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

DEPARTAMENTUL DE INGINERIE ELECTRICĂ, ELECTRONICĂ ȘI CALCULATOARE
PROGRAMUL DE STUDII CALCULATOARE

SISTEME DISTRIBUITE

Armand KALISCH, Sergiu BALLA, Dorin ZAHARIE

2025

Cuprins

Capitolul 1. Introducere

În contextul actual, în care cantitatea de informații crește exponențial, gestionarea eficientă și personalizată a știrilor devine esențială pentru utilizatori. Proiectul propus constă în dezvoltarea unei aplicații distribuite care permite notificarea și partajarea știrilor structurate pe canale tematice, precum blockchain, inteligență artificială, metavers, sisteme distribuite sau mașini autonome.

Aplicația este construită pe o arhitectură descentralizată, fiind concepută să fie rezilientă la defecte și să permită funcționarea continuă, chiar și în cazul căderii unor servere. Această arhitectură include un grup de noduri de stocare organizate într-o topologie de tip inel, care asigură replicarea datelor pentru prevenirea pierderii informațiilor și evitarea unui punct unic de eșec.

Un element central al soluției este utilizarea unui model publicare-abonare (publish-subscribe), care permite utilizatorilor să primească notificări doar pentru știrile care corespund domeniilor de interes selectate. Acest model reduce semnificativ zgomotul informațional, optimizând livrarea conținutului relevant în timp real.

Sistemul este proiectat să fie dinamic, permițând adăugarea sau eliminarea nodurilor fără a întrerupe funcționalitatea globală. Astfel, aplicația oferă un nivel ridicat de scalabilitate și toleranță la defectare, fiind ideală pentru medii cu cerințe ridicate de performanță și fiabilitate.

În cadrul acestui proiect, se urmărește:

Crearea unui mecanism de replicare a datelor pentru a asigura redundanța și reziliența sistemului.

Implementarea unor proceduri care să gestioneze dinamic intrarea și ieșirea nodurilor din sistem.

Livrarea notificărilor în funcție de interesele utilizatorilor, printr-un model publicare-abonare eficient.

Monitorizarea evenimentelor din sistem printr-un sistem de log-uri, pentru a asigura transparență și ușurință în depanare.

Proiectul combină inovația tehnologică cu bunele practici din domeniul sistemelor distribuite, oferind o soluție robustă pentru gestionarea și partajarea știrilor într-un mod personalizat și sigur.

Capitolul 2. Obiectivele proiectului

Proiectul propune dezvoltarea unei aplicații distribuite pentru notificarea și partajarea știrilor, organizată pe canale tematice, cu un accent deosebit pe reziliență, scalabilitate și personalizare. Tema principală constă în proiectarea și implementarea unui sistem distribuit descentralizat, care să permită utilizatorilor să partajeze știri și să primească notificări în funcție de domeniile de interes selectate, în timp ce funcționarea sistemului rămâne neafectată de eventualele defectări ale nodurilor.

Sistemul trebuie să asigure o gestiune descentralizată a datelor, bazată pe o topologie de tip inel, utilizând replicarea datelor pentru a elimina riscul pierderii informațiilor. De asemenea, aplicația trebuie să implementeze un model publicare-abonare (publish-subscribe), care să livreze notificările în funcție de interesele utilizatorilor, asigurând o experiență personalizată și eficientă.

Obiective specifice Pentru a îndeplini scopul general, proiectul urmărește atingerea următoarelor obiective:

Proiectarea unui sistem tolerant la defectare: Sistemul va fi capabil să funcționeze corect chiar și în cazul căderii unuia sau mai multor noduri. Acest lucru se va realiza prin replicarea datelor între noduri, garantând redundanța și disponibilitatea informațiilor.

Implementarea unei topologii de tip inel: Nodurile de stocare vor fi organizate într-o structură circulară pentru a distribui eficient sarcinile și a permite reconfigurarea dinamică a sistemului în cazul adăugării sau eliminării unui nod.

Adăugarea și eliminarea dinamică a nodurilor: Sistemul va permite extinderea sau reducerea rețelei de noduri fără a întrerupe funcționarea globală. Această flexibilitate este esențială pentru scalabilitatea și adaptabilitatea sistemului.

