Apunts CE_5075 5.1

Iloc:	Institut d'Ensenyaments a Distància de les Illes Balears	Imprès per:	Carlos Sanchez Recio
Curs:	Big data aplicat	Data:	dijous, 13 de febrer 2025, 18:31
Llibre:	Apunts CE_5075 5.1		

Taula de continguts

1. Introducció

En aquest lliurament ens ocuparem de tres aspectes diferents relacionats amb la gestió dels sistemes distribuïts i, més en concret, dels clústers Hadoop.

Començarem introduint el que entenem per qualitat de les dades. Ja hem vist al llarg del curs que les dades de les quals disposam en un entorn de Big Data solen tenir una qualitat molt baixa: contenen informació incompleta, amb errors, amb *outliers* que distorsionen qualsevol anàlisi, etc. Així doncs, si volem treure profit d'aquestes dades (i encara més si volem utilitzar-les per a entrenar models d'aprenentatge automàtic), necessitam dur a terme un procés de millora de la seva qualitat. Aquí veurem les fases d'aquest procés, però els detalls de les tècniques emprades en aquestes fases es veuran en altres mòduls del curs d'especialització.

A continuació, ens centrarem en la integritat de les dades, el mecanisme que ens proporciona protecció contra l'alteració voluntària (maliciosa) o involuntària de les dades. La principal tècnica per a garantir aquesta integritat és el que anomenam sumes de verificació (*checksums*).

Per últim, la major part d'aquest lliurament la dedicarem al monitoratge, un dels aspectes més importants de qualsevol sistema d'informació, que encara es torna més crucial en el cas dels sistemes distribuïts. Veurem quines són les mètriques que els sistemes de monitoratge recullen i acabarem el lliurament veient algunes eines que podem emprar per al monitoratge de clústers Hadoop.

2. Qualitat de les dades

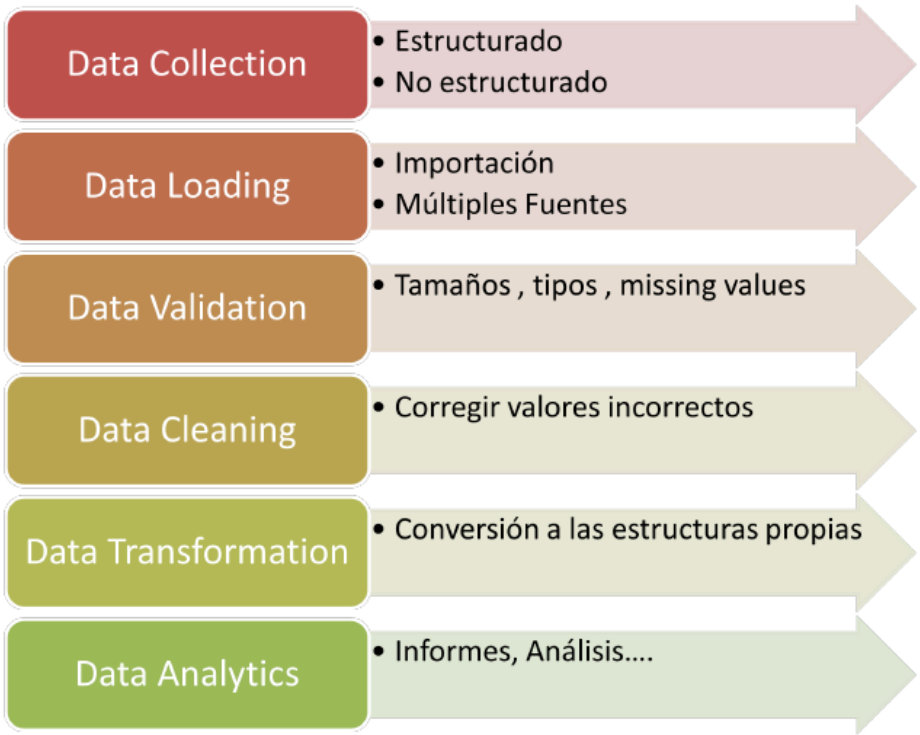
La qualitat de les dades se sol concebre com un concepte multidimensional definit com a l'**aptitud de les dades per a ser usades**.

La qualitat de les dades pot dependre de diversos factors (dimensions o característiques) com la completitud, la coherència, la disponibilitat, etc. L'avaluació de la qualitat de les dades implica el mesurament de les dimensions o criteris de qualitat que són rellevants per a cada cas d'ús. Una mètrica o mesura és un procediment per a mesurar una dimensió de qualitat de dades amb l'ajuda d'una eina. Aquestes mètriques són heurístiques que estan dissenyades per a adaptar-se a una situació d'avaluació específica.

Estudis recents han demostrat que la qualitat de les dades guardades en un magatzem de dades en Hadoop sol distar molt d'unes característiques acceptables. La mala qualitat de les dades pot suposar una pèrdua important per a l'organització en termes de reducció d'ingressos, serveis ineficients, major cost de manteniment, etc.

Així doncs, obtenir unes dades de bona qualitat, fiables, precises i significatives, és una tasca fonamental en una organització que vulgui treure profit de les seves dades massives. Tant és així que s'estima que molt més de la meitat del temps invertit en projectes de Big Data es destina al procés de preparació de les dades, és a dir, de millora de la seva qualitat. Aquest procés sol involucrar les tasques següents:

- ***Data Collection*** (Captura de Dades): Les dades es recullen de diverses fonts com a correus electrònics, bases de dades, arxius CSV, fulls de càlcul, sistema d'arxius, etc.
- ***Data Loading*** (Càrrega de Dades): Es carreguen les dades en un repositori de big data, per exemple, un entorn Hadoop (HDFS) o una base de dades NoSQL. Depenent dels requisits del client, l'actualització de les dades extretes es realitza amb freqüència diària, setmanal o mensual.
- ***Data Validation*** (Validació de Dades): Es comprova la qualitat de les dades, per exemple, el tipus de dades i els seus formats: números de telèfon, valors numèrics, etc.
- ***Data Cleaning*** (Neteja de Dades): Es detecten les dades incompletes, incorrectes, inexactes o irrellevants en els conjunts de dades i es proposa una solució a cadascuna de les situacions.
- ***Data Transformation*** (Transformació de Dades): Es converteixen les dades originals al format o estructura necessari (per exemple, per entrenar un model).
- ***Data Analytics*** (Analítica de Dades): Elaborar informes que van des del propi procés de validació de les dades (consistència, integritat, etc.) fins a tasques d'intel·ligència de negoci en general.



Imatge: *Procés de millora de la qualitat de les dades. Font: Universitat Castella-La Mancha*

Dins de la infraestructura Big Data es fa necessari, per tant, desenvolupar processos que analitzin les dades que han estat capturades mitjançant càrregues per lots o incrementals i emmagatzemades en HDFS per a la seva explotació per les diverses eines de l'ecosistema Hadoop. El sistema de qualitat de dades ha de permetre l'anàlisi de gran quantitat d'informació (de l'ordre de TB's) en temps reduïts. Funcionalment ha de ser capaç de realitzar tant les validacions bàsiques (valors perduts, tipus de dades i expressions regulars de format) com a validacions adaptades a les regles de negoci, així com generar els informes a aquest efecte que poden incloure: les anomalies detectades, el tipus d'incorrecció, la localització exacta de la mateixa (arxiu i línia), estadístiques diverses sobre la qualitat de les dades etc.

