UNIVERSITATEA DIN CRAIOVA

**FACULTATEA DE ȘTIINȚE**

**DEPARTAMENTUL DE INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

**Coordonator științific**

Dr. Popârlan Cristina

**Absolvent**

Denisov Vladislav

**Introducere**

În era modernă a tehnologiei web, dezvoltarea continuă și inovația sunt strâns legate de implementarea și rafinarea algoritmilor distribuiți. Aceștia constituie inima și creierul sistemelor distribuite, având un rol vital în coordonarea activităților și gestionarea resurselor în mediile de calcul distribuit.

Mediul de calcul distribuit reprezintă acum baza pentru o gamă largă de tehnologii web, cum ar fi cloud computing-ul, rețelele sociale, sistemele de calcul paralel și blockchain-ul. În acest univers interconectat, o colecție finită de entități computaționale autonome, denumite noduri, colaborează pentru a rezolva probleme complexe și a îndeplini sarcini specifice. Algoritmii distribuiți sunt esențiali pentru a asigura funcționarea eficientă și fiabilă a acestor sisteme. Ei oferă regulile și procedurile necesare pentru coordonarea activităților între noduri, gestionarea resurselor și comunicarea pentru a rezolva problemele întâlnite.

În esență, algoritmii distribuiți trebuie să abordeze numeroase provocări specifice, cum ar fi cunoașterea limitată a rețelei, memorie și comunicare limitate și probleme de sincronizare. Ei trebuie să fie, de asemenea, rezilienți la defecțiuni și atacuri, să se adapteze la schimbările dinamice ale rețelei și să funcționeze eficient în medii eterogene.

De la sistemele de fișiere distribuite la rețelele de livrare a conținutului și blockchain-ul, algoritmii distribuiți sunt fundamentali pentru gestionarea eficientă a resurselor, asigurând consistența datelor și coordonarea proceselor în medii complexe și dinamice. Astfel, dezvoltarea tehnologiilor web este strâns legată de progresul și inovația în domeniul algoritmilor distribuiți.

**Capitolul 1 Dezvoltarea aplicațiilor web folosind algoritmi distribuiți**

Principiile fundamentale ale algoritmilor distribuiți sunt esențiale pentru dezvoltarea aplicațiilor web care funcționează în medii distribuite. Aceste principii includ menținerea coerenței datelor pentru a asigura consistența informațiilor, sincronizarea între noduri pentru a evita conflictele și a gestiona accesul concurent la resurse, toleranța la defecțiuni pentru a asigura funcționarea continuă a aplicației în ciuda eșecurilor, și eficiența și scalabilitatea pentru a gestiona volumul crescut de date și trafic. Prin aplicarea acestor principii, dezvoltatorii pot crea aplicații web robuste și reziliente care oferă o experiență fiabilă și scalabilă utilizatorilor lor.

**1.1 Avantajele și dezavantajele sistemelor distribuite**

**Avantaje**:

* **Scalabilitate:** Algoritmii distribuiți permit sistemelor să crească în capacitate prin adăugarea de noi noduri în rețea. Această abordare, cunoscută și sub numele de scalabilitate orizontală, permite distribuirea sarcinilor pe mai multe servere sau noduri, în funcție de nevoile crescânde ale aplicațiilor. De exemplu, în cazul unei aplicații web care primește un număr mare de cereri, scalabilitatea orizontală permite adăugarea de noi servere pentru a gestiona volumul crescut de trafic, fără a afecta performanța globală a sistemului.
* **Redundanță și toleranță la defecțiuni:** Un alt beneficiu major al algoritmilor distribuiți este capacitatea de a asigura redundanță și toleranță la defecțiuni. Distribuirea datelor și sarcinilor pe mai multe noduri face ca sistemul să fie mai rezistent la eșecuri hardware sau software. Dacă un nod întâmpină o defecțiune, sarcinile pot fi preluate de alte noduri fără a afecta în mod critic disponibilitatea serviciului. Acest aspect este crucial în aplicațiile critice unde indisponibilitatea pentru o perioadă lungă poate avea consecințe grave.
* **Performanță îmbunătățită:** Algoritmii distribuiți permit distribuirea sarcinilor și proceselor pe mai multe procesoare sau noduri, ceea ce poate duce la o îmbunătățire semnificativă a performanței sistemului. Prin paralelizarea operațiunilor, sistemul poate efectua mai multe calcule sau operațiuni simultane, reducând timpul de răspuns și creștând eficiența generală. De exemplu, în calculul paralel distribuit, problemele complexe pot fi împărțite în subprobleme mai mici, care sunt apoi rezolvate independent de diferite noduri, accelerând astfel procesul de calcul.
* **Flexibilitate geografică:** Sistemele distribuite permit accesul și procesarea datelor de la distanță geografică. Aceasta facilitează colaborarea între utilizatori și echipe din locații diferite, permitându-le să lucreze împreună în timp real pe aceleași seturi de date și resurse. De exemplu, într-o organizație globală, utilizarea unui sistem distribuit permite echipei dintr-o țară să acceseze și să modifice datele stocate pe servere din alte regiuni, facilitând astfel eficiența operațională și gestionarea informațiilor.
* **Eficiență în utilizarea resurselor:** Utilizarea distribuită a resurselor permite o gestionare mai eficientă a capacității de calcul și stocare disponibile. Prin distribuirea sarcinilor pe mai multe noduri, se poate reduce încărcarea excesivă a unui singur server sau sistem, optimizând utilizarea resurselor hardware. Acest lucru poate contribui la economii semnificative de costuri și la o mai bună utilizare a investițiilor în infrastructură IT.