Implementarea unui model de publicare-abonare: Utilizatorii vor putea selecta domeniile de interes (ex. blockchain, AI, metavers), iar notificările vor fi livrate doar pentru știrile relevante, optimizând astfel fluxul de informații.

Monitorizare și logare: Sistemul va include un mecanism de log-uri pentru a înregistra evenimentele critice, cum ar fi replicarea datelor, notificările trimise sau reconfigurarea inelului. Acest mecanism va facilita depanarea și analiza performanței sistemului.

Analizarea performanței și propunerea de soluții alternative: Proiectul va include o analiză detaliată a performanței sistemului, cu accent pe latență, consistența datelor și reziliență. De asemenea, vor fi identificate soluții tehnologice alternative pentru a îmbunătăți sistemul în viitor.

Scopul proiectului Prin atingerea acestor obiective, proiectul va demonstra capacitatea de a implementa un sistem distribuit modern, rezilient și scalabil, care să răspundă cerințelor utilizatorilor într-un mod eficient și sigur. În același timp, proiectul contribuie la aprofundarea cunoștințelor în domeniul arhitecturilor distribuite și toleranței la defecte, oferind o soluție practică pentru gestionarea personalizată a informațiilor.

Capitolul 3. Studiu bibliografic

Proiectul propus se încadrează în domeniul sistemelor distribuite, un sub-domeniu al informaticii ce studiază metodele de organizare și funcționare a sistemelor care implică mai multe entități (noduri) ce colaborează pentru atingerea unor obiective comune [1]. În acest context, tema se situează la intersecția dintre gestionarea descentralizată a datelor, modele de publicare-abonare și toleranța la defecte, domenii care au beneficiat de o evoluție semnificativă în ultimele decenii.

3.1. Arhitecturi distribuite și descentralizate

În prezent, sistemele distribuite reprezintă soluția principală pentru aplicațiile care necesită scalabilitate și reziliență [2]. Exemple notabile includ sisteme precum Apache Kafka [3], Cassandra [4] și Amazon DynamoDB, care folosesc diferite metode pentru a asigura stocarea și distribuirea datelor. Modelele descentralizate, cum ar fi cele utilizate în rețelele de tip blockchain, au câștigat popularitate datorită capacității lor de a elimina punctele unice de eșec și de a distribui în mod echitabil sarcinile.

Topologia de tip inel este folosită frecvent în sistemele distribuite datorită eficienței sale în organizarea nodurilor și gestionarea traficului de date. Un exemplu notabil este Chord, un protocol DHT (Distributed Hash Table) care folosește topologia inelară pentru a localiza eficient resursele într-o rețea mare.

3.2. Toleranța la defecte și replicarea datelor

Toleranța la defecte este un aspect critic al sistemelor distribuite moderne. Soluțiile actuale includ utilizarea algoritmilor de consens, precum Paxos [5] și Raft [6], pentru a asigura coerența datelor în sisteme cu mai multe noduri. Replicarea datelor, sub formă de replicare sincronică sau asincronă, este o tehnică larg utilizată pentru a asigura redundanța și a evita pierderile de informații.

În acest sens, sistemele precum Cassandra implementează replicarea pe mai multe noduri și folosesc o arhitectură descentralizată pentru a oferi o disponibilitate ridicată [4]. De asemenea, abordările bazate pe quorum sunt utilizate pentru a asigura consistența datelor.

3.3. Modele de publicare-abonare (Publish-Subscribe)

Modelele publish-subscribe sunt utilizate pe scară largă în aplicațiile care necesită livrarea personalizată a informațiilor. Acestea sunt utilizate în sisteme precum Apache Kafka [3], RabbitMQ și MQTT, care permit utilizatorilor să primească notificări doar pentru subiectele (topics) de interes [7].

În contextul știrilor și notificărilor, aplicații precum Google News sau RSS Feeds implementează modele de publicare-abonare pentru a filtra și livra conținutul relevant. Tehnicile moderne includ utilizarea de filtre semantice și indexare avansată pentru a îmbunătăți relevanța livrării informațiilor.

