Hadoop Distributed File System

Masterand

Ing. Voicu Adrian – Master IISC AN 2

Cuprins

[1 Introducere 3](#_Toc429668543)

[1.1 Ce este HDFS? 3](#_Toc429668544)

[1.2 Ce este Hadoop si care este legaura cu HDFS? 3](#_Toc429668545)

[1.3 Modul de interactiune cu HDFS 3](#_Toc429668546)

[2 Arhitectura HDFS 4](#_Toc429668547)

[2.1 Name Nodes si Data Nodes 4](#_Toc429668548)

[2.2 NameNode 5](#_Toc429668549)

[2.3 DataNode 5](#_Toc429668550)

[2.4 Relatiile dintre NameNode si DataNode 5](#_Toc429668551)

[2.5 Replicarea Datelor 6](#_Toc429668552)

[2.6 Organizarea Datelor 6](#_Toc429668553)

[4 Fiabilitatea de stocare a datelor 7](#_Toc429668554)

[5 Integritatea datelor 8](#_Toc429668555)

[6 Sincronizarea metadatelor 8](#_Toc429668556)

[7 Bibliografie 9](#_Toc429668557)

# 1 Introducere

## 1.1 Ce este HDFS?

Hadoop Distributed File System (HDFS) – este un sistem de fisiere distribuit, cu o toleranta foarte mare la defecte, sistem proiectat sa ruleze pe componente hardware low-cost. HDFS ofera viteza ridicata in transferul de date ceea ce face ca acest sistem sa fie perfect pentru seturi mari de date (big data).

## 1.2 Ce este Hadoop si care este legaura cu HDFS?

Apache Hadoop este un software open-source care permite prelucrarea distribuita a unor seturi mari de date, distribuite peste clustere de servere. Este conceput pentru a scala de la un singur server la mii de masini avand un grad de toleranta la erori foarte ridicat. Elasticitatea cluster-elor provine de la capacitatea software-ului de a detecta si de a manipula esecurile la nivelul aplicatiei, fara sa se bazeze in totatalitate pe hardware de buna calitate.

Hadoop este ideal pentru stocarea cantitatilor mari de date, cum ar fi Terabytes si pentabytes si foloseste HDFS ca si sistem de stocare. HDFS permite distribuirea datelor pe mai multe noduri din diferite clustere. Accesul fisierelor se face transparent, utilizatorul nu sezisează ca fisierele pe care le accesează sunt impartite de-a lungul nodurilor din cluster/ clustere.

Accesul la fisiere de date se face prin rularea unor comenzi direct, pe baza modelului de procesare MapReduce.

## 1.3 Modul de interactiune cu HDFS

HDFS poate sa fie accesat prin mai multe metode. HFDS pune la dispozitie un interfata programabila Java (API) si un wrapper in limbajul C pentru API-ul respective. In plus, se poate folosi un browser pentru a putea cauta fisiere HDFS.

Iata cateva aplicatii care ofera o interfata pentru HDFS:

* FileSystem shell
* DFSAdmin
* Fsck
* Name nodes and data nodes

# 2 Arhitectura HDFS

HDFS este format din noduri interconectate, acolo unde se gasesc fisierele si directoarele. Un grup HDFS consta dintr-un singur nod numit NameNode. Acest NameNode este responsabil de gestiunea spatiului de nume de fisiere si reglementarea sistemului de acces client pentru fisiere.

DataNodes (nodurile de date) sunt entitatile care stocheaza blocuri de date in fisiere.

## 2.1 Name Nodes si Data Nodes

In cadrul HDFS, Name node-ul gestioneaza operatiile de deschidere, inchidere si redenumire de fisiere si directoare. NameNode-ul este responsabil si de maparea blocurilor de date de nodurile de date. Acestea din urma sunt cele care gestioneaza cererile de scriere si citire provenite de la clientii HDFS.

In figura de mai jos este ilustrata structura clusterelor si sistemul de management distribuit al datelor. Pentru fiecare cluster se poate observa ca exista decat un singur NameNode.