A més, una dimensió a tenir en compte és l'obsolescència de les dades, és per això que les tasques de validació han de permetre inferir la vigència de les dades durant un període de temps.

IMPORTANT

En el lliurament 7 del mòdul de Sistemes de big data veurem en detall algunes de les tècniques més emprades en aquest procés de preparació de dades (o *data wrangling*), com són la detecció de valors absents, d'outliers i de valors duplicats o el reescalat de les dades.

3. Integritat de les dades

La integritat de les dades és la garantia d'exactitud i coherència de les dades al llarg del seu cicle de vida (des que es registren fins que es destrueixen).

La integritat de les dades implica que s'han registrat les dades tal com es pretenia i que no s'han modificat involuntàriament al llarg del seu cicle de vida. El concepte és senzill, però la pràctica no ho és. La integritat de les dades és un component crític per a crear o dissenyar qualsevol sistema de programari que emmagatzemi o mogui dades.

La integritat de les dades és un aspecte molt rellevant dins d'un entorn Big Data atès que la majoria de les decisions crítiques es basen en les dades d'una organització. Si no es disposa de dades amb integritat i qualitat la majoria de les preguntes crucials per al progrés de l'organització quedaran sense resposta o aquesta serà imprecisa.

La integritat de les dades proporciona protecció contra l'alteració voluntària (maliciosa) o involuntària de les dades. Algunes de les causes de problemes amb la integritat de les dades són les següents:

- **Errors humans**: introducció de dades incorrectes, duplicació o eliminació de dades sense seguir el protocol adequat, errors en els procediments de protecció de la informació.
- **Errors de transferencia**: errors introduïts pels mecanismes de transmissió de dades.
- **Atacs**: els atacs de manipulació de dades, els atacs a la relació de confiança i els atacs de segrest de sessió són alguns dels més coneguts contra la integritat de les dades.
- **Errors de maquinari**: problemes de funcionament d'alguns dels dispositius del sistema. Són fallades que indiquen que el maquinari emmagatzema o transmet dades de manera incorrecta o incompleta, limita o elimina l'accés a les mateixes o, en general, dificulta l'ús de la informació.

Aquests són alguns conceptes relacionats amb el manteniment de la integritat de les dades:

- ***Data Provenance***: El concepte de provinença de les dades fa referència a les fonts des de les quals obtenim les dades. En Big Data, és comú que les dades de diferents fonts i orígens s'integrin (fusió de dades). No obstant això, si una o diverses fonts de dades combinades són de baixa qualitat és difícil filtrar-les després d'haver estat combinades i agregades. La procedència de les dades es basa en la identificació clara de cada unitat de dades i permet eliminar les dades de fonts no fiables. Suposa la comprovació de tots els estats de les dades des de l'estat inicial fins a l'estat actual. La depuració, la seguretat i els models de confiança són diverses aplicacions d'aquest concepte. Sense informació sobre la procedència de les dades, l'usuari mai arriba a saber d'on procedeixen les dades i quines transformacions s'han aplicat a aquestes.
- ***Data trustworthiness***: Per a aconseguir la confiabilitat de les dades en un entorn Big Data les organitzacions han d'implementar i automatitzar processos per a auditar, avaluar i netejar les seves dades. Això ha d'anar acompanyat per una infraestructura de dades que tengui en compte els processos humans juntament amb el programari. És necessari crear una cultura centrada en les dades que funcioni en conjunt amb l'automatització de la qualitat de les dades. Serà necessari, per tant, implementar tant mesures preventives per a identificar i resoldre els problemes de les dades en tota l'organització, com a tractaments eficaços per a combinar processos automatitzats i manuals per a resoldre els problemes que es poden presentar. La infraestructura necessària ha d'aprofitar els coneixements dels usuaris del negoci per a netejar les dades, així com eines sofisticades que permetin als enginyers de dades realitzar operacions complexes sense necessitat de coneixements de codificació. La confiabilitat de les dades es podria mesurar considerant dimensions com l'exhaustivitat, transparència, oportunitat i certificació de les dades. Amb un accés obert a dades completes, netes i fiables, els usuaris finals poden prendre decisions millors amb la màxima confiança.
- ***Data Loss***: Com a recursos amb valor les dades estan subjectes a robatori (pot ser pitjor que la pèrdua, si l'entitat que ho roba abusa del recurs), la indisponibilitat (no tan greu com la pèrdua o la corrupció, però no deixa de ser un inconvenient), la corrupció o la pèrdua permanent. Les dades es perden o deixen d'estar disponibles per al seu ús si el dispositiu d'emmagatzematge o el lloc amfitrió que conté les dades s'espenya o no respon. La indisponibilitat de les dades es pot evitar mantenint múltiples

còpies de les dades en diferents dispositius/llocs o, com fa Hadoop, mitjançant el particionament i distribució de les dades en diferents hosts.

- ***Data Deduplication***: La detecció de duplicats es refereix al procés d'identificar tuples en una o més relacions que es refereixen a la mateixa entitat del món real. Aquest procés sol implicar molts passos i opcions, incloent el disseny de mètriques de similitud per a avaluar la similitud d'un parell de registres, l'entrenament de classificadors per a determinar si un parell de registres són duplicats o no, l'agrupació de tots els registres per a obtenir clústers de registres que representen la mateixa entitat del món real, consolidant els clústers de registres en representacions úniques, dissenyar estratègies de bloqueig o distribuïdes per a augmentar l'escala del procés de deduplicació i humans per a decidir si un parell de registres són duplicats quan les màquines no estan segures.

3.1. Sumes de verificació

Garantir la integritat de les dades és una necessitat bàsica o en un entorn de processament Big Data per a aconseguir resultats precisos.

En un ecosistema Big Data les dades es manegen de forma distribuïda a causa del seu enorme volum. Per tant, aconseguir la integritat de les dades és un autèntic repte. HDFS ha estat concebut per a emmagatzemar qualsevol tipus de dades de manera distribuïda en forma de blocs de dades (descompon un enorme volum de dades en un conjunt de blocs individuals) amb compromís d'integritat de dades. Pot haver-hi múltiples causes que generin blocs de dades corruptes en HDFS, començant per l'operació d'I/O en el disc del sistema, la fallada de la xarxa, etc.

Una **suma de verificació** (***checksum***) és un número calculat a partir d'un bloc de dades arbitrari. Aquest número pot utilitzar-se per a detectar errors que puguin haver-se introduït durant la seva transmissió o emmagatzematge. La integritat de les dades es pot comprovar en qualsevol moment posterior tornant a calcular la suma de verificació i comparant-la amb l'emmagatzemada. Si les sumes de verificació coincideixen, és molt probable que les dades no hagin estat alterades.

