UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENŢĂ / MASTER:
Informatică / Informatică Aplicată/ Inteligenţă Artificială şi Calcul Distribuit / Inginerie Software/ Securitate Cibernetică/ Bioinformatică

LUCRARE DE LICENŢĂ/DIZERTAŢIE

COORDONATOR:
Prof./Conf./Lect. Dr. Nume Prenume

ABSOLVENT:
Nume Prenume

TIMIŞOARA
2020

UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENŢĂ
Informatică

LUCRARE DE LICENŢĂ

COORDONATOR:
Lect. Dr. Fortis Alexandra

ABSOLVENT:
Furmanek Carina

TIMIŞOARA
2020

Abstract

Seturile mari de date reprezintă o paradigmă revoluționară în gestionarea și analiza informațiilor, schimbând fundamental modul în care interacționăm cu lumea datelor. Această lucrare explorează provocările și soluțiile asociate cu prelucrarea și analiza acestor volume masive de date, evidențiind importanța abordărilor inovatoare, cum ar fi machine learning și framework-urile Big Data, în contextul platformelor precum Apache Hadoop și Apache Spark.

Cuprins

1 Introducere **5**
1.1 Problema abordată 5
1.2 Posibile Abordări 7
1.3 Perspective asupra aplicatiei 9

Bibliography **13**

Capitolul 1

Introducere

1.1 Problema abordată

Seturile mari de date (în engleză 'Big data') reprezintă o cantitate masiva de date care nu poate fi stocată, procesată și analizată folosiind metode tradiționale precum baze de date relaționale. Analiza acestor seturi de date reprezintă procesul de extragere a informațiilor semnificative din acestea.

Seturile mari de date au fost numite o revoluție care vă schimba modul în care trăim, lucrăm și gândim. Scopul principal al acestei revoluții este utilizarea acestor cantități mari de date pentru a permite descoperirea de cunoștințe și luarea de decizii mai bune. Analiza datelor implică diverse abordări, tehnologii și instrumente, cum ar fi vizualizarea datelor și analiză statistică.

În această eră caracterizată de un nivel imens existent de date, este necesar să utilizăm tehnici de analiză cât mai sofisticate pe seturi imense și diverse de date pentru a produce aceste cunoștințe și informații utile.

Această practică se regăsește în domenii precum sectorul bancar și de asigurări, sănătate, educație, industria social media și de divertisment, aplicații bioinformatice, agricultură etc. Prelucrarea seturilor mari de date folosind metode traditonale este o sarcină imposibilă. Astfel, pentru a descoperi tipare de date, tendințe și asocieri ascunse, într-un mod eficient, se pot utiliza algoritmi de machine learning.

Evoluția tehnologiei a condus la explozia fenomenului Big Data din mai multe motive semnificative:

Generarea masivă de date: În era actuală, fiecare acțiune pe care o întreprindem generează date. De la interacțiunile online și tranzacțiile comerciale până la dispozitivele IoT (Internet of Things) și senzori, cantitatea de date produse constant este imensă.

Imposibilitatea gestionării cu instrumente tradiționale: Cantitatea enormă de date generată nu poate fi gestionată eficient cu instrumentele tradiționale de procesare și stocare a datelor. Bazele de date relaționale și alte soluții clasice devin inadecvate în fața volumului și diversității datelor.

Avansarea tehnologiilor IoT: Dispozitivele conectate la internet, precum senzorii, dispozitivele smart și alte obiecte IoT, contribuie la generarea continuă a datelor. Aceste date sunt diverse că format și vin în timp real, impunând necesitatea unor tehnologii care să facă față acestei complexități.

Impactul rețelelor de socializare: Rețelele de socializare joacă un rol esențial în generarea Big Data. Postările, comentariile, like-urile și alte activități online generează o cantitate uriașă de date care necesită instrumente specializate pentru analiză și extragerea de informații relevante.

Necesitatea de a extrage cunoștințe utile: Big Data oferă o cantitate imensă de informații, dar aceasta trebuie să fie transformată în cunoștințe utile și relevante. Analiza Big Data folosind tehnici precum machine learning și data analytics permite identificarea de modele, tendințe și asocieri care pot oferi informații valoroase pentru luarea deciziilor.