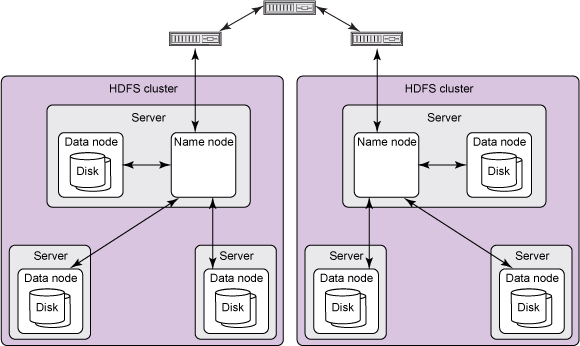


Fig 1 – Arhitectura HDFS

## 2.2 NameNode

Namespace-ul HDFS este o ierarhie de fisiere si directoare. Fisierele si directoarele sunt reprezentate pe NodeName prin inodes. Aceste inodes inregistreaza date cu privire la permisiuni, modificari, timpi de acces, spatii de nume sau informații cu privire la cantitatea de spatiu pe disc. Continutul fisierului este impartit in blocuri mari (de obicei 128 MB) iar fiecare bloc al fisierului este replicat in mod independent pe mai multe NodeData (noduri de date).

NameNode, referitor la modul de replicare al fisierelor, acesta mentine arborele spatiului de nume si maparea blocurilor de fisiere cu DataNodes. Fiecare cluster are NameNode-ul propriu. Fiecare cluster poate avea mii de DadaNodes si zeci de mii de client HDFS astfel incat fiecare DataNode sa poata executa mai multe sarcini in acelasi timp, concurent.

## 2.3 DataNode

Fiecare replica bloc de pe un DataNode este formata din doua fisiere in sistemul de fisiere local. Un fisier contine datele in sine iar celalalt contine metadatele blocului. Dimensiunea fisierului de date este egala cu lungimea efectiva a blocului si nu are nevoie de spatiu suplimentar pentru a rotunji la domesiunea nominala a blocului asa cum se intampla in sistemele de fisiere traditionale. Astfel, daca un bloc este pe jumatate plin are nevoie de doar jumatate din spatiul intreg al blocului pe discul local.

In timpul pornirii fiecare DataNode se conecteaza la NameNode si efectueaza o strangere de mana. Scopul aceste strangeri de mana este de a verifica ID-ul spatiului de nume si versiunea software-ului DataNode -ului. In cazul in care id-ul nu se potriveste cu cel din NameNode, DataNode-ul respectiv se inchide automat.

Id-ul namespace-ului este atribuit instantei sistemului de fisiere atunci cand este formatat. Id-ul este stocat pe toate nodurile din cluster. In cazul in care id-ul DataNode-ului este diferit acesta nu se va putea alatura cluster-ului, astfel se protejeaza integritatea sistemului de fisiere. Unui DataNode nou initializat in este permisa alaturarea la cluster pentru a putea primi ID-ul spatiului de nume al cluster-ului.

## 2.4 Relatiile dintre NameNode si DataNode

HDFS este constriuit folosind limbajul de programare Java, astfel orice masina care accepta limbajul de programare Java poate rula HDFS. O instalare tipica de cluster are un singur NameNode si posibil un DadaNode. Fiecare masina din cluster ruleaza un DadaNode.

DataNode-urile cer in mod repetitiv NameNode-ului pentru intructiuni. NameNode-ul nu se poate conecta direct la un DataNode, el returneaza valorile functiilor invocate de catre DataNode.

Fiecare DataNode mentine un socket server deschis astfel incat alte noduri sa poate scrie, citi date. Hostul sau portul socket server-ului este cunoscut de NameNode, care furnizeaza informatii altor clienti sau nodurilor de date.

Toate protocoalele de comunicare pe care HDFS le foloseste sunt construite pe baza protocolului TCP/IP. Clientii HDFS se conecteaza la un port TCP (Transmission Control Protocol) deschis pe NameNode iar mai apoi comunica cu NameNode-ul printr-un protocol de tip RPC (Remote Procedure Call). DataNode-urile comunica cu NameNode-ul folosind un protocol bazat pe SMB (Server Message Block).