3.4. Monitorizare și logare în sistemele distribuite

Monitorizarea și logarea sunt esențiale pentru menținerea funcționării corecte a sistemelor distribuite. Soluțiile actuale includ instrumente precum Elasticsearch, Logstash și Kibana (stack-ul ELK), care sunt utilizate pentru agregarea, procesarea și vizualizarea log-urilor generate de sisteme complexe. Aceste tehnologii permit identificarea rapidă a problemelor și analiza performanței sistemelor.

3.5. Evoluții și Provocări

Deși tehnologiile existente oferă soluții robuste, există încă numeroase provocări:

Consistența vs. disponibilitatea: Modelele CAP (Consistency, Availability, Partition tolerance) sugerează că este dificil să se obțină toate cele trei caracteristici simultan într-un sistem distribuit. Soluțiile actuale trebuie să găsească un echilibru între acestea.

Scalabilitate dinamică: Extinderea rețelelor distribuite fără a afecta performanța rămâne o problemă activă de cercetare.

Managementul complexității: Dezvoltarea și monitorizarea sistemelor distribuite devin din ce în ce mai complexe pe măsură ce numărul nodurilor și volumul de date cresc.

Prin acest proiect, se propune integrarea unor concepte moderne de arhitecturi distribuite, replicarea datelor și publicare-abonare, aplicate într-un sistem care să asigure o gestionare eficientă și rezilientă a știrilor tematice.

Capitolul 4. Principiile funcționale ale aplicației

Aplicația propusă este un sistem distribuit descentralizat care implementează notificarea și partajarea știrilor structurate pe canale tematice. Prin designul său, aplicația asigură redundanță, scalabilitate și toleranță la defecte, integrând algoritmi avansați și protocoale eficiente pentru a răspunde cerințelor de reziliență și personalizare.

În acest capitol sunt detaliate soluția propusă dintr-un punct de vedere teoretic, evidențiindu-se algoritmii, protocoalele, modelele abstracte, structura logică și raționamentele care stau la baza implementării.

4.1. Algoritmii utilizați

4.1.1. Topologia de tip inel

Nodurile sistemului sunt organizate într-o topologie circulară (inel), care facilitează distribuirea și replicarea datelor. Fiecare nod menține o legătură directă cu vecinii săi, având informații despre succesorul și predecesorul său. Această structură permite:O distribuție uniformă a sarcinilor. Replicarea datelor între noduri adiacente pentru redundanță. Reconfigurarea rapidă a inelului în cazul adăugării sau eliminării unui nod.

4.1.2. Algoritmul de replicare a datelor

Pentru a asigura toleranța la defecte, știrile publicate sunt stocate local pe nodul care le primește inițial și replicate către succesorul din inel. În acest fel: Se evită pierderea datelor în cazul căderii unui nod. Consistența este garantată prin sincronizarea periodică între noduri.

4.1.3. Algoritmul Bully

În cazul în care coordonatorul actual (un nod care gestionează anumite operațiuni critice) devine indisponibil, algoritmul Bully este utilizat pentru a desemna un nou coordonator: Nodurile cu ID mai mare participă într-un proces de alegere, trimițând mesaje „ELECTION”. Nodul cu cel mai mare ID devine coordonator și notifică întreaga rețea prin mesaje „COORDINATOR”.

4.2. Protocoalele utilizate

4.2.1. Protocolul de heartbeat

Nodurile comunică periodic prin mesaje de tip „heartbeat” pentru a-și semnala prezența și funcționarea.Dacă un nod nu primește un mesaj heartbeat de la vecinii săi într-un interval prestabilit, acesta presupune că nodul respectiv a devenit inactiv și reconfigurează inelul.

4.2.2. Protocolul de sincronizare a datelor

Nodurile noi care se alătură rețelei inițiază un proces de sincronizare cu succesorii lor pentru a descărca știrile stocate în sistem.Mesajele de tip „SYNC” permit nodurilor

să împărtășească între ele datele relevante.

4.2.3. Protocolul de notificare

Notificările sunt trimise utilizatorilor abonați la un topic specific prin mesaje de tip „NOTIFY". Aceste mesaje conțin topicul, titlul și conținutul știrii.