Fenomenul big data poate fi înțeles mai clar cunoscând cele diferite V-uri asociate cu ele - Volum, Viteză, Variație, Veridicitate și Valoare.
1. Volum: Acesta indică cantitatea imensă de date produse în fiecare secundă, oscilând între terabytes și zettabytes. Aceste seturi masive de date pot fi gestionate cu ajutorul sistemelor distribuite.
2. Viteză: Termenul reprezintă ritmul la care datele sunt produse și procesate pentru a satisface cerințele.
3. Variație: Acesta indică gama diversificată de date pe care le putem utiliza.
4. Veridicitate: Acesta se referă la calitatea datelor. Adică indică sesizările, zgomotul, anormalitățile etc. în date.
5. Valoare: Acesta indică cunoștințele prețioase dezvăluite din date.

Machine Learning este o zonă de cercetare interdisciplinară care combină idei din mai multe ramuri ale științei, cum ar fi inteligența artificială, statistica, matematica, etc. Scopul principal al cercetării în domeniul machine learning este pe dezvoltarea de algoritmi rapizi și eficienți care pot face predicții pe datele analizate. Când este vorba de analiza datelor, machine learning este o abordare utilizată pentru crearea de modele de predicție.

În funcție de natura datelor disponibile, cele două categorii principale de sarcini de învățare sunt: învățarea supervizată, când atât întrările, cât și ieșirile dorite (etichetele) sunt cunoscute, și sistemul învață să facă asocierea între întrări și ieșiri, și învățarea nesupervizată, când ieșirile dorite nu sunt cunoscute, iar sistemul descoperă singur structura din date. Clasificarea și regresia sunt exemple de învățare supervizată: în clasificare, ieșirile iau valori discrete (etichete de clasă), în timp ce în regresie, ieșirile sunt continue. Exemple de algoritmi de clasificare includ k-nearest neighbour, regresia logistică și Mașina de Suport Vectorial (SVM), în timp ce exemple de regresie includ Regresia Vectorilor de Suport (SVR), regresia liniară și regresia polinomială. Anumiți algoritmi, cum ar fi rețelele neuronale, pot fi folosiți atât pentru clasificare, cât și pentru regresie. Învățarea nesupervizată include clustering-ul, care grupează obiectele pe baza unor criterii de similitudine stabilite; k-means este un exemplu de astfel de algoritm. Analiza predictivă se bazează pe învățarea automată pentru a dezvolta modele construite cu date din trecut în încercarea de a prezice viitorul; mai mulți algoritmi, inclusiv SVR, rețele neuronale și Naïve Bayes, pot fi folosiți în acest scop. O presupunere comună în machine este că algoritmii învăța mai bine cu mai multe date disponibile și, prin urmare, pot oferi rezultate mai precise Majoritatea algoritmilor tradiționali machine learning au fost implementați pentru seturi de date care puteau fi încarcați în memorie în întregime. O altă presupunere este că întregul set de date este disponibil pentru procesare în momentul antrenamentului. Big Data încalcă aceste

presupuneri, făcând algoritmii tradiționali inutilizabili sau împiedicându-le semnificativ performanța. Au fost dezvoltate mai multe tehnici pentru a adapta algoritmii de învățare automată pentru a lucra cu seturi de date mari: exemplele includ noi paradigme de procesare, cum ar fi MapReduce și cadre de procesare distribuită, cum ar fi Hadoop. Ramuri ale machine learning-ului, inclusiv deep learning și online, au fost de asemenea adaptate în efortul de a depăși provocările învățării automate cu Big Data.

1.2 Posibile Abordări

Apache Hadoop este o platformă care permite procesarea distribuită a seturilor extinse de date pe grupuri de calculatoare. Poate scala de la gestionarea sarcinilor pe un singur server la administrarea acestora pe mii de mașini, fiecare oferind propria capacitate locală de calcul și stocare. În loc să se bazeze exclusiv pe hardware pentru a asigura disponibilitatea, Hadoop este proiectat pentru a detecta și gestiona eșecuri la nivel aplicație. Abordarea aceasta garantează un serviciu cu disbonibilitate ridicată, chiar și în situația unor eventuale eșecuri în computerele individuale din cadrul grupurilor.

Hadoop este alcătuit din 3 componente concepute pentru a lucra cu volumuri mari de date:

Unitatea de stocare - sistemul distribuit de fișiere Hadoop (HDFS - Hadoop Distributed File System).

