fost implementat şi testat pe un cluster cu arhitectura hibrida RoudRunner. Din rezultatele testelor se poate observa ca prin dublarea numărului de noduri QS22 pe care este execută aplicaţia, timpul de execuţie scade cu un procent foarte apropriat de 50% (mai apar întârzieri datorate comunicaţiei dintre nodurile QS22).

MULŢUMIRI

Cluster-ul folosit pentru realizarea experimentelor a fost achiziţionat în cadrul proiectului ”GRID PENTRU DEZVOLTAREA APLICAŢIILOR DE RECUNOAŞTEREA FORMELOR ŞI INTELIGENŢĂ ARTIFICIALĂ DISTRIBUITĂ – GRIDNORD”, 80/13.09.2007, PN II, Programul Capacităţi.

REFERINŢE BIBLIOGRAFICE

[1] Zhonghui Feng, Bing Zhou, Junyi Shen, *A parallel hierarchical clustering algorithm for PCs cluster system,* Neurocomputing, Volume 70, Issues 4-6, January 2007, Pages 809-818

[2] Michael W. Berry, Malu Castellanos , *Survey of Text Mining II: Clustering, Classification, and Retrieva*l, Springer, 2008,ISBN 1848000456, 9781848000452

[3] Nicholas O. Andrews and Edward A. Fox, Recent Developments in Document Clustering, October 16, 2007

[4] Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y.; *An efficient k-means clustering algorithm: analysis and implementation,* Pattern Analysis and Machine Intelligence, IEEE Transactions on Volume 24, Issue 7, July 2002 Page(s):881 – 892

[5] IBM BladeCenter QS22, http://www-03.ibm.com/systems/bladecenter/hardware/servers/qs22/

[6] Roadrunner: Hardware and Software Overview, IBM RedBooks.

[7] E. Garcia, *Term Vector Theory and Keyword Weights,*2005), http://www.miislita.com/term-vector/term-vector-1.html

[8] Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.