## 2.5 Replicarea Datelor

HDFS replica blocuri de fisiere pentru toleranta ridicata la erori. In momentul crearii unui fisier se poate specifica numarul de replici care ulterior poate sa fie modificat in orice moment. NameNode-ul este cel care ia toate deciziile cu privire la replicarea blocurilor de date.

HDFS foloseste un sistem de replicare inteligent pentru o fiabilitate si performanta sporita. Optimizarea sistemul de replicare face ca HDFS sa fie unic comparativ cu celelalte sisteme distribuite. De asemenea, un alt beneficiu adus de optimizarea sistemului de replicare este folosirea unei politici care face posibila utilizarea constienta si eficienta a benzii de retea.

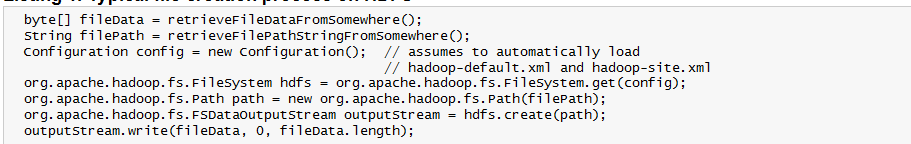
Un sistem HDFS mare opereaza de obicei peste foarte multe clustere de calculatoare. Comunicatia dintre doua noduri de date din clustere diferite este mai lenta decat nodurile de date aflate in acelasi cluster. Prin urmare, NameNode incearca sa optimizeze comunicarea dintre nodurile de date. NameNode-ul identifica locatia nodurilor de date pe baza Id-ul rack-ului din care fac parte.

## 2.6 Organizarea Datelor

Unul dintre principalele obiective ale sistemului HDFS este sa suporte cantitati mari de date. Marimea tipica a unui bloc de date HDFS este de 64MB. Astfel fiecare fisier HDFS contine unul sau mai multe blocuri de 64 MB. HDFS incearca sa plaseze fiecare bloc pe noduri de date separate.

Manipularea fisierelor de pe HDFS este siminalara cu procesele utilizate cu alte sisteme de fisiere. Cu toate acestea, pentru ca HDFS este un sistem multi-masina care este transparent utilizatorilor ca un singur disk, tot codul care manipuleaza fisierele de pe HDFS foloseste o subclasa numita org.apache.hadoop.fs.FileSystem.

In imaginea de mai jos este ilustrat un proces tipic de creare a unui fisier in HDFS.



Atunci cand un client creaza un fisier in HDFS, el mai intai cache-uieste datele intr-un fisier local temporal apoi redirectioneaza scrierile ulterioare catre fisierul temporal. Cand fisierul temporal acumuleaza suficiente date pentru a umple un bloc HDFS, clientul raporteaza NameNode-ului acest lucru, care mai apoi ia decizia si transforma blocul de date temporal intr-un nod permanent de date. Din momentul acela clientul inchide fisierul temporal si curata toate datele ramase in nodul de date nou creat.

Dupa curatarea datelor temporale NameNode-ul face commit la nodul de date pe disk.

Atunci cand un client acumuleaza un bloc complet cu date de utilizator, acesta preia o lista de noduri de date de la NameNode care contin replica blocului respectiv. Clientul trimite intregul bloc de date primului nod de date identificat in lista. In timp ce nodul primeste bucati de date, le scrie pe disk si transfera copii catre celelalte noduri de date din lista preluata. Urmatorul nod efectueaza acelasi proces. Procesul de tip pipeline este repetat pana cand factorul de replicare se incheie.

# 4 Fiabilitatea de stocare a datelor

Unul dintre obiectivele importante ale HDFS este acela de a stoca date fiabil, chiar si atunci cand apar defectiuni la NameNode-uri, noduri de date sau la partitiile de retea.

Pentru a depasi esecurile de acest gen primul pas facut de HDFS este de detecta problemele. HDFS foloseste mesaje de tip „heartbeat” pentru a detecta conexiunea intre NameNode si nodurile de date.