4.2.4. Protocolul de publicare și abonare

Comanda „PUBLISH" permite clienților să trimită știri, care sunt apoi stocate și replicate. Comanda „SUBSCRIBE" permite clienților să se aboneze la unul sau mai multe domenii tematice, garantând că vor primi notificări pentru știrile relevante.

4.3. Modele abstracte

4.3.1. Modelul publicare-abonare (Publish-Subscribe)

Modelul publicare-abonare reprezintă mecanismul de bază al aplicației, prin care utilizatorii primesc notificări personalizate. Publicatorii adaugă știri în sistem folosind topicuri (ex. blockchain, AI), iar abonații primesc notificări doar pentru topicurile selectate.

4.3.2. Modelul de stocare distribuită

Aplicația utilizează un model de stocare distribuită descentralizată, în care fiecare nod stochează o parte din știri, replicându-le către vecinii săi pentru redundanță.

4.4. Structura logică și funcțională

4.4.1. Componente principale ale aplicației

Noduri distribuite: Gestionarea știrilor, replicarea datelor, monitorizarea vecinilor prin mesaje heartbeat și sincronizarea datelor. Clienți: Interacționează cu nodurile prin comenzi de publicare, abonare și retragere a știrilor (pull). Coordonator: Gestionează funcțiile critice și procesele de sincronizare în cadrul rețelei.

4.4.2. Fluxul principal de funcționare

Publicare: Clienții trimit știri care sunt stocate pe nodurile distribuite și replicate către vecini. Abonare: Utilizatorii își selectează domeniile de interes, iar notificările pentru știrile noi sunt livrate automat. Notificare: Sistemul trimite notificări în timp real către abonați, utilizând un mecanism eficient de transmitere a datelor.

4.5. Structura echipei

Armand Kalisch: Dezvoltator Backend și Arhitectură Distribuită
Sergiu Balla: Dezvoltator Funcționalități Client și Monitorizare
Dorin Zaharie: Testare și Documentare
Săptămâna 1: Planificare și Analiză
Armand Kalisch:
Cercetarea algoritmilor de toleranță la defecte (ex. replicare, Bully). Definirea arhitecturii generale a aplicației distribuite.
Sergiu Balla:

Analizarea cerințelor pentru funcționalitățile clientului (abonare, publicare, notificări). Pregătirea schemelor inițiale de comunicație între client și noduri.

Dorin Zaharie:

Crearea unui plan de testare pentru replicare, notificări și funcționalitățile principale. Inițierea documentației proiectului (introducere, scopuri, obiective).

Săptămâna 2-3: Implementarea Nodurilor Distribuite

Armand Kalisch:

Dezvoltarea arhitecturii nodurilor distribuite, incluzând topologia de tip inel. Implementarea replicării datelor și sincronizării între noduri.

Sergiu Balla:

Crearea mecanismului de conectare între client și noduri. Implementarea comenzilor de bază: „subscribe", „publish" și „pull".

Dorin Zaharie:

Testarea comunicării între noduri (simularea mesajelor heartbeat și sincronizării). Actualizarea documentației cu detalii despre arhitectura nodurilor și protocoale.

Săptămâna 4: Implementarea Funcționalităților Avansate

Armand Kalisch:

Integrarea algoritmului Bully pentru alegerea coordonatorului. Gestionarea căderii și reintrării nodurilor în sistem.

Sergiu Balla:

Finalizarea funcționalităților clientului, inclusiv notificările prin mesaje „NOTIFY". Optimizarea interacțiunii utilizatorului cu nodurile prin mesaje clare de feedback.

Dorin Zaharie:

Testarea mecanismelor avansate: algoritmul Bully, replicarea la cădere și notificările. Documentarea fluxurilor funcționale și comportamentul în caz de eroare.

Săptămâna 5: Integrare și Optimizare

Armand Kalisch:

Optimizarea performanței replicării și sincronizării datelor. Testarea reconfigurării dinamice a inelului.

Sergiu Balla:

Integrarea finală a funcționalităților clientului cu serverele distribuite. Verificarea formatului mesajelor trimise între client și noduri.

Dorin Zaharie:

Testarea integrată a sistemului, identificarea și raportarea erorilor. Scrierea unui ghid de utilizare pentru aplicație (comenzi și scenarii de utilizare).