Stocarea unei cantități masive de date pe un singur calculator este nerealistă și de multe ori imposibilă, de aceea datele sunt distribuite pe mai multe calculatoare și HDFS împarte datele în mai multe blocuri, care sunt apoi stocate pe mai multe noduri de date în cluster. Dimensiunea implicită a fiecărui bloc este de 128 MB. Dacă un nod de date se defectează, partea de date în cauza nu este pierdută deoarece HDFS creza copii ale datelor și le stochează pe mai multe sisteme, făcându-l astfel tolerant la erori. Când un bloc este creat, acesta este replicat și stocat pe diferite noduri de date.

Procesarea datelor - MapReduce

După ce datele sunt stocate, acestea trebuie procesate, iar aici intervine a două componentă a Hadoop, numită MapReduce. Prin utilizarea unei metode tradiționale de prelucrare a datelor, întregul set de date ar fi procesat pe o singură mașînă cu un singur procesor. Această abordare este extrem de ineficientă, în special atunci când se lucrează cu o varietate extinsă de date. Pentru a remedia această ineficiență, MapReduce împarte datele în segmente și procesează fiecare segment în mod independent pe nodurile de date. Rezultatele individuale sunt apoi agregate pentru a obține rezultatul final. MapReduce procesează fiecare parte a datelor mari în mod individual și consolidează rezultatele la sfârșit. Această abordare îmbunătățește echilibrul încărcării și reduce semnificativ timpul de procesare.

Odată ce job-ul MapReduce este pregătit, acesta este executat pe clusterul Hadoop. Execuția depinde de un set de resurse, inclusiv RÂM, lățimea de bandă a rețelei și CPU. Mai multe job-uri pot rula pe Hadoop simultan, iar fiecare job necesită resurse specifice pentru a se finaliza cu succes. Pentru a gestiona eficient aceste resurse, inter-

vine a treia componentă a Hadoop, YARN (Yet Another Resource Negotiator).

YARN este alcătuit dintr-un într-un manager de resurse, un manager de noduri, un application master și containere. Managerul de resurse alocă resurse, managerul de noduri gestionează nodurile și monitorizează utilizarea resurselor, în timp ce containerele găzduiesc o colecție de resurse fizice. YARN procesează cererile de job-uri și supraveghează resursele clusterului în Hadoop.

În plus față de aceste componente, Hadoop integrează diverse instrumente și framework-uri pentru gestionarea, procesarea și analizarea datelor mari. Ecosistemul Hadoop include și alte componente precum Hive, Pig și Apache Spark.

Apache Spark este un motor de analiză unificat open-source pentru prelucrarea datelor la scară mare. A fost dezvoltat pentru a depăși limitările modelului Hadoop MapReduce, introducând o abordare mai flexibilă și eficientă în prelucrarea datelor.

Sistemul Spark se bazează pe setul de date distribuit și rezistent (RDD), care este o abstractizare ce reprezintă o colecție de obiecte read-only, împărțite pe un cluster de calcul. RDD-ul poate fi creat din fișiere text, baze de date SQL, baze de date NoSQL, HDFS, stocare în cloud etc. RDD-urile permit funcții standard de MapReduce, dar și unirea seturilor de date, filtrarea acestora și agregarea lor. Procesarea RDD-urilor se face în întregime în memorie.

Prelucrarea unui RDD se realizează prin două componente numete drivere și executori. Când un program este executat, începe cu driverul, care creează un SparkContext (orchestrator), care la rândul lui analizează codul și determină sarcinile posibile de realizat. Acesta generează un plan fizic și apoi utilizează un manager de cluster pentru a coordona toți executorii pentru a programa și rula sarcinile. Planificatorul din cadrul managerului de cluster este numit un DAG (graf aciclic direct), care vă atribui sarcini și ordinea de executare către nodurile de lucru, iar apoi executorii aflați în nodurile de lucru sunt lansați dinamic de către managerul de cluster. Acești executori rulează o sarcină și returnează rezultatul către driver. Tot ceea ce este descris aici constituie motorul de bază pentru Spark. Pe lângă aceasta, există mai multe biblioteci care permit o interacțiune ușoară cu motorul de bază.

