UNIVERSITATEA TEHNICA
DIN CLUJ-NAPOCA
CENTRUL UNIVERSITAR NORD DIN BAIA MARE
FACULTATEA DE INGINERIE

DEPARTAMENTUL CALCULATOARE

Calculator Distribuit

Autor: **Cătălin-Beniamin Danciu**

2026

Cuprins

Capitolul 1. Introducere

1.1. Context general

În contextul dezvoltării sistemelor distribuite și al aplicațiilor care necesită procesarea eficientă a datelor, apare necesitatea utilizării mai multor noduri de calcul care să lucreze în paralel. Sistemele distribuite permit împărțirea sarcinilor de procesare între mai multe componente independente, crescând performanța și scalabilitatea aplicațiilor.

Calculatorul distribuit reprezintă un exemplu didactic relevant pentru înțelegerea modului în care mai multe procese pot colabora pentru a rezolva o problemă comună. Prin distribuirea calculelor matematice către mai mulți workeri, aplicația demonstrează concepte precum paralelismul, comunicarea între procese și agregarea rezultatelor parțiale.

1.2. Scopul și motivația proiectului

Scopul principal al acestui proiect este realizarea unui calculator distribuit capabil să execute operații matematice complexe utilizând mai multe noduri de calcul (workeri). Aplicația urmărește aplicarea practică a conceptelor studiate în cadrul disciplinei Sisteme Distribuite, precum arhitectura client–server, distribuirea sarcinilor și coordonarea proceselor.

Motivația realizării proiectului constă în dorința de a înțelege diferența dintre un calculator clasic și unul distribuit, precum și avantajele oferite de execuția paralelă a calculelor asupra unor seturi mari de date.

1.2.1. Obiectivele proiectului

Obiectivele principale ale proiectului sunt:

- realizarea unei arhitecturi distribuite de tip coordinator–worker;
- implementarea execuției paralele a operațiilor matematice;
- suport pentru operații matematice simple și complexe;
- afișarea contribuției fiecărui worker;
- dezvoltarea unei interfețe web intuitive.

1.3. Lista MoSCoW

Categorie	**Descriere**
Must have	• Operații distribuite • Workeri multipli • Agregarea rezultatelor
Should have	• Interfață web • Istoric al operațiilor • Selectarea tipului de operație
Could have	• Afișarea timpilor de execuție • Grafice de performanță • Exportul rezultatelor
Won't have	• Autentificare utilizatori • Persistență în bază de date • Deploy în cloud

Tabela 1.1: Lista cerințelor MoSCoW pentru aplicația Calculator Distribuit

1.4. Cazuri de utilizare

Aplicația este destinată unui utilizator final care poate interacționa cu sistemul prin intermediul interfeței web. Principalele cazuri de utilizare sunt:

- introducerea unei operații matematice;
- selectarea tipului de operație;
- distribuirea calculelor către workeri;
- vizualizarea rezultatului final;
- analizarea rezultatelor parțiale ale fiecărui worker;
- consultarea istoricului operațiilor efectuate.

Capitolul 2. Arhitectura aplicației

2.1. Prezentare generală

Aplicația *Calculator Distribuit* este construită pe o arhitectură de tip **coordinator–worker**, specifică sistemelor distribuite. Această arhitectură permite împărțirea sarcinilor de calcul între mai multe noduri de procesare independente, coordonate de un nod central.

Rolul principal al coordinatorului este de a primi cererile utilizatorului, de a distribui datele către workeri și de a agrega rezultatele parțiale pentru obținerea rezultatului final. Workerii sunt procese independente care execută calcule matematice asupra unei părți din datele primite.

2.2. Structura generală a aplicației

Aplicația este alcătuită din următoarele componente principale:

- **Frontend web** – interfața de interacțiune cu utilizatorul;
- **Coordinator** – componenta centrală de control;
- **Workeri** – noduri de calcul independente.

Fiecare componentă are un rol bine definit, iar comunicarea dintre ele se realizează prin mecanisme standard de rețea.

2.3. Frontend web

Frontend-ul aplicației este realizat folosind tehnologii web moderne, precum HTML, CSS și JavaScript. Acesta oferă o interfață grafică intuitivă care permite utilizatorului:

- selectarea operației matematice dorite;
- introducerea valorilor de calcul;
- inițierea procesului de calcul distribuit;
- vizualizarea rezultatului final și a rezultatelor parțiale.

Interfața web comunică cu coordinatorul prin intermediul unor cereri HTTP, acesta fiind responsabil de inițierea procesului de calcul distribuit.

2.4. Coordinator

Coordinatorul reprezintă componenta centrală a aplicației și este implementat în limbajul Python. Principalele responsabilități ale coordinatorului sunt:

- primirea cererilor de calcul de la frontend;
- validarea datelor introduse de utilizator;
- împărțirea setului de date în sub-seturi;
- distribuirea sub-seturilor către workeri;
- colectarea rezultatelor parțiale;
- calcularea rezultatului final prin agregare.

Pentru operațiile care permit distribuirea, coordinatorul utilizează execuția paralelă pentru a îmbunătăți performanța aplicației.

