UNIVERSITATEA TEHNICA “Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Sisteme Distribuite și Tehnologii Web
DISCIPLINA: FUNDAMENTELE PROCESĂRII DISTRIBUITE

Algoritmul Berkeley

SINCRONIZAREA CEASURILOR

Coordonator,
Ș.l.dr.ing. Adrian ALEXANDRESCU

Student,
Camelia-Maria MODIGA, 1A

Iași, 2022

Cuprins

1. Descrierea problemei

Sincronizarea ceasurilor într-un sistem distribuit este necesară pentru a ordona în timp evenimentele computaționale și de comunicare. Atunci când comunicarea se face prin transmitere de mesaje, unitățile de procesare trebuie să aducă la un numitor comun ceasurile lor, prin aplicarea unui mecanism de sincronizare bazat pe dialog.

Deoarece ceasurile fizice (hardware clock) nu pot fi controlate de către unitățile de procesare, presupunem ca fiecare unitate de procesare are acces la o componenta speciala de stare numita adjust (ajustare), pe care o poate modifica. Ceasul corectat al unui proces este o combinatie între valoarea ceasului fizic și valoarea de ajustare.

2. Idei de rezolvare

Algoritmul Berkeley este o metodă de sincronizare a ceasurilor în sistemele distribuite. Acest algoritm este folosit în cazul în care sistemele rețelei distribuite prezintă una din următoarele probleme: mașina nu are o sursă precisă pentru timp sau nu are un server UTC.

Algoritmul Berkeley consideră următoarea abordare: un nod individual este ales ca nod *master* dintr-o serie de noduri din rețea iar restul nodurilor se comportă ca *slave.* Fiecare proces slave va trimite către procesul *master* valoarea timpului pe care o cunoaște. Bazându-se pe răspunsuri, procesul *master* calculează un timp mediu prin intermediul căruia se încearcă diminuarea diferenței de timp. Face acest lucru prin trimiterea către celelalte noduri o valoare prin care le spune să avanseze sau încetinească ceasurile până când a fost realizată o anumită reducere a diferenței. Această valoarea poate fi pozitivă sau negativă. Dacă masterul eșuează atunci este ales un nou proces *master* care să preia rolul predecesorului său.

3. Algoritm

3.1. Descriere

În cadrul acestui algoritm, un nod individual este ales ca nod master dintr-o serie de noduri din rețea. Acest nod este nodul principal din rețea și se comportă ca *master* iar restul nodurilor se comportă ca *slave.* Nodul master este ales folosind un algoritm de alegere a liderului.

În mod periodic, nodul *master* întreabă nodurile *slave* și preia timpul acestora folosind algoritmul lui Cristian.

Nodurile *slave* trimit ca răspuns timpul dat de ceasul sistemului lor. Nodul *master* calculează media diferenței de timp dintre toți timpii ceasurilor primiți și ceasul nodului master. Aceasta diferență este adaugată la timpul curent al nodului *master*, iar noul timp este trimis la nodurile *slave*.

3.2. Pseudocod

```
1: function BERKELEY
2:     Read values from input file
3:     Assign the coordinator process rank from the first line of the file
4:     The leader polls the slaves who reply with their time
5:     Send HC (current hardware clock value) to all processing units
6:     for i ← 1 to n − 1 do
7:         wait a message T
8:         time_difference ← time_at_master_node − time_at_slave_node
9:         average_time_difference ← sum(all_time_differences) / number_of_slaves
10:        synchronized_time ← current_master_time + average_time_difference
11:        Send synchronized_time to all processing units
```

3.3. Exemplu de aplicare

Considerând următoarele noduri dintr-un sistem distribuit și timpul dat de ceasul lor:

N1 → 3:00

N2 → 3:10

N3 → 2:50

N4 → 3:20

Inițial este ales liderul sistemului ca fiind nodul N1 iar acesta ia rolul de master.

Procesul *master* cere nodurilor *slave* timpul ceasurilor.