Acestea includ Spark SQL pentru lucrul cu date structurate și cadre de date, ce pot fi manipulate folosind SQL sau Hive, precum și Java, Python sau Scala. MLlib pentru furnizarea unui cadru distribuit de machine learning și GraphX pentru prelucrarea distribuită a graficelor. Spark poate fi rulat pe un motor de procesare, cum ar fi Hadoop Yarn, precum și Apache Mesos, Kubernetes și Docker Swarm. O soluție Spark administrată poate fi gestionată cu Amazon EMR, Google Cloud Dataproc și Azure HDInsight pe diferite platforme.
Cel mai mare avantaj a lui Spark este viteza viteza acestuia. Prelucrarea rapidă datorată designului RDD și procesării în memorie îl face să ruleze la viteze semnificativ mai mari decât alte opțiuni. Este foarte flexibil în programabilitate, permițând dezvoltatorilor să utilizeze limbajul lor de programare preferat, poate gestiona prelucrarea datelor aproape în timp real datorită capacității de a citi datele de flux în memorie, iar Spark permite analize mai avansate datorită capacității de a utiliza funcții SQL, de machine learning, prelucrare grafică și module suplimentare cum ar fi Spark R pentru a lucra cu limbajul de programare R. Aceste caracteristici au făcut că Spark să fie opțiunea preferată față de MapReduce, care are o prelucrare lentă și este mai puțîn ideal pentru funcțiile de date în timp real OLTP, filtrare și execuție iterativă.

O abordare comună este utilizarea HDFS (sistemul de fișiere distribuit Hadoop) că strat de stocare a datelor, iar apoi Spark că strat de prelucrare și interacțiune a datelor. Principalul dezavantaj al lui Spark este cantitatea mare de RÂM necesară pentru a realiza toată prelucrarea în memorie.

1.3 Perspective asupra aplicatiei

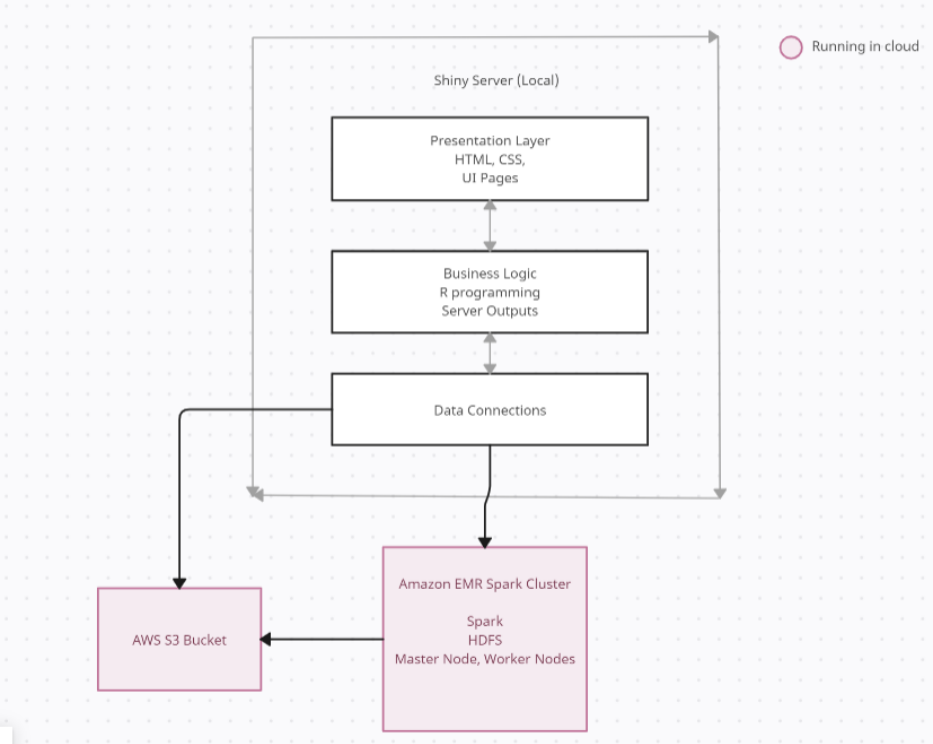

Figura 1.1: Diagrama a arhitecturii sistemului

Diagrama descrie arhitectura aplicației, cu accent pe componente și pe interacțiunile acestora.

Componenta Amazon EMR Spark Cluster: Aceasta este componenta din cloud unde are loc procesarea datelor. Cluster-ul constă dintr-un ”master node” și ”worker nodes”, rulând Spark pe baza HDFS (Hadoop Distributed File System). Cluster-ul este responsabil pentru executarea sarcinilor de procesare a datelor și stocarea rezultatelor intermediare în HDFS.