2.5. Workeri

Workerii sunt procese independente care rulează pe porturi diferite și acționează ca noduri de calcul. Fiecare worker:

- primește o parte din datele de intrare;
- execută operația matematică specificată;
- returnează rezultatul parțial către coordinator.

Workerii nu comunică între ei, ceea ce asigură o separare clară a responsabilităților și o scalabilitate crescută a aplicației.

2.6. Comunicarea între componente

Comunicarea dintre coordinator și workeri se realizează folosind socket-uri TCP. Această abordare permite transmiterea eficientă a datelor și este potrivită pentru aplicațiile distribuite care necesită control asupra procesului de comunicare.

Comunicarea dintre frontend și coordinator se realizează prin intermediul unei interfețe web, folosind cereri HTTP.

2.7. Avantajele arhitecturii utilizate

Arhitectura aleasă oferă următoarele avantaje:

- execuție paralelă a operațiilor matematice;
- scalabilitate prin adăugarea de workeri suplimentari;
- separarea clară a responsabilităților;
- evidențierea practică a conceptelor de sistem distribuit.

Capitolul 3. Implementarea Frontend-ului

3.1. Rolul Frontend-ului

Frontend-ul reprezintă componenta aplicației prin intermediul căreia utilizatorul interacționează cu calculatorul distribuit. Acesta are rolul de a prelua datele introduse de utilizator, de a transmite cererile către coordinator și de a afișa rezultatele obținute într-o formă clară și intuitivă.

Interfața a fost concepută astfel încât să fie ușor de utilizat, punând accent pe claritatea informațiilor afișate și pe evidențierea caracterului distribuit al aplicației.

3.2. Tehnologii utilizate

Frontend-ul aplicației este realizat utilizând următoarele tehnologii:

- **HTML** – pentru structura paginii web;
- **CSS** – pentru stilizarea și organizarea vizuală a interfeței;
- **JavaScript** – pentru logica de interacțiune și comunicarea cu backend-ul.

Aceste tehnologii permit dezvoltarea unei interfețe web moderne, compatibile cu majoritatea browserelor.

3.3. Structura interfeței grafice

Interfața grafică este structurată în două zone principale:

- zona de control a calculatorului;
- zona de afișare a istoricului operațiilor.

Zona de control permite selectarea operației matematice, introducerea valorilor de calcul și inițierea procesului de calcul. Zona de istoric afișează operațiile efectuate anterior împreună cu rezultatele corespunzătoare.

3.4. Selectarea operațiilor matematice

Utilizatorul poate selecta tipul de operație matematică dintr-o listă predefinită. Printre operațiile disponibile se numără:

- sumă;
- produs;
- medie aritmetică;
- minim și maxim;
- factorial;
- putere;
- CMMDC și CMMMC.

Această abordare reduce riscul introducerii unor comenzi invalide și îmbunătățește experiența utilizatorului.

3.5. Introducerea datelor

Valorile numerice necesare efectuării calculelor sunt introduse sub forma unor numere separate prin spațiu. Interfața este adaptată în funcție de operația selectată, oferind indicații clare utilizatorului privind numărul și tipul valorilor necesare.

3.6. Afișarea rezultatelor

După finalizarea calculelor, frontend-ul afișează:

- rezultatul final al operației;
- contribuția fiecărui worker sub forma rezultatelor parțiale;
- istoricul operațiilor efectuate.

Rezultatele sunt afișate în format zecimal complet, fără utilizarea notației științifice, pentru a asigura lizibilitatea numerelor foarte mari.

3.7. Istoricul operațiilor

Istoricul operațiilor este afișat într-o zonă dedicată a interfeței și este actualizat automat după fiecare calcul. Acesta permite utilizatorului să urmărească succesiunea operațiilor efectuate și rezultatele obținute.

3.8. Avantajele interfeței realizate

Principalele avantaje ale interfeței frontend sunt:

- ușurința în utilizare;
- claritatea informațiilor afișate;
- evidențierea caracterului distribuit al aplicației;
- feedback vizual rapid pentru fiecare operație.

Capitolul 4. Implementarea Backend-ului

4.1. Rolul Backend-ului

Backend-ul reprezintă componenta aplicației responsabilă de procesarea cererilor de calcul și de coordonarea execuției distribuite. Acesta asigură legătura dintre interfața web și nodurile de calcul, având rolul de a gestiona distribuirea sarcinilor și agregarea rezultatelor.

Prin intermediul backend-ului, aplicația realizează execuția paralelă a operațiilor matematice și evidențiază caracterul distribuit al sistemului.

4.2. Tehnologii utilizate

Backend-ul aplicației este implementat în limbajul Python și utilizează următoarele tehnologii și mecanisme:

- programare concurentă prin utilizarea firelor de execuție;
- socket-uri TCP pentru comunicarea între coordinator și workeri;
- FastAPI pentru expunerea unei interfețe web;
- mecanisme de agregare a rezultatelor parțiale.

Aceste tehnologii permit realizarea unui sistem flexibil și ușor de extins.