El procediment que produeix la suma de verificació a partir de les dades es denomina **funció de suma de verificació** o **algorisme de suma de verificació**. Un bon algorisme de sumes de verificació llançarà un resultat diferent amb alta probabilitat quan les dades es corrompin accidentalment; si les sumes de verificació coincideixen, les dades tenen la mateixa alta probabilitat d'estar lliures d'errors accidentals.

Les funcions de suma de verificació es confonen sovint amb els *hash* o fins i tot amb els xifrats. Totes estan relacionades, però cadascuna té els seus propis usos i està dissenyada de manera diferent.

Existeixen diversos tipus i versions d'algorismes de verificació de dades que constitueixen una pràctica estàndard recomanada per a la detecció d'errors accidentals o intencionals esdevinguts en fitxers. Els algorismes criptogràfics són els més utilitzats per a aquesta tasca. Es basen en generar, a partir de les dades originals, un *hash* (empremta o resum digital), molt més curt que l'original, que identifiqui de manera unívoca les dades. El problema és que aquestes empremtes mai són unívoques, apareixen el que anomenam **col·lisions *hash***. Estadísticament, hi ha múltiples dades possibles que tenen el mateix *hash*.

L'algorisme de reducció criptogràfica **MD5** ha estat un dels algorismes més utilitzats en el passat. Es basa en generar un hash o empremta de 128 bits. Però ja s'ha aconseguit rompre aquest algorisme, podent introduir bits sense sentit en les dades de manera que produeixin la mateixa suma de verificació. Això significa que qualsevol intrús intern o extern podria ser capaç de substituir contingut digital valuós per dades sense sentit, sense que el sistema de revisió contra errors pogués detectar-ho. Per aquesta raó, MD5 encara pot ser útil en la transmissió de dades, però no per a informació emmagatzemada ni quan la seguretat sigui un factor capital.

Altres algorismes amb longituds de *hash* més grans són els de la família **SHA** (***Secure Hash Algorithm***). SHA-1 treballa amb 160 bits, però també ja s'ha demostrat vulnerable. SHA-2 és una evolució de SHA-1 i disposa de diferents versions amb longituds de 224, 256, 384 i 512 bits, i resulta computacionalment similar a SHA-1, però molt més difícil de rompre. El constant creixement de la potència computacional significa que, a llarg termini, la seguretat aportada per aquests algorismes i les seves sumes de verificació també estarà en risc.

```
$ md5sum quijote.txt
5ec9ea693fafdad2379b838f392b631e quijote.txt

$ sha256sum quijote.txt
5f110a42988f80e119de8aec4a0a1ec399e53c60b7027b63996c3e0458192f95 quijote.txt

$ sha512sum quijote.txt
5a3bfb9f283f940c9d383381eccb265a06f0e18a0ca4fe0b699d8500542
59cca0abdad5cfb8d94c70dbd2e95def0298bb4042f5bc764e8d9b223f61
d1e1fc2de quijote.txt
```

Un altre algorisme molt popular és el de comprovació per redundància cíclica o **CRC** (***cyclic redundancy check***). No entrarem tampoc en massa detalls tècnics, però CRC es basa en representar les dades originals com un polinomi i dividir-lo per un polinomi generador. El residu d'aquesta divisió és la cadena o suma de verificació. Depenent de la longitud del polinomi generador triat, tenim diferents versions de CRC: 9 bits (CRC-8), 17 bits (CRC-16), 33 bits (CRC-32) i 65 bits (CRC-64). Aquesta longitud del polinomi generador també determina la longitud de la suma de verificació, que serà d'un bit menys: 8 bits per a CRC-8, 16 bits per a CRC-16, 32 bits per a CRC-32 i 64 bits per a CRC-64.

3.2. Sumes de verificació en HDFS

HDFS utilitza sumes de verificació per a la integritat de les dades.

HDFS calcula les sumes de verificació (*checksums*) per a cada bloc de dades i finalment les emmagatzema en un arxiu ocult separat en el mateix espai de noms de HDFS. Internament, són els Data Nodes els encarregats de, quan reben un bloc de dades, generar la suma de verificació del bloc i emmagatzemar-la en el seu disc. A més, cada Data Node fa comprovacions periòdiques de les sumes de verificació i manté un registre persistent on guarda per a cada bloc de dades, a més de la suma de verificació, quan s'ha fet la darrera comprovació.

HDFS utilitza **CRC-32** (comprovació de redundància cíclica de 32 bits) com a algorisme de suma de verificació per defecte, ja que té una longitud de 32 bits i una sobrecàrrega d'emmagatzematge inferior a l'1%.

De la mateixa manera, HDFS comprova les sumes de verificació mentre un client llegeix les dades dels Data Nodes. Si hi ha alguna discrepància o error en el valor de la suma de verificació, es llança una excepció CheckSumException al client durant la recuperació de les dades per al seu processament.

Vegem un exemple d'error de suma verificació:

```
hadoop fs –copyFromLocal ~/Desktop/dtlScaleData/attr.txt
/tmp/hadoop-name/dfs/data/attr2.txt
13/03/15 15:02:51 INFO util.NativeCodeLoader: Loaded the native-hadoop library
13/03/15 15:02:51 INFO fs.FSInputChecker: Found checksum error: b[0, 0]=
org.apache.hadoop.fs.ChecksumException: Checksum error: /home/name/attr.txt at 0
at org.apache.hadoop.fs.ChecksumFileSystem\$ChecksumFSInputChecker.
readChunk(ChecksumFileSystem.java:219)
at org.apache.hadoop.fs.FSInputChecker.readChecksumChunk(FSInputChecker.java:237)
at org.apache.hadoop.fs.FSInputChecker.read1(FSInputChecker.java:189)
at org.apache.hadoop.fs.FSInputChecker.read(FSInputChecker.java:158)
at java.io.DataInputStream.read(DataInputStream.java:100)
at org.apache.hadoop.io.IOUtils.copyBytes(IOUtils.java:68)
...
at org.apache.hadoop.fs.FileUtil.copy(FileUtil.java:176)
at org.apache.hadoop.fs.FileSystem.copyFromLocalFile(FileSystem.java:1183)
at org.apache.hadoop.fs.FsShell.copyFromLocal(FsShell.java:130)
at org.apache.hadoop.fs.FsShell.run(FsShell.java:1762)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:65)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:79)
at org.apache.hadoop.fs.FsShell.main(FsShell.java:1895)
copyFromLocal: Checksum error: /home/name/attr.txt at 0
```

En tot cas, es pot desactivar la comprovació de la suma de verificació en enviar un treball al clúster.

4. Què és el monitoratge?

El **monitoratge** és un dels aspectes més importants de qualsevol sistema d'informació, ja que permet mesurar i garantir la disponibilitat. Un sistema sense monitoratge no garanteix que compleixi el seu objectiu. Per això és necessari controlar de manera automàtica que els sistemes estiguin funcionant i puguin reaccionar de manera ràpida davant les fallades. El monitoratge no sols millora la disponibilitat, també proporciona informació a nivell operatiu, permetent realitzar estimacions de temps i recursos. Aquests són els motius pels quals és necessari monitorar tots els aspectes d'un clúster, des de l'entorn físic on es troba, fins als serveis dels quals depèn.