Amazon S3 Bucket: Acesta este un serviciu de stocare în cloud folosit pentru a stoca rezultatele datelor procesate de către cluster-ul Spark.

Serverul Shiny (Local): Aceasta este componenta locală care rulează pe un laptop sau pe o mașînă locală. Include:

Stratul de Prezentare: Conține HTML, CSS și pagini UI, pentru a reda interfața utilizatorului a aplicației web.

Business Logic: Conține codul în R, care include logica pentru solicitarea și manipularea datelor pentru vizualizare.

Data Connections: Gestionează conexiunile la sursele externe de date, în acest caz, la recipientul AWS S3 și cluster-ul Amazon EMR.

Datele sunt procesate și analizate în cloud pe Amazon EMR Spark Cluster. Rezultatele sunt stocate în serviciul de stocare AWS S3. Serverul local Shiny preia datele procesate din S3 și le folosește pentru afișarea vizualizare . Diagrama este de asemenea, marcată cu un simbol care indică faptul că componentele colorate în roșu rulează în cloud, distingându-le de componentele locale.

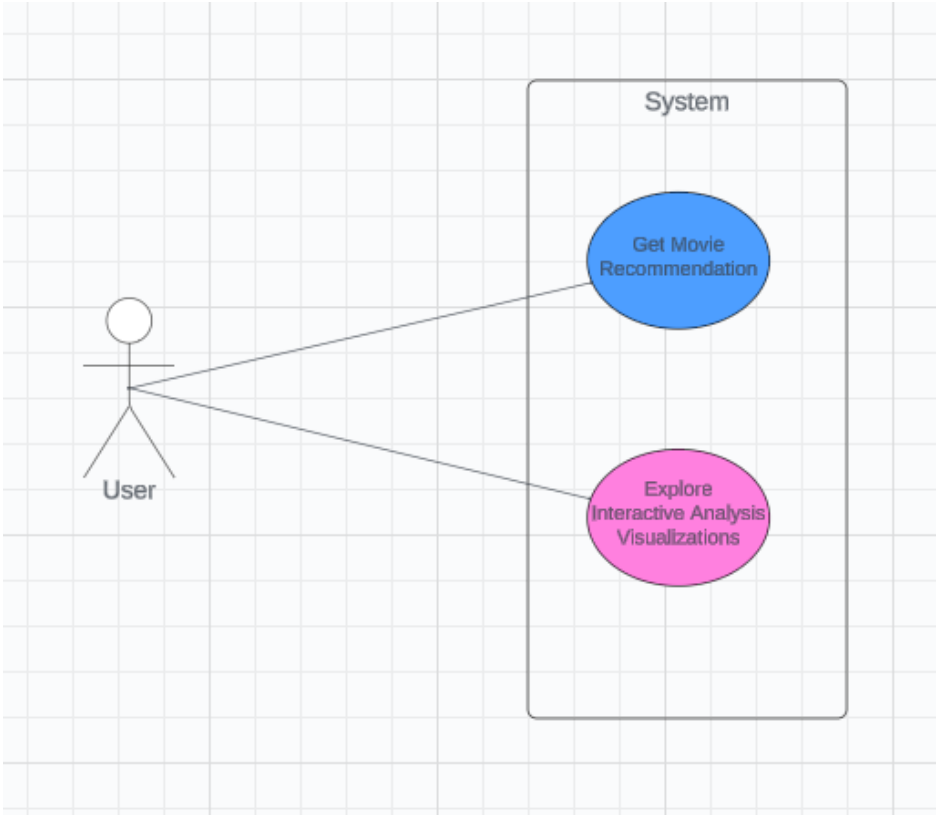

Figura 1.2: Cazuri de utilizare

Cazul de utilizare "Get Movie Recommendations" este un proces simplificat conceput pentru a utiliza filtrarea colaborativă pentru a oferi utilizatorilor sugestii personalizate de filme.

Atunci când un utilizator interacționează cu aplicația, i se cere să introducă o listă de filme pe care le preferă. Aplicația folosește aceste date de intrare că bază pentru a obține un rezultat personalizat.