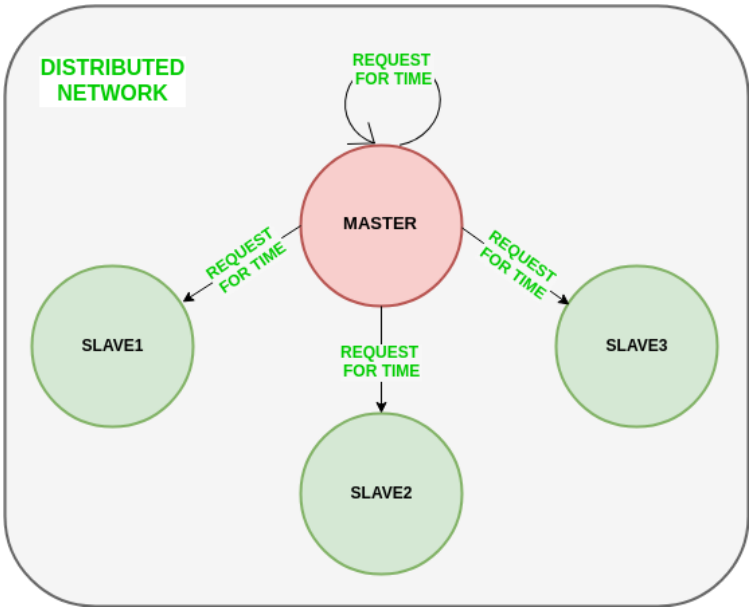

Figura 1. Cererea timpului de către procesul master

Procesele *slave* trimit înapoi către *master* timpul ceasurilor

N1 → timp: 3:00
N2 → timp: 3:10
N3 → timp: 2:50
N4 → timp: 3:20

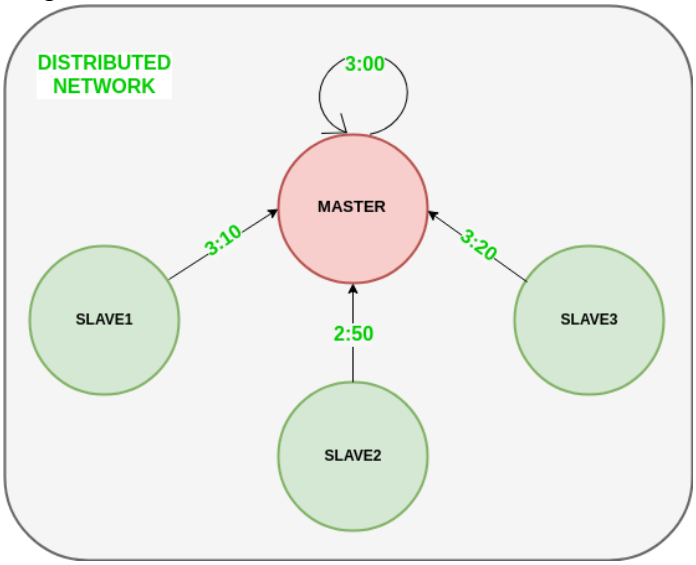

Figura 2. Trimiterea timpului către master de către procesele slave

Nodul *master* calculează media diferenței de timp dintre toți timpii ceasurilor primiți și trimite înapoi către nodurile *slave* corecția.

N1 → time: 3:00 (+5)
N2 → time: 3:05 (-5)
N3 → time: 2:50 (+15)
N4 → time: 3:20 (-15)

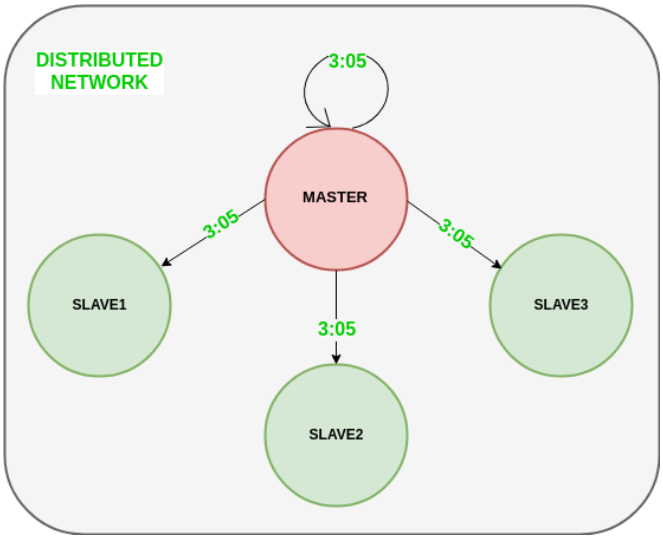

Figura 3. Trimiterea corecției de către nodul master către nodurile slave

3.4. Corectitudine

Conform exemplului dat anterior, algoritmul este corect pentru rezolvarea problemei de sincronizare a ceasurilor.

Să presupunem că există două constante pozitive *d* și *u*, *d*>*u*, astfel încât orice mesaj are o întârziere a cărei valoare se încadrează în intervalul *[d-u, d]*. Astfel, dacă *pi* transmite mesajul *m* către *pj* la momentul de timp real *t*, atunci evenimentul de comunicare prin care mesajul *m* este lui *pj* împreună cu evenimentul computațional prin care mesajul este prelucrat *pj* se va termina: cel mai devreme la momentul *t*+*d*-*u* și cel mai târziu la momentul *t*+*d*.