Els sistemes bàsics que s'han de monitorar són:

- Dades en temps real i l'històric: És necessari que es pugui visualitzar de manera gràfica la informació rellevant del sistema, a més s'ha de disposar d'un històric on s'emmagatzemin les dades i es puguin consultar
- Gestió d'esdeveniments i alarmes: S'han de definir esdeveniments com un ús excessiu d'una CPU o una baixada en la velocitat de la connexió.

El mesurament del rendiment d'un sistema és una activitat clau en el desenvolupament i manteniment d'aquest. Amb la infraestructura de maquinari i programari adequada, necessitam ser capaços de recollir informació en temps real de paràmetres com ara: temperatura (ambiental i dels nodes), humitat, utilització de recursos (xarxa, disc, memòria), etc.

Els tipus de monitoratge són:

- Actius: Des d'un sistema de monitoratge es comprova l'estat i mètriques dels dispositius.
- Passius: El sistema que està essent monitorat envia una alarma quan succeeix un esdeveniment que ha de ser atès.

4.1. Objectius del monitoratge

Els **objectius principals** d'un monitoratge són:

- **Correlació entre diferents fonts de dades:** El monitoratge de sistemes Big Data pot cobrir un ampli rang de fonts de dades potencials. Aquí es poden incloure gran quantitat de components dels sistemes amb una gran varietat de programari per a la computació de nodes, *switches*, arxius de sistema... Les dades recopilades a partir d'aquestes fonts han de ser correlacionades per a poder proporcionar una visió del comportament del sistema. Això requereix la captura d'una gran quantitat de dades de fonts molt dispars, amb comportaments diferents i amb interfícies molt variades.
- **Gestió de les dades i anàlisis en el temps:** En general les plataformes Big Data tendeixen a durar múltiples anys, i la cerca de tendència en el seu comportament pot ser de gran importància per a entendre i prevenir el seu comportament en el temps. A causa de la gran quantitat de dades de monitoratge que es produeixen en aquestes plataformes, l'escala, gestió i emmagatzematge d'aquesta informació és un factor de gran importància per a realitzar aquestes anàlisis de tendència.
- **Captura d'informació rellevant:** Després de realitzar les anàlisis de les dades i correlació, es pot determinar que certs resultats proporcionen informació més rellevant sobre l'estat i la salut d'un clúster. Aquests tipus d'anàlisis permetran reduir els requisits d'emmagatzematge de dades, a causa de l'eliminació de reports poc rellevants.

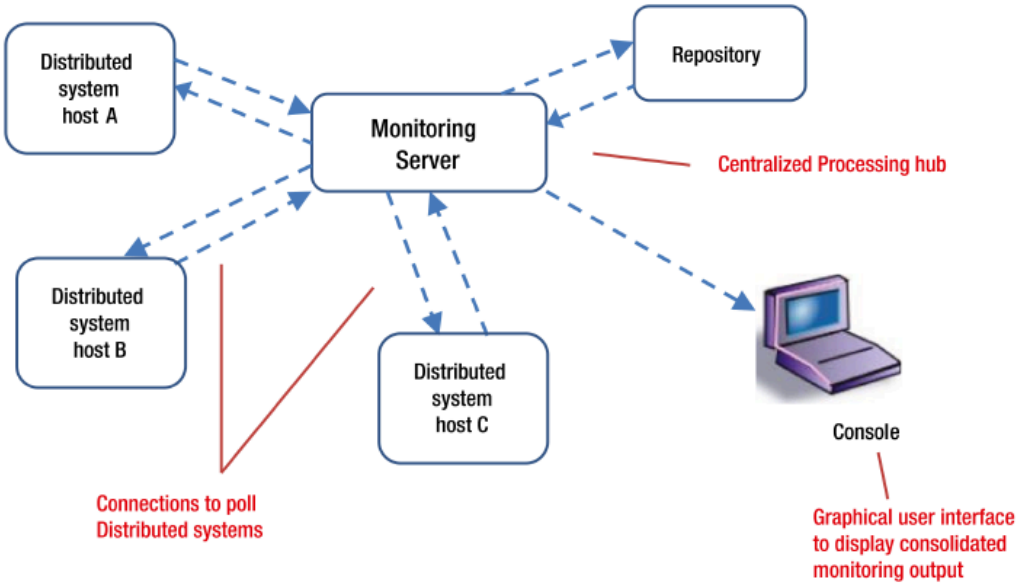
En arquitectures Big Data la complexitat creix exponencialment, així com el cost dels equips, i el seu manteniment. A causa d'això, existeixen diferents **objectius secundaris** del monitoratge, relacionats amb el cost econòmic del clúster:

- **Utilització de recursos i optimització del repartiment de càrrega:** Per a això és necessari obtenir informació efectiva dels recursos crítics constants per a diferents tipus de treball.
- **Predir la tolerància a fallades i predir fallades de maquinari:** Normalment a causa de la gran quantitat de components en els clústers, el temps mitjà entre fallades o *Mean Time Between Failures (MTBF)* disminueix amb rapidesa. Algunes aplicacions del món real com els estudis de la dinàmica de molècules poden durar des d'algunes hores fins a diversos dies per a completar la seva execució. A causa d'això és normal que sorgeixin diferents fallades en una execució. La presa de dades precises per a la detecció i previsió d'aquestes fallades pot ajudar a millorar el sistema en l'execució de futures aplicacions.
- **Millorar l'eficiència energètica:** El consum energètic és un dels factors clau a l'hora de dissenyar un clúster, així com el desenvolupament de programari que compleixi requisits d'energia específics. Una de les recomanacions per a millorar l'eficiència és tenir poca diversitat d'interfícies de mesurament d'energia i potència, de diferents venedors, ja que aquestes poden diferir entre venedors, i podria afectar els processos de mesurament.
- **Notificació d'anomalies i errors:** Mitjançant les alarmes es pot avisar a l'usuari que alguna cosa no funciona correctament o està a punt d'arribar a ser una fallada.
- **Coneixement del rendiment:** A causa de la complexitat dels sistemes, és necessari poder comprovar l'estat actual del sistema i si el rendiment és l'esperat.
- **Millorar la gestió de logs:** Els logs són útils per a trobar errors i fallades. Per tant, si es disposa d'un sistema centralitzat de logs és més senzill veure per què ha fallat el sistema.
- **Optimització de sistemes:** Amb les dades generades a partir de monitorar un sistema, es poden realitzar optimitzacions futures del sistema.
- **Detectar l'origen dels incidents:** El monitoratge dels sistemes permet detectar quin component del sistema falla.
- **Planificació de creixement:** Mitjançant l'ús de sistemes monitoratge, es pot detectar els elements que han de millorar.

4.2. Sistema Simple de Monitoratge

Un sistema de monitoratge senzill necessita quatre components clau: un **servidor o procés coordinador**, **connexions per a sondejar *els hosts*** del sistema distribuït i recopilar la informació necessària, un **repositori** per a emmagatzemar la informació recopilada i una **interfície gràfica d'usuari** com a *front-end*.