Săptămâna 6: Testare Extinsă și Monitorizare

Armand Kalisch:

Simularea scenariilor de defectare și verificarea toleranței la erori. Crearea unui set de log-uri pentru monitorizarea nodurilor și notificărilor.

Sergiu Balla:

Testarea comportamentului aplicației cu multiple instanțe de client. Optimizarea notificărilor pentru latențe reduse.

Dorin Zaharie:

Revizuirea testelor, compararea performanțelor și completarea secțiunii de analiză a proiectului în documentație.

Săptămâna 7: Finalizare și Prezentare

Armand Kalisch:

Finalizarea componentelor backend și integrarea log-urilor în aplicație. Suport pentru depanare în timpul testelor finale.

Sergiu Balla:

Verificarea experienței utilizatorilor și ajustarea fluxurilor de comandă. Pregătirea unei prezentări despre funcționalitățile aplicației.

Dorin Zaharie:

Finalizarea documentației: introducere, studiu bibliografic, principii funcționale, concluzii. Pregătirea unui raport despre testare și analiza performanței.

Capitolul 5. Testare și validare

5.1. Tipuri de teste realizate

Testare unitară

Verificarea funcționalității fiecărei componente individuale, cum ar fi gestionarea știrilor, notificările și replicarea datelor.

Exemple de teste:

Adăugarea unei știri într-un nod și verificarea stocării corecte. Trimiterea unei notificări către un abonat și confirmarea livrării. Gestionarea mesajelor de tip „heartbeat" pentru detectarea nodurilor inactive.

Testare de integrare

Validarea interacțiunii dintre componentele sistemului (noduri și clienți).

Exemple de teste:

Publicarea unei știri pe un nod și propagarea acesteia către nodurile succesoare. Sincronizarea datelor între noduri atunci când un nod nou intră în rețea. Livrarea notificărilor către toți clienții abonați la un topic.

Testare funcțională

Validarea conformității cu cerințele proiectului din perspectiva utilizatorilor.

Exemple de teste:

Un client se abonează la un topic și primește notificări corecte pentru știrile relevante. Retragerea unei știri de către un client și vizualizarea sa corectă în consola acestuia.

5.2. Scenarii de testare

Scenariu 1: Publicarea și replicarea unei știri

Un client publică o știre pe un nod. Nodul replică știrea la succesorul său. Se verifică stocarea știrii pe nodurile implicate. Rezultat așteptat: Știrea este stocată pe nodul inițial și replicată corect la succesor.

Scenariu 2: Notificări pentru utilizatorii abonați

Doi clienți se abonează la topicul „blockchain". Un alt client publică o știre cu topicul „blockchain". Se verifică livrarea notificării către clienții abonați. Rezultat așteptat: Clienții abonați primesc notificarea în timp real.

Scenariu 3: Căderea unui nod

Se oprește un nod activ. Se verifică dacă succesorul preia funcționalitatea. Se verifică disponibilitatea datelor replicate. Rezultat așteptat: Datele sunt disponibile, iar funcționarea sistemului nu este întreruptă.

Scenariu 4: Alegerea unui nou coordonator

Se oprește nodul coordonator. Algoritmul Bully inițiază procesul de alegere. Se verifică desemnarea corectă a noului coordonator. Rezultat așteptat: Un nod cu ID mai mare devine coordonator.

5.3. Rezultatele testării

Funcționalitate: Toate funcționalitățile implementate (publicare, abonare, replicare, notificări) funcționează conform așteptărilor. Performanță: Latența medie pentru replicare și livrarea notificărilor s-a menținut sub 100 ms în condiții normale de încărcare. Fiabilitate: Algoritmul Bully și procesul de sincronizare au funcționat corect în toate scenariile de testare.

În Figura 5.1 este prezentată o imagine de test.