**Dezavantaje**:

* **“Procesoarele pot fi separate si comunicarea dintre ele este mult mai problematica”**[[1]](#footnote-21102): Aceasta indică o dezavantajare, deoarece separarea fizică a procesoarelor poate duce la dificultăți în comunicarea și coordonarea între ele. În sistemele distribuite, gestionarea comunicațiilor între noduri poate fi mai complexă și mai predispusă la probleme.
* **“Itarzierile de comunicare nu pot fi anticipate”[[2]](#footnote-20841)** : Întârzierile imprevizibile în transmiterea datelor între procesoare pot afecta performanța sistemului distribuit și pot complica sincronizarea și coordonarea activităților.
* **“Legaturile de comunicatie pot sa nu fie de incredere”** : Fiabilitatea comunicațiilor este crucială în sistemele distribuite pentru a asigura trimiterea corectă și sigură a datelor între procesoare.
* **Complexitatea implementării algoritmilor distribuiți**: Adaptarea algoritmilor pentru a funcționa eficient într-un mediu distribuit poate fi dificilă din cauza necesității de a lua în considerare distribuția datelor și sincronizarea operațiunilor între multiple noduri.
* **Problemele de coerență și consistență a datelor**: În sistemele distribuite, asigurarea coerenței și consistenței datelor poate fi o provocare semnificativă, din cauza nevoii de a gestiona actualizările simultane ale datelor pe mai multe noduri.
* **Consumul de resurse suplimentar**: Implementarea și operarea algoritmilor distribuiți pot necesita o cantitate mai mare de resurse de calcul și de rețea decât sistemele centralizate, din cauza necesității de a gestiona comunicarea și coordonarea între nodurile distribuite.

**1.2 Arhitectura aplicațiilor web distribuite**

În mediul actual al tehnologiei digitale, aplicațiile web se confruntă cu cerințe tot mai complexe, inclusiv gestionarea volumelor mari de date și menținerea performanței în fața unui trafic sporit și a diversității utilizatorilor. Arhitectura aplicațiilor web distribuite oferă o soluție esențială pentru a gestiona aceste provocări, permițând dezvoltarea și implementarea aplicațiilor scalabile și eficiente.

Un aspect fundamental al acestei arhitecturi este reprezentat de algoritmii distribuiți. Aceștia permit distribuirea eficientă a sarcinilor și a datelor pe mai multe noduri sau servere interconectate, facilitând o comunicație fluentă și o gestionare eficientă a resurselor. Algoritmii distribuiți sunt esențiali pentru a asigura performanța și fiabilitatea sistemului, asigurând în același timp o experiență consistentă utilizatorilor indiferent de locația lor geografică sau de volumul de trafic.

În cadrul acestei secțiuni, vom explora rolul și importanța algoritmilor distribuiți în cadrul arhitecturii aplicațiilor web distribuite. Vom analiza modul în care aceștia contribuie la:

* Eficiența operațională și gestionarea resurselor într-un mediu distribuit.
* Asigurarea comunicării și sincronizării corecte a datelor între diverse componente ale sistemului.
* Implementarea unor practici avansate pentru securitatea datelor și protejarea accesului la resursele aplicației.

De asemenea, vom examina importanța entităților în cadrul acestei arhitecturi, evidențiind modul în care acestea sunt gestionate și integrate în sistemele distribuite pentru a asigura coerența și fiabilitatea operațională.

Această secțiune oferă o perspectivă cuprinzătoare asupra modului în care algoritmii distribuiți contribuie la dezvoltarea și operaționalizarea eficientă a aplicațiilor web într-un mediu digital dinamic și global.