El servidor de monitoratge consolida l'entrada rebuda mitjançant el sondeig dels *hosts* del sistema distribuït i escriu la sortida detallada (així com resumida) en un repositori. Una consola proporciona opcions de visualització per a les dades que es poden resumir utilitzant diversos paràmetres, com l'esdeveniment de monitoratge, el servidor, el tipus d'alerta, etc.



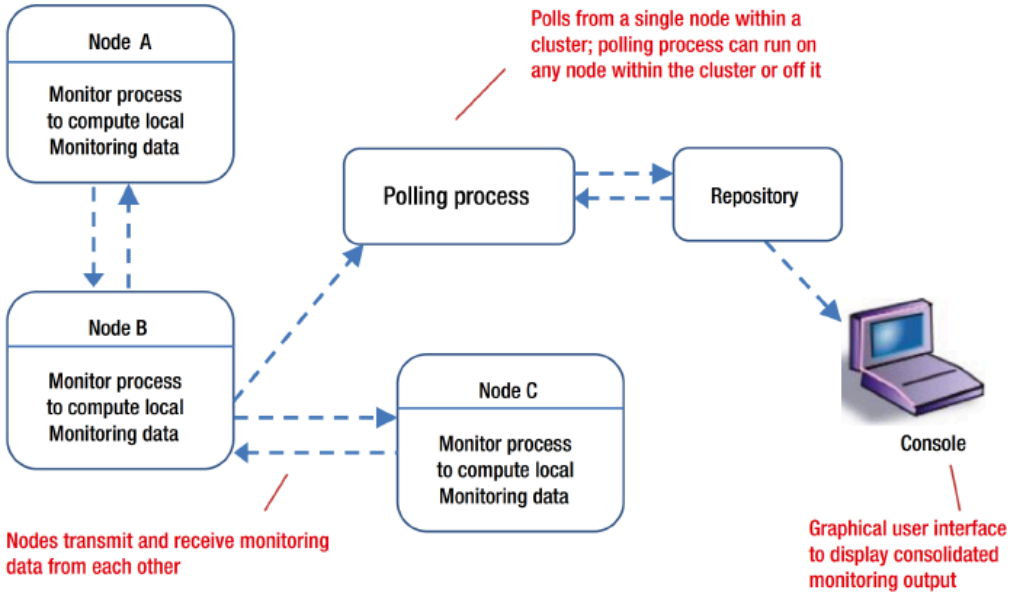
Imatge: *Arquitectura simple de monitoratge. Font: Universitat de Castella-La Mancha*

Una arquitectura de sistema de monitoratge simple com aquesta no és escalable. Si fos necessari monitorar milers de *hosts*, el servidor de monitoratge hauria de gestionar milers de connexions, processar i consolidar la sortida, i presentar-la en la consola en uns pocs segons. A més, cada *host* que s'afegeixi al sistema, suposa que la càrrega del servidor de monitoratge augmenti. En realitat, després d'un determinat número de *hosts*, el sistema arribarà a no ser capaç d'afegir-ne més. A més, el gran volum de sondejos augmentaria el trànsit de la xarxa i afectaria el rendiment general del sistema. A això cal afegir les complexitats d'un clúster Hadoop, en el qual cal tenir en compte el paper d'un node mentre es consoliden les dades per a ell, així com resumir les dades per a múltiples nodes amb el mateix rol.

4.3. Sistema de Monitoratge en Hadoop

Un sistema de monitoratge simple segueix la mateixa disposició de processament que el disseny tradicional client-servidor: un únic servidor de monitoratge centralitzat realitza tot el processament, i a mesura que el número de hosts augmenta, també ho fa la càrrega de processament. El trànsit de xarxa també és important a nivell de càrrega, ja que les dades sondejades dels hosts es consoliden en el servidor de monitoratge.

Igual que l'arquitectura distribuïda de Hadoop suposa una notable millora de l'eficiència respecte al processament tradicional client-servidor, un model de processament distribuït pot millorar un sistema de monitoratge simple. En Hadoop ja no hi ha un servidor centralitzat que es converteixi en un coll de botella de processament o en un punt únic de fallada. Cada node és un participant actiu realitzant part del processament en paral·lel. Cadascun d'aquests processos localitzats pot transmetre dades a altres nodes en el clúster i també rebre còpies de dades d'altres nodes en el clúster. Un procés de sondeig pot obtenir les dades de monitoratge de tot el clúster des de qualsevol dels nodes del clúster amb una freqüència predeterminada. Les dades es poden escriure en un repositori i emmagatzemar per al seu posterior processament o mostrar-se mitjançant un *front-end* gràfic o basat en web.



Imatge: *Arquitectura distribuïda de monitoratge. Font: Universitat de Castella-La Mancha*

Amb aquesta arquitectura, fins i tot afegint 1000 hosts, el rendiment del procés de monitoratge/supervisió no es veuria afectat. Per exemple, el procés de sondeig pot continuar sondejant des de qualsevol dels nodes i no ha de fer múltiples connexions. D'altra banda, els nodes del clúster transmeten dades a un canal comú que és rebut per tots els altres nodes, per la qual cosa la sobrecàrrega seria inapreciable.

Per tant, l'augment del nombre de nodes no afecta al procés de sondeig ni al rendiment del sistema, cosa que fa que l'arquitectura sigui altament escalable. En comparació amb els sistemes de monitoratge tradicionals, l'únic treball extra que cal fer és aplicar la configuració del procés de monitoratge a tots els nodes.

En resum, els processos de monitoratge en els nodes individuals calculen "dades de monitoratge locals". Les dades de monitoratge han de ser calculades localment; perquè Hadoop és un sistema distribuït de múltiples nodes en el qual les dades es distribueixen en els seus nombrosos DataNodes i segons la filosofia de Hadoop de "portar el processament a dades", les dades es processen localment, és a dir, a on resideixen, en els DataNodes.

5. Mètriques de monitoratge

Les **mètriques** capturen un valor pertanyent al sistema en un punt específic en el temps, per exemple, el nombre d'usuaris connectats a una aplicació web. Per tant, les mètriques se solen recollir una vegada per segon, una per minut o en un altre interval regular per a supervisar un sistema al llarg del temps.

Hi ha dues categories importants de mètriques: **mètriques de treball** i **mètriques de recursos**. És necessari, per a cada component de la infraestructura, determinar quines mètriques de cadascun dels tipus estan disponibles i, a continuació, establir els procediments necessaris de recol·lecció.

A més de les mètriques, que es recullen de manera més o menys contínua, alguns sistemes de monitoratge també poden capturar **esdeveniments**: successos discrets i poc freqüents que poden proporcionar un context crucial per a entendre què ha canviat en el comportament del sistema.

Veurem els dos tipus principals de mètriques i els esdeveniments en els subpartats següents. A continuació, introduirem les característiques principals de les dades recollides per un sistema de monitoratge. I acabarem l'apartat veient quines són les mètriques principals d'un clúster Hadoop.