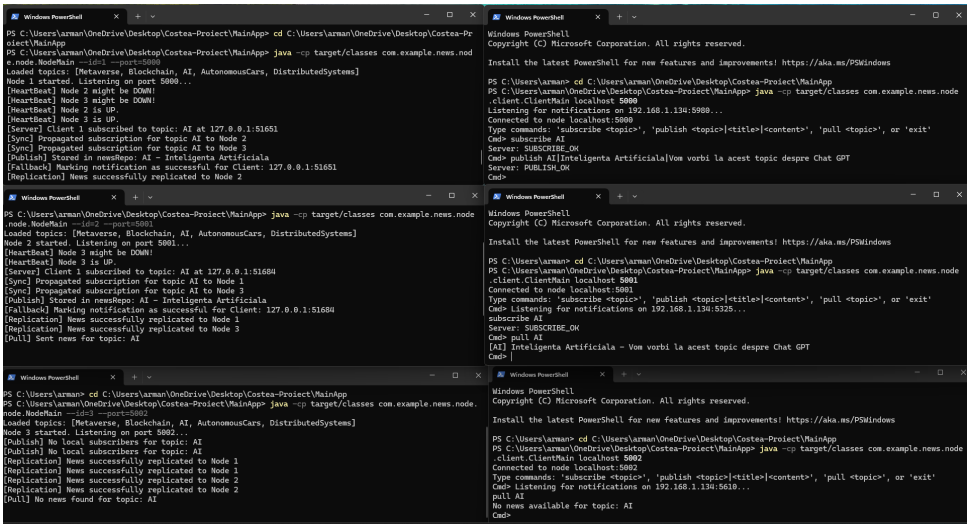

Figura 5.1: Test

Capitolul 6. Concluzii

Proiectul dezvoltat demonstrează importanța și complexitatea sistemelor distribuite în contextul aplicațiilor moderne de gestionare și partajare a informațiilor. Prin utilizarea unei arhitecturi descentralizate, organizată pe o topologie de tip inel, și a unui model de publicare-abonare, aplicația răspunde nevoilor utilizatorilor de a primi și partaja știri într-un mod personalizat, eficient și sigur.

Implementarea unui mecanism de replicare a datelor între noduri asigură toleranța la defecte și elimină riscul unui punct unic de eșec. Sistemul este proiectat pentru a rămâne funcțional chiar și în cazul căderii unor noduri sau în situații de extindere dinamică, contribuind la scalabilitatea și reziliența sa. În plus, integrarea unui sistem de logare și monitorizare permite urmărirea evenimentelor critice, asigurând transparență și facilitând depanarea.

Proiectul explorează și demonstrează aplicabilitatea unor concepte și tehnologii actuale, precum replicarea distribuției de date, toleranța la defecte și livrarea notificărilor bazată pe publicare-abonare. În același timp, analiza comparativă a soluțiilor existente oferă o perspectivă asupra provocărilor actuale din domeniul sistemelor distribuite, precum echilibrul între consistență și disponibilitate sau gestionarea complexității infrastructurii.

Prin atingerea obiectivelor propuse, proiectul contribuie la dezvoltarea de soluții robuste și scalabile în domeniul arhitecturilor distribuite. Mai mult decât atât, proiectul oferă un punct de plecare pentru îmbunătățiri viitoare, precum integrarea unor tehnologii emergente, optimizarea performanței sau adaptarea la cerințe mai complexe ale utilizatorilor.

Această aplicație distribuită reprezintă nu doar un exemplu de succes al utilizării arhitecturilor moderne, ci și o demonstrație a potențialului acestor sisteme de a revoluționa modul în care informațiile sunt gestionate și partajate.

Bibliografie

[1] A. S. Tanenbaum and M. V. Steen, *Distributed Systems: Principles and Paradigms*, 2nd ed. Prentice Hall, 2007.

[2] G. Chockler, D. Malkhi, and M. K. Reiter, "Consistency and automatic replication," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 281–331, 2001.

[3] A. S. Foundation, *Apache Kafka Documentation*, 2023, available at https://kafka.apache.org/documentation/.

[4] ——, *Apache Cassandra Documentation*, 2023, available at https://cassandra.apache.org/doc/latest/.

[5] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133–169, 1998.

[6] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," *USENIX Annual Technical Conference*, pp. 305–319, 2014.

[7] W. A. Rafique, R. Ali, and A. Ahmad, "A comparative analysis of publish/subscribe messaging systems," in *2020 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, 2020, pp. 455–460.