**1.2.1 Entități**

În programarea web, în special în contextul algoritmilor distribuiți, o "entitate" reprezintă o unitate distinctă de date și comportament identificabilă în mod unic, având atribute și metode specifice. Aceste entități sunt esențiale pentru distribuirea datelor, sincronizarea și menținerea consistenței între noduri, rezolvarea conflictelor și asigurarea performanței și disponibilității în sistemele distribuite. De exemplu, în baze de date distribuite precum Cassandra sau MongoDB, entitățile sunt stocate ca documente sau rânduri distribuite între noduri, iar în arhitecturi de microservicii, fiecare serviciu gestionează un set specific de entități, cum ar fi utilizatorii. În sisteme peer-to-peer, fișierele sau părțile de fișiere sunt considerate entități distribuite între noduri, evidențiind rolul crucial al entităților în organizarea și gestionarea datelor și logicii în aplicațiile distribuite.

Fiecare entitate x ∈ E are o memorie locală (privată și nepartajată) Mx​. Capabilitățile entității includ accesul (stocarea și recuperarea) la memoria locală, procesarea locală și comunicarea (pregătirea, transmiterea și recepția mesajelor). Memoria locală include registre definite ale căror valori sunt inițial stabilite; printre acestea se numără registrul de stare (denumit status(x) ) și registrul valorii de intrare (denumit value(x) ). Registrul status(x) poate avea valori dintr-un set finit de stări de sistem S precum „Idle”, „Processing”, „Waiting” etc.

În plus, fiecare entitate x ∈ E are un ceas de alarmă local cX​ pe care îl poate seta și reseta (opri).

O entitate poate efectua patru tipuri de operațiuni:

* stocare și procesare locală
* transmiterea mesajelor
* setarea/resetarea ceasului de alarmă
* schimbarea valorii registrului de stare

**Evenimente Externe**

În programarea web și în contextul algoritmilor distribuiți, entitățile și evenimentele externe interacționează frecvent pentru a permite aplicațiilor să reacționeze la schimbări și să proceseze informații în timp real.

Evenimentele externe sunt acțiuni sau schimbări care au loc în afara unei entități, dar care o afectează sau necesită o reacție din partea acesteia. Exemple de evenimente externe includ:

* **Interacțiuni Utilizator**: Click-uri pe butoane, formulare trimise.
* **Schimbări de Stare în Sistem**: Mesaje primite de la alte servicii, modificări în baza de date.
* **Evenimente Temporale**: Alarme și cronometre care declanșează acțiuni la anumite momente.

**Interacțiunea dintre Entități și Evenimente Externe**

1. **Recepționarea Evenimentelor**:

* **Webhooks**: Entitățile pot recepționa evenimente externe prin webhook-uri. De exemplu, un serviciu de plăți poate trimite notificări despre tranzacții finalizate.
* **API-uri**: Entitățile pot primi solicitări prin API-uri REST sau GraphQL, declanșând acțiuni specifice pe baza acestor solicitări.
* **Sisteme de Mesagerie**: Utilizarea cozii de mesaje (cum ar fi RabbitMQ sau Kafka) pentru a primi și procesa evenimente externe.

1. **Procesarea Evenimentelor**:

* **Actualizarea Datelor**: Evenimentele externe pot determina modificări ale atributelor entităților. De exemplu, un eveniment de "comandă livrată" poate actualiza starea unei entități "Comandă" la "Livrată".
* **Declanșarea de Acțiuni**: Pe baza evenimentelor externe, entitățile pot executa acțiuni precum trimiterea de emailuri, actualizarea altor entități sau inițierea de procese complexe.

1. **Persistența Evenimentelor**:

* **Logare și Auditare**: Evenimentele externe sunt adesea logate pentru auditare și urmărire. Acest lucru asigură trasabilitatea și ajută la diagnosticarea problemelor.
* **Managementul Stării**: Evenimentele externe pot fi stocate pentru a gestiona starea entităților în mod consistent, mai ales în sisteme de tip Event Sourcing.

1. **Sincronizare și Consistență**:

* **Sincronizare în Timp Real**: Evenimentele externe ajută la menținerea sincronizării în timp real a entităților distribuite pe mai multe noduri.
* **Gestionarea Concurenței**: Mecanismele de gestionare a concurenței sunt esențiale pentru a preveni conflictele atunci când multiple evenimente externe afectează aceeași entitate simultan.

1. curs1 pag.1 [↑](#footnote-ref-21102)
2. curs1. Pag1 [↑](#footnote-ref-20841)