5.1. Mètriques de Treball

Les mètriques de treball indiquen l'estat de salut del sistema mesurant el seu rendiment útil. Se solen desglorar en quatre subtipus:

- ***throughput***: La productivitat és la quantitat de treball que el sistema realitza per unitat de temps. Sol registrar-se com un número absolut.
- ***success***: Les mètriques d'èxit representen el percentatge de treball que s'ha executat amb èxit.
- ***error***: Les mètriques d'error capturen el nombre de resultats erronis, normalment expressats com una taxa d'errors per unitat de temps o normalitzats pel rendiment per a obtenir errors per unitat de treball. Les mètriques d'error sovint es capturen per separat de les mètriques d'èxit quan hi ha diverses fonts potencials d'error, algunes de les quals són més greus o procesables que d'altres.
- ***performance***: Les mètriques de rendiment quantifiquen l'eficiència amb la qual un component fa el seu treball. La mètrica de rendiment més comú és la **latència**, que representa el temps necessari per a completar una unitat de treball. La latència pot expressar-se com una mitjana o com un percentil.

Aquestes mètriques són molt importants per a l'observabilidad del sistema, atès que permeten respondre ràpidament a preguntes com: està el sistema disponible i fent activament allò per al que va ser construït? Amb quina rapidesa està produint treball? Quina és la qualitat d'aquest treball?

5.2. Mètriques de Recursos

La majoria dels components de la infraestructura de programari poden considerar-se com a recursos. Alguns recursos són de baix nivell; per exemple, la CPU, la memòria, els discos i les interfícies de xarxa. Però un component de nivell superior, com una base de dades, també pot considerar-se un recurs, si un altre sistema el necessita.

Les mètriques de recursos pretenen oferir una imatge detallada de l'estat d'un sistema, i per això, són molt valuoses per a l'anàlisi i la diagnosi de problemes. Les mètriques a recopilar de cada recurs es poden organitzar en quatre àrees clau:

- ***utilization***: La utilització és el percentatge de temps en què el recurs està ocupat, o el percentatge de la capacitat del recurs que està en ús.
- ***saturation***: La saturació és una mesura de la quantitat de treball sol·licitat que el recurs encara no pot atendre, sovint en cua.
- ***errors***: Els errors representen errors interns que poden no ser observables en el treball que produeix el recurs.
- ***availability***: La disponibilitat representa el percentatge de temps que el recurs ha estat responent a les sol·licituds. Aquesta mètrica només està ben definida per als recursos la disponibilitat dels quals es pot comprovar de manera activa i regular.

5.3. Esdeveniments

Com ja hem dit anteriorment, entenem per esdeveniments aquells successos discrets i poc freqüents que poden proporcionar un context que ens ajuda a entendre què ha canviat en el comportament d'un sistema.

Alguns exemples d'esdeveniments rellevants són els següents:

- Canvis: Llançaments de codi intern o fallades de construcció
- Alertes: Alertes generades internament o notificacions de tercers
- Escalat: Addició o eliminació de *hosts*.

Un esdeveniment sol contenir suficient informació perquè pugui ser interpretat per si mateix, a diferència d'un punt de dades d'una mètrica, que generalment només té sentit en el seu context. Els esdeveniments capturen el que ha ocorregut, en un moment donat, amb informació addicional opcional.

Els esdeveniments s'utilitzen a vegades per a generar alertes, per exemple, algú ha de ser notificat quan un treball crític ha fallat. A més, també s'utilitzen per a analitzar problemes i correlacionar-los entre sistemes. En general, són dades valuoses que s'han de recollir sempre que sigui possible.

5.4. Característiques de les Dades

Les dades recollides per un sistema de monitoratge han de tenir les següents característiques:

- **Comprensibles:** S'ha de poder determinar ràpidament com es va capturar cada mètrica o esdeveniment i què representa.
- **Freqüents:** Si es recullen mètriques amb molt poca freqüència, o bé si es fa una mitjana dels valors durant llargs períodes de temps, es pot perdre la capacitat de reconstruir amb precisió el comportament del sistema. Per exemple, els períodes d'utilització del 100% dels recursos quedaran ocults si es fan una mitjana amb períodes de menor utilització. Per tant, s'hauran de recollir les mètriques de cada sistema amb una freqüència que no oculti els problemes, sense recollir-les amb tanta freqüència que el monitoratge es converteixi en una càrrega perceptible per al sistema (***efecte observador***) o creï soroll en les dades del monitoratge mitjançant el mostreig d'intervals de temps que són massa curts per a contenir dades significatives.
- **Llarga durada:** És necessari conservar les dades en brut (*raw*), sense agregar, durant un llarg període de temps, per a així disposar de material suficient per a alimentar els sistemes d'anàlisi automàtica. Conservar les dades sense processar durant un període de temps significatiu fa que sigui molt més fàcil saber quin és el comportament "normal" del sistema, especialment si les mètriques tenen variacions mensuals, estacionals o anuals.

5.5. Mètriques de Hadoop

Des del punt de vista de les operacions, els clústers de Hadoop són increïblement resistents a les fallades del sistema. El disseny de Hadoop és l'adequat per a tenir una eficient tolerància a fallades i gestionar la caiguda de *racks* sencers.

Quan funciona correctament, un clúster de Hadoop pot manejar una quantitat realment massiva de dades: hi ha molts clústers de producció que gestionen petabytes de dades. La supervisió de cadascun dels subcomponentes de Hadoop és essencial per a mantenir els treballs en marxa i el clúster en funcionament.

Les mètriques de Hadoop (**Hadoop Metrics**) són col·leccions d'informació estadística que els procesos (dimonis) de Hadoop recullen i exposen i que suposen la base per al monitoratge, diagnosi de problemes i optimització del rendiment dels clústers Hadoop. Hi ha moltes mètriques disponibles per defecte.

AMPLIACIÓ

Podeu trobar el detall de totes les mètriques disponibles en la documentació de Hadoop.

Podeu trobar també la documentació detallada de Metrics 2.0, el paquet Java que gestiona les mètriques de Hadoop.

Totes aquestes mètriques es poden visualitzar mitjançant les diverses interfícies web dels dimonis de Hadoop (ho veurem en detall en l'apartat 6.1) i també poden ser sondejades per eines externes de monitorització com Ganglia, Nagios o Apache Ambari.

En els següents apartats ens centrarem en les mètriques relacionades amb els tres components principals del nucli de Hadoop: **HDFS**, **MapReduce** i **YARN**.

5.6. Mètriques relacionades amb HDFS

Aquestes són mètriques a nivell de host específiques d'un DataNode en particular.

Remaining: Un únic DataNode que es quedi sense espai podria provocar ràpidament fallades en tot el clúster, ja que les dades s'escriuen en un grup cada vegada més reduït de DataNodes disponibles. El seguiment d'aquesta mètrica al llarg del temps és essencial per a mantenir un clúster en un funcionament òptim.

Serà necessari alertar sobre la capacitat restant del DataNode quan caigui a menys del 10 %.