La terminarea execuției toate nodurile vor avea aceeași oră.

3.5. Analiza complexității

Considerând n numărul de noduri slave care urmează sa fie parcurse la trimiterea request-ului pentru a primi valoarea timpului acestora s-a ajuns la concluzia ca algoritmul are complexitatea O(n).

3.6. Implementare

Pentru implementarea algoritmului a fost creat fișierul *berkeley.txt* unde sunt stocate datele de intrare. Prima linie din fișier reprezintă *rank-ul* procesului care a fost ales drept *master* iar următoarele linii reprezintă valorile pentru ceasurile unităților de procesare. Numele fișierului trebuie dat ca argument al liniei de comandă la rularea programului.

Inițial se verifică dacă numele fișierului este valid și dacă datele stocate în fișier respectă formatul pentru oră. Apoi procesul *master* trimite timpul său către celelalte noduri iar nodurile trimit către procesul *master* diferența de timp dintre timpul procesului *master* și timpul acestora. Se calculează diferența medie de timp iar procesul coordonator trimite valoarea cu care urmează să fie corectat fiecare ceas.

Ultimul pas este reprezentat de sincronizarea ceasurilor tuturor proceselor din sistem și afișarea valorii acestuia.

3.7. Testare

Testarea aplicației a fost realizată folosind principiile testării de tip *black box*. Astfel, a fost testată funcționalitatea algoritmului fără a fi necesară cunoașterea detaliilor de implementare. Acest tip de testare se bazează în totalitate pe cerințele și specificațiile software ale aplicației, concentrându-se doar pe datele de intrare introduse de utilizator și prelucrate de aplicație, respectiv pe cele de ieșire.

În scopul testării algoritmului a fost creat un fișier cu extensia *txt* unde au fost salvate date referitoare la rank-ul nodului master precum și timpurile tuturor ceasurilor din cadrul sistemului. În timpul execuției programului sunt citite datele din interiorul fișierului și se calculează timpul ajustat pentru fiecare nod pe baza datelor regăsite în fișier.

Testarea a fost realizată prin introducerea diferitor valori în fișierul *berkeley.txt* și observarea rezultatelor obținute.

Pentru lansarea în execuție a aplicației trebuie rulate următoarele comenzi:

mpic++ -o berkeley berkeley.c -lm
mpirun -np 4 berkeley berkeley.txt

În urma lansării în execuție a programului cu datele de test prezentate la punctul 3.3 se obține următorul rezultat:

```
cami@cami:~/Desktop/fpd$ mpirun -np 4 berkeley berkeley.txt
I am process with rank 0 acting as the coordinator process
Coordinator process is sending time 3:0
Process 0 has received time differential value of 0
Process 1 has received time 3:0
Process 1 is sending time differential value of 10 to process 0
Process 0 has received time differential value of 10
Process 3 has received time 3:0
Process 3 is sending time differential value of 20 to process 0
Process 0 has received time differential value of 20
Process 2 has received time 3:0
Process 2 is sending time differential value of -10 to process 0
Process 0 has received time differential value of -10
Time differential average_diff is 5.000000
Coordinator process is sending the clock adjustment value of 5.000000 to process 0
Coordinator process is sending the clock adjustment value of -5.000000 to process 1
Coordinator process is sending the clock adjustment value of 15.000000 to process 2
Coordinator process is sending the clock adjustment value of -15.000000 to process 3
Adjusted local time at process 0 is 3:5
Process 2 has received the clock adjustment value of 15.000000
Adjusted local time at process 2 is 3:5
Process 1 has received the clock adjustment value of -5.000000
Adjusted local time at process 1 is 3:5
Process 3 has received the clock adjustment value of -15.000000
Adjusted local time at process 3 is 3:5
```

Figura 4. Exemplu de lansare în execuție a programului

4. Concluzii

În cadrul acestui proiect s-a urmărit implementarea algoritmului Berkeley cu scopul de a sincroniza ceasurile din cadrul unui sistem.

Obiectivele propuse în temă au fost atinse. A fost implementată o modalitate de sincronizare a ceasurilor de către un nod master care decide modul în care se modifică ceasul unui nod slave utilizând MPI.

Cu toate că algoritmul Berkeley prezintă anumite limitări se poate considera o alegere optimă pentru pentru rezolvarea problemei de sincronizare a ceasurilor.