Pe baza acestei intrări, backend-ul execută un algoritm de filtrare colaborativă. Acest algoritm analizează datele stocate în HDFS pentru a identifica tipare și similitudini cu preferințele altor utilizatori. Scopul este de a găsi alți utilizatori cărora le-au plăcut filme similare și apoi de a sugera noi filme utilizatorului actual pe baza preferințelor pe care le-au avut alti utilizatori similari.

Cazul de utilizare "Explore Interactive Analysis Visualizations" este o componentă care permite utilizatorilor observe și să interacționeze cu vizualizări bazate pe analiza setului de date.

La selectarea opțiunii de explorare a vizualizărilor, utilizatorului i se prezintă o interfață care prezintă diferite opțiuni de vizualizare. Aceste date se bazează pe datele de film procesate și analizate, stocate în Amazon S3.

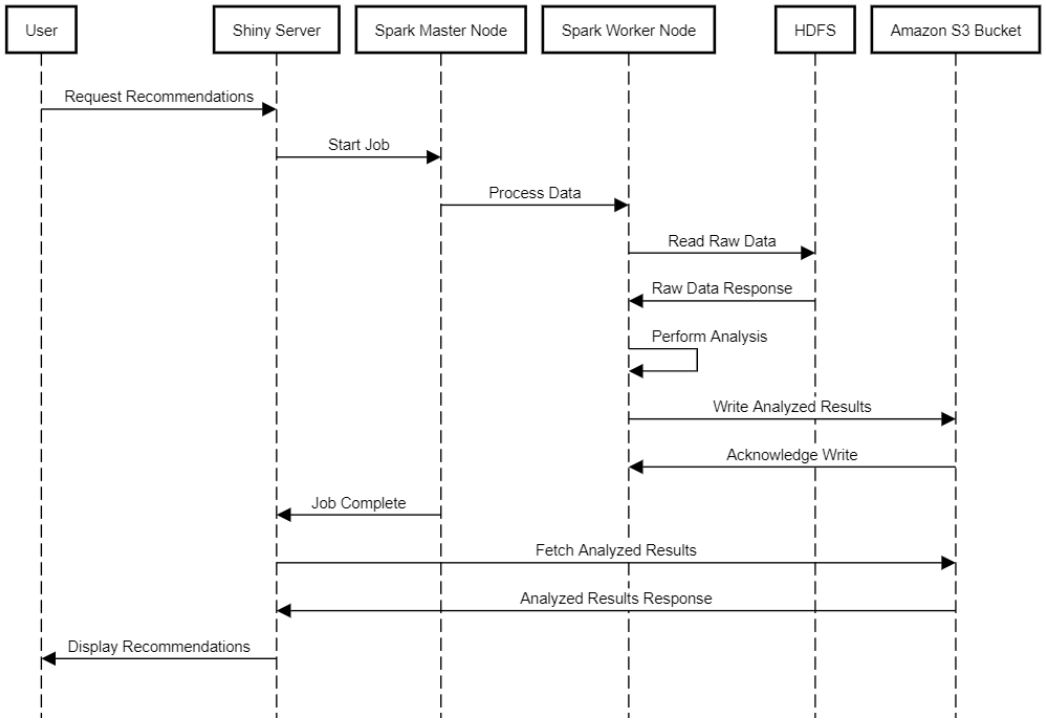

Figura 1.3: Diagrama de secventa - Request Movie Visualizations

Diagrama ilustrează fluxul de interacțiuni pentru cazul de utilizare "Request Recommendations" și include cinci entități: Uș̦er, serverul Shiny, master node-ul Spark, worker node-ul Spark, HDFS și Amazon S3 Bucket. Secvența de evenimente este:

Utilizatorul inițiază cererea: Procesul începe cu solicitarea de către utilizator a recomandărilor de filme, care este primită de către serverul Shiny.

Inițierea sarcinii: Serverul Shiny, la primirea cererii utilizatorului, inițiază o sarcină de lucru prin trimiterea unui semnal către master node-ul Spark. Rolul acestui nod este de a controla sarcinile de procesare a datelor prin delegarea sarcinilor către worker node-urile disponibile.

Prelucrarea datelor: Worker node-ul este cel care este responsabil pentru procesarea datelor. Acesta comunică cu sistemul distribuit HDFS pentru a citi datele necesare pentru a genera recomandări. HDFS răspunde cu datele neprocesate, pe care worker node-ul le utilizează pentru a efectua analiza necesară pentru procesul de filtrare colaborativă.