NumFailedVolumes: Per defecte, un sol volum que falli en un DataNode farà que tot el node es desconnecti. Quan un DataNode es desconnecta, el NameNode ha de copiar qualsevol bloc infra-replicat que s'hagi perdut en aquest node, causant un increment gran de trànsit de xarxa i una potencial degradació del rendiment. Per a evitar una allau d'activitat de xarxa per la fallada d'un sol disc s'ha d'establir el nivell tolerat de fallada de volum en la configuració d'un nivell acceptable (*N*) en la propietat dfs.datanode.failed.volumes.tolerated:

```
<property>
  <name>dfs.datanode.failed.volumes.tolerated</name>
  <value>N</value>
</property>
```

5.7. Mètriques relacionades amb MapReduce

El marc de monitoratge de MapReduce es basa en una sèrie de mètriques de comptatge que permeten rastrejar les estadístiques de l'execució de treballs de MapReduce. Els comptadors són un mecanisme que permet veure el que realment està succeint durant l'execució d'un treball MapReduce proporcionant un recompte del nombre de vegades que ha ocorregut un esdeveniment determinat. Aquesta informació serà molt útil per als desenvolupadors d'aplicacions ja que podrien analitzar colls de botella i identificar potencials optimitzacions en les seves aplicacions. Els comptadors de MapReduce es poden dividir en dues categories principals: els **comptadors de treballs** (***job counters***) i els **comptadors de tasques** (***task counters***). També podem definir els nostres propis comptadors (***custom counters***).

Comptadors de treball

Els podem trobar a org.apache.hadoop.mapreduce.JobCounter

- ***MILLIS_MAPS/MILLIS_REDUCES:*** Aquestes mètriques rastregen el temps en totes les tasques *de map* i *reduce*. La seva utilitat completa es realitza quan es rastreja juntament amb el comptador de tasques *GC_TIME_MILLIS*; quan es prenen en conjunt, és possible determinar el percentatge de temps gastat en la recol·lecció de fems. Utilitzant els tres comptadors, podem calcular el percentatge de temps dedicat a la recol·lecció de fems amb la següent fórmula:

GC_TIME_PERCENTAGE = GC_TIME_MILLIS / (MILLIS_MAPS + MILLIS_REDUEIXES)

Els treballs en els quals la recol·lecció de fems consumeix un percentatge significatiu del temps de computació (per exemple, més del 20 %) poden beneficiar-se d'augmentar la quantitat del *heap* disponible per a ells.

- ***NUM_FAILED_MAPS/NUM_FAILED_REDUCES:*** Aquest parell de mètriques rastreja el nombre de tasques *map/reduce* fallides per a un treball. Les tasques poden fallar per moltes raons: la sortida interrompuda de la JVM, els errors en temps d'execució i les tasques penjades són algunes de les més comunes. Les tasques fallides són toleradas fins a aconseguir el valor establert per les següents propietats:
  - mapreduce.map.maxattempts
  - mapreduce.reduce.maxattempts

Si una tasca falla el màxim d'intents, es marca com a fallida.

- ***DATA_LOCAL_MAPS / RACK_LOCAL_MAPS / OTHER_LOCAL_MAPS:*** Aquests comptadors registren el nombre de tasques de *map* que s'han realitzat, agregades per ubicació. Les dades poden estar situades en el node que executa la tasca *map*, en un node del mateix *rack* que el node que fa la tasca de *map* o en un node situat en un *rack* diferent en un altre lloc del clúster. Els beneficis d'operar amb dades locals inclouen la reducció de la sobrecàrrega de la xarxa i una execució més ràpida. Aquests avantatges es perden quan el càlcul s'ha de realitzar en un node que no està físicament a prop de les dades que es processen. Atès que l'enviament de dades a través de la xarxa porta temps, correlacionar el baix rendiment del treball amb aquests comptadors pot ajudar a determinar per què s'ha ressentit el rendiment. Si moltes operacions *map* s'haguessin d'executar en nodes on les dades no estiguessin allotjades localment, seria raonable esperar una degradació del rendiment.

ALERTA

Si moltes tasques fallen en un DataNode en particular, el problema podria estar relacionat amb el maquinari.

Un treball fallarà íntegrament si més del percentatge de ***mapreduce.map.failures.maxpercent*** de les tasques *map* en el treball fallen o més del percentatge de ***mapreduce.reduce.maxpercent*** de les tasques *reduce* fallen.

Comptadors de tasques

Els podem trobar a org.apache.hadoop.mapreduce.TaskCounter

- ***REDUCE_INPUT_RECORDS:*** Vigilar el nombre de registres d'entrada per a cada tasca de *reduce* en un treball serveix per a identificar problemes de rendiment relacionats amb un conjunt de dades esbiaixades (*sesgadas* en castellà). Les dades esbiaixades es caracteritzen per una distribució desproporcionada dels registres d'entrada entre totes les operacions reduce d'un treball. Els conjunts de dades esbiaixades poden fer que els treballs tardin molt més a completar-se de l'esperat.
- ***SPILLED_RECORDS:*** Les dades de sortida de les operacions *map* s'emmagatzemen en memòria, però quan *el buffer* s'omple, les dades es bolquen al disc. Un fil separat es genera per a fusionar totes les dades bolcades en un únic arxiu ordenat, que després es farà servir per les operacions *reduce*. Els bolcats al disc s'han de minimitzar, ja que provocarien un augment de l'activitat en el disc. El seguiment d'aquesta mètrica pot ajudar a determinar si és necessari realitzar canvis en la configuració (com augmentar la memòria disponible per a les tasques *map*) o si és necessari un processament addicional (compressió) de les dades d'entrada.
- ***GC_TIME_MILLIS:*** Útil per a determinar el percentatge de temps que les tasques passen realitzant la recol·lecció de fems.

Comptadors personalitzats

MapReduce permet als usuaris implementar comptadors personalitzats (*custom counters*) que són específics per a la seva aplicació. Els comptadors personalitzats es poden utilitzar per a fer un seguiment més detallat dels recomptes. Hi ha un límit pràctic per al nombre de comptadors a emprar, limitat per la quantitat de memòria física disponible. Hadoop té un valor de configuració, ***job.counters.max***, que limita el nombre de comptadors personalitzats que es poden crear; per defecte és 120.

5.8. Mètriques relacionades amb YARN

Les mètriques de YARN es poden agrupar en tres categories diferents: de clúster, d'aplicació o del Node Manager. Vegem-les a continuació.

Mètriques de clúster

Les mètriques del clúster proporcionen una visió d'alt nivell de l'execució de les aplicacions de YARN. Vegem les més importants.