Pierderea de conectivitatea intre cele doua entitati poate sa fie cauzata de mai multe lucruri. Prin urmare fiecare nod de date trimite mesaje heartbeat periodice catre NameNode-ul de care apartine, astfel incat NameNode-ul sa stie ca s-a pierdut conexiunea cu nodul de date din momentul in care nu a incetat sa mai primeasca mesaje. In acest caz NameNode-ul marcheaza nodul de date respectiv ca fiind mort sau indisponibil si inceteaza sa mai trimita cereri. Datele stocate pe acel node care nu mai este disponibil nu mai sunt disponibile clientului de HDFS, el fiind eliminat din sistem din motive de eficienta.

In cazul in care factorul de replicare scade sub valoarea lor minima din pricina unui nod care nu mai raspunde, NameNode-ul initiaza procesul de replicare suplimentara astfel incat factorul de replicare sa fie la o stare normalizata.

In figura de mai jos se poate observa comunicare de tip mesaje heartbeat dintre NameNode si nodurile de date.

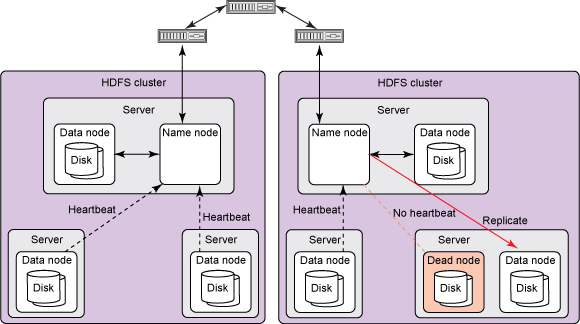


Fig 2 – Procesul heartbeat HDFS

# 5 Integritatea datelor

In ceea ce priveste integritatea datelor, HDFS foloseste validarea de tip „cheksum” a continutului fisierelor HDFS prin stocare „checksum” in fisiere separate, ascunse in acelasi spatiu de nume ca si datele reale. Atunci cand un client preaia date de fisiere, se poate verifica daca datele preluate sunt aceleasi ca „checksum” stocat in fisierul asociat.

Namespace-ul HDFS este stocat cu ajutorul unui jurnal de tranzactii tinut de fiecare NameNode. Namespace-ul sistemului de fisiere, impreuna cu fisierele bloc de mapare si proprietatile sistemul de fisiere sunt stocate intr-un fisier numic Fslmage. Cand un NameNode este initializat, citeste fisierul mentionat anterior impreuna cu alte fisiere si aplica tranzactiile si informatiile de status identificate in aceste fisiere.

# 6 Sincronizarea metadatelor

Un NameNode utilizeaza un fisier jurnal cunoscut sub numele de EditLog pentru a inregistra toate tranzactiile care au loc in sistemul de fisiere HFDS al metadatelor. Daca fisierele EditLog si FsImage devin corupte, instanta HDFS de care apartin inceteaza sa mai functioneze. Prin urmare, un NameNode accepta mai multe copii ale acestor fisiere. Cu mai multe copii ale acestor fisiere, orice modificare facuta se propaga sincron si la restul de copii. Atunci cand un NameNode reporneste, se foloseste cea mai recenta versiune a fisierelor FsImage si FileLog pentru a se initializa.

# 7 Bibliografie

[1][**http://www.aosabook.org/en/hdfs.html**](http://www.aosabook.org/en/hdfs.html)

[2][**http://blog.matthewrathbone.com/2013/04/17/what-is-hadoop.html**](http://blog.matthewrathbone.com/2013/04/17/what-is-hadoop.html)

[3][**https://hadoop.apache.org/**](https://hadoop.apache.org/)

[4][**https://en.wikipedia.org/wiki/Apache\_Hadoop**](https://en.wikipedia.org/wiki/Apache_Hadoop)

[5][**http://hortonworks.com/hadoop/**](http://hortonworks.com/hadoop/)

[6][**http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html**](http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html)

[7][**http://www.sas.com/en\_us/insights/big-data/hadoop.html**](http://www.sas.com/en_us/insights/big-data/hadoop.html)