4.3. Coordinator

Coordinatorul este componenta centrală a backend-ului și are rolul de a gestiona întregul proces de calcul. Principalele responsabilități ale coordinatorului sunt:

- primirea cererilor de calcul de la frontend;
- parsarea și validarea comenzilor introduse de utilizator;
- identificarea operațiilor care pot fi distribuite;
- împărțirea datelor de intrare în sub-seturi;
- distribuirea sub-seturilor către workeri;
- colectarea rezultatelor parțiale;
- calcularea rezultatului final.

Coordinatorul utilizează execuția paralelă pentru a comunica simultan cu mai mulți workeri, reducând astfel timpul total de procesare.

4.4. Distribuirea sarcinilor

Pentru operațiile care permit distribuirea, setul de date este împărțit în mai multe sub-seturi, fiecare fiind transmis către un worker diferit. Această abordare permite executarea calculelor în paralel și crește eficiența aplicației pentru seturi mari de date.

Fiecare worker procesează independent datele primite și returnează un rezultat parțial către coordinator.

4.5. Agregarea rezultatelor

După primirea tuturor rezultatelor parțiale, coordinatorul efectuează agregarea acestora pentru a obține rezultatul final. Metoda de agregare depinde de tipul operației matematice executate:

- pentru sumă și produs, rezultatele parțiale sunt combinate prin operații aritmetice corespunzătoare;
- pentru medie, se agregă suma parțială și numărul de elemente;
- pentru minim și maxim, se determină valoarea extremă dintre rezultatele parțiale.

Rezultatul final este transmis către frontend împreună cu informații despre contribuția fiecărui worker.

4.6. Workeri

Workerii sunt procese independente care rulează pe porturi diferite și execută calcule matematice asupra datelor primite. Fiecare worker:

- primește o comandă și un set de date;
- execută operația specificată;
- returnează rezultatul parțial către coordinator.

Workerii nu comunică între ei, ceea ce asigură o arhitectură simplă și ușor de scalat prin adăugarea de noduri suplimentare.

4.7. Gestionarea erorilor

Backend-ul include mecanisme de tratare a erorilor pentru situații precum:

- introducerea unor date invalide;
- indisponibilitatea unui worker;
- erori de comunicare.

În cazul apariției unei erori, coordinatorul poate continua procesarea cu workerii disponibili și transmite un mesaj corespunzător către frontend.

4.8. Avantajele implementării backend

Implementarea backend-ului oferă următoarele avantaje:

- execuție paralelă eficientă;
- scalabilitate prin adăugarea de workeri;
- separarea clară a responsabilităților;
- evidențierea clară a principiilor sistemelor distribuite.

Capitolul 5. Concluzii

5.1. Obiective atinse

În urma implementării aplicației *Calculator Distribuit*, toate obiectivele stabilite în faza de analiză au fost îndeplinite. A fost realizată o arhitectură distribuită funcțională, bazată pe modelul coordinator–worker, care permite execuția paralelă a operațiilor matematice.

Aplicația demonstrează în mod practic modul în care sarcinile de calcul pot fi împărțite între mai multe noduri independente și agregate ulterior pentru obținerea unui rezultat final.

5.2. Rezultatele obținute

Rezultatele obținute confirmă funcționarea corectă a sistemului distribuit. Operațiile matematice sunt executate în paralel, iar contribuția fiecărui worker este afișată explicit în interfața aplicației.

Interfața web oferă utilizatorului o experiență intuitivă, permițând selectarea operațiilor, introducerea datelor și analizarea rezultatelor într-un mod clar și organizat. Istoricul operațiilor facilitează urmărirea calculelor efectuate.

5.3. Avantajele soluției propuse

Principalele avantaje ale soluției implementate sunt:

- execuție paralelă a calculelor matematice;
- scalabilitate prin adăugarea de workeri suplimentari;
- separarea clară a responsabilităților între componente;
- evidențierea practică a conceptelor de sisteme distribuite;
- interfață web modernă și ușor de utilizat.

5.4. Limitări ale aplicației

Deși aplicația îndeplinește obiectivele propuse, există anumite limitări:

- performanța este influențată de latența rețelei;
- pentru seturi mici de date, beneficiile distribuției sunt mai puțin vizibile;
- aplicația nu include mecanisme de persistență a datelor.

5.5. Direcții de dezvoltare viitoare

Aplicația oferă multiple posibilități de extindere, printre care:

- adăugarea unui mecanism de monitorizare a timpilor de execuție;
- implementarea unui sistem de balansare a sarcinilor;
- salvarea istoricului operațiilor într-o bază de date;
- extinderea aplicației pentru rulare în medii cloud;
- adăugarea unor mecanisme avansate de tratare a erorilor.

5.6. Concluzie finală

Prin realizarea acestui proiect a fost demonstrată importanța și utilitatea sistemelor distribuite în procesarea eficientă a datelor. Aplicația *Calculator Distribuit* constituie un exemplu relevant de utilizare a arhitecturii distribuite și oferă o bază solidă pentru dezvoltări ulterioare.