Stocarea analizei și a rezultatelor: După finalizarea analizei, Spark Worker Node scrie rezultatele analizate în serviciul de stocare în cloud Amazon S3 Bucket.

Confirmarea scrierii: S3 Bucket confirmă operațiunea de scriere, confirmând că rezultatele analizate au fost stocate cu succes.

Finalizarea și recuperarea lucrărilor: Master node-ul semnalează serverului Shiny finalizarea sarcinii. Ulterior, serverul Shiny solicită rezultatele analizate de la S3 Bucket.

Răspuns cu rezultate: S3 Bucket răspunde cu rezultatele analizate.

Afișarea recomandărilor: Server-ul Shiny prezintă utilizatorului recomandările de filme.

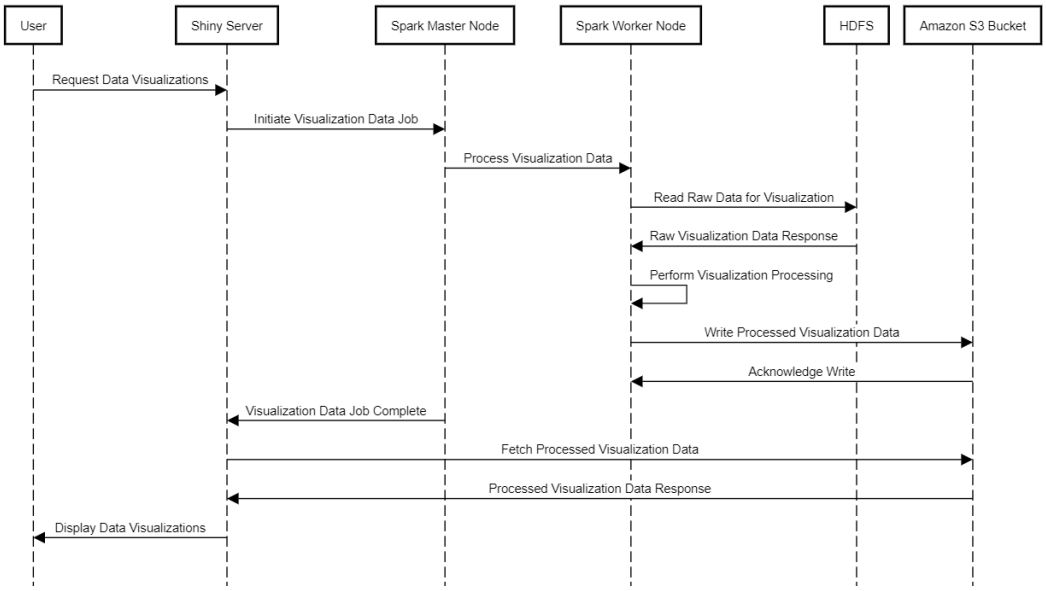

Figura 1.4: Diagrama de secventa - Request Data Visualizations

Diagrama descrie fluxul de proces pentru cazul de utilizare "Request Data Visualizations" .Fluxul de evenimente, așa cum este descris în diagramă, este:

Cerere din partea utilizatorului: Procesul începe cu o cerere de vizualizare a datelor din partea unui utilizator, făcută către serverul Shiny. După ce solicitarea este primită, serverul interpretează cererea și crează o sarcină pentru a genera vizualizările solicitate.

Master Node-ul primește sarcina de la server, pe care o distribuie la worker node-urile disponibile.

Worker node-ul execută etapele de procesare a datelor pentru a crea vizualizări.

Worker node-ul interacționează cu HDFS pentru a citi datele necesare. HDFS servește drept depozit inițial pentru datele neprocesate și răspunde cu informațiile solicitate. Nodul procesează date pentru a fi transformate într-un format vizualizabil.

După ce datele au fost procesate, acestea sunt stocate în Amazon S3 Bucket. Acest serviciu de stocare în cloud confirmă operațiunea de scriere, confirmând stocarea cu succes a datelor de vizualizare procesate. La finalizarea sarcinilor de procesare și stocare a datelor, nodul principal Spark informează serverul Shiny că sarcina este finalizată.

Serverul Shiny extrage apoi datele pentru vizualizare procesate din S3 Bucket, pe care le afișează utilizatorului.

Vizualizările sunt prezentate într-un format interactiv, permițându-i utilizatorului să interacționeze cu datele într-un mod semnificativ.

Bibliografie