- ***unhealthyNodes:*** YARN considera com a no saludable qualsevol node amb una utilització de disc que excedeixi el valor especificat en la configuració de YARN (propietat *yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage*). Comptar amb l'espai suficient en en disc és fonamental per a garantir el funcionament d'un clúster Hadoop, i un gran nombre de nodes no saludables (*unhealthy*) ha de ser ràpidament resolt. Una vegada que un node és marcat com a *unhealthy*, altres nodes han d'omplir el buit, el que pot causar una cascada d'alertes a mesura que un node darrere l'altre s'acosta a la utilització total del disc.
- ***activeNodes/lostNodes:*** La mètrica *activeNodes* rastreja el recompte actual de nodes actius, o que funcionen normalment. Aquesta mètrica hauria de romandre estàtica en absència d'un manteniment preventiu. Si un gestor de nodes no manté el contacte amb el gestor de recursos, finalment es marcarà com a "perdut" i els seus recursos deixaran d'estar disponibles per a la seva assignació. El llindar de temps d'espera és configurable (al fitxer *yarn.resourcemanager.nm.liveness-monitor.expire-interval-ms en yarn-site.xml*), amb un valor per defecte de 10 minuts (600000 ms). Els nodes poden perdre el contacte amb el ResourceManager per diverses raons, incloent problemes de xarxa, fallades de maquinari o un procés penjat.
- ***appsFailed:*** Si el percentatge de tasques *map* o *reduece* fallides supera un llindar específic (propietats *mapreduce.map.failures.maxpercent* i *mapreduce.reduce.failures.maxpercent* respectivament), l'aplicació en el seu conjunt fallarà.
- ***totalMB/allocatedMB:*** Ofereixen una visió d'alt nivell de l'ús de la memòria del clúster. Per a augmentar la memòria disponible per al clúster, es poden afegir nous nodes, ajustar la quantitat de memòria reservada per a les aplicacions YARN o canviar la quantitat mínima de RAM assignada als contenidors.

Mètriques d'aplicació

Les mètriques d'aplicació proporcionen informació detallada sobre l'execució de les aplicacions individuals de YARN. La mètrica més important d'aquest àmbit és ***progress***. El progrés proporciona una finestra en temps real de l'execució d'una aplicació YARN. El seu valor reportat sempre estarà en el rang de 0 a 1 (ambdós inclosos), amb un valor de 1 indicant una execució completa. Atès que l'execució de l'aplicació pot ser sovint opaca quan s'executen centenars d'aplicacions en milers de nodes, el seguiment del progrés juntament amb altres mètriques pot ajudar a determinar millor la causa de qualsevol degradació del rendiment.

Mètriques del NodeManager

Les mètriques del NodeManager proporcionen informació sobre els recursos a nivell de node individual. Des del punt de vista de YARN, la mètrica més important és ***containersFailed*** que registra el nombre de contenidors que no s'han pogut llançar en aquest NodeManager en particular. La causa més comuna d'aquesta fallada sol estar relacionada amb el maquinari, per exemple si el disc està ple.

6. Eines de monitoratge

El millor sistema de monitoratge per a un clúster Hadoop és aquell que s'adapti a l'entorn i les necessitats dels seus administradors i usuaris. Tant el monitoratge de l'ús dels recursos del sistema, com la notificació d'alertes immediata si es produeix un canvi sobtat en l'ús d'aquells, són funcionalitats que han d'incloure aquests sistemes de monitoratge.

Una primera aproximació al monitoratge de clústers Hadoop és mitjançant les interfícies que HDFS i YARN ens proporcionen. Amb aquestes eines ja podem recuperar algunes mètriques i detectar possibles problemes.

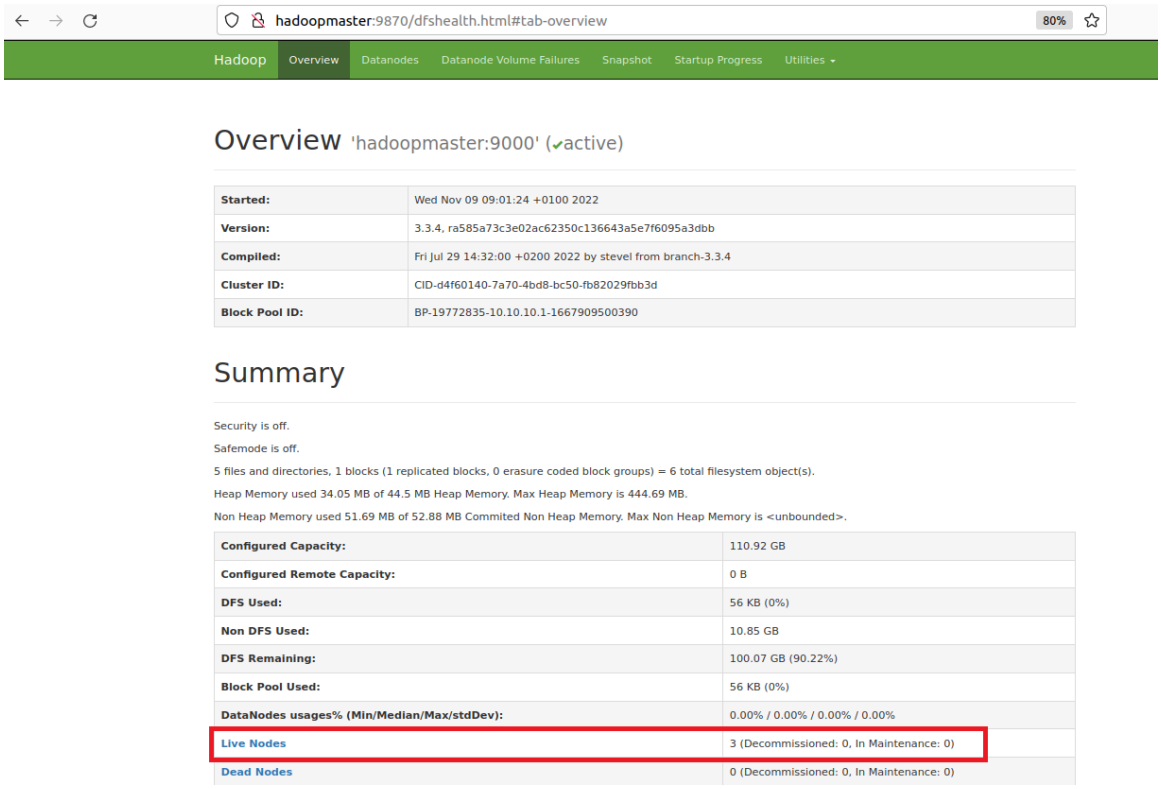
Un segon enfocament més professional és emprar eines generals de monitoratge com Ganglia o Nagios. Aquestes dues eines de codi obert i molt populars, ofereixen gran varietat de possibilitats per a supervisar els recursos del sistema, les connexions i qualsevol altra part d'un clúster que tècnicament es pugui monitorar. Ganglia presenta una alta eficiència en la recopilació de mètriques, el seguiment de les mateixes en el temps, i l'agregació dels resultats, mentre que Nagios se centra en proporcionar un mecanisme d'alertes. Atès que la recopilació de mètriques i les alertes són tots dos aspectes essencials del monitoratge, Ganglia i Nagios funcionen perfectament junts. En aquest curs ens centrarem en la recollida de mètriques mitjançant Ganglia, cosa que es fa mitjançant agents que s'executen en tots els hosts del clúster, que recullen informació a través d'un procés de sondeig (*pooling*).

Per últim, la tercera possibilitat és utilitzar Apache Ambari, una eina que forma part de l'ecosistema Hadoop, específicament dissenyada per a la gestió de clústers. En concret, Ambari permet el monitoratge dels clústers Hadoop, proporcionant una visió detallada de l'estat del clúster, incloent mètriques sobre CPU, memòria, disc i xarxa.

6.1. Eines internes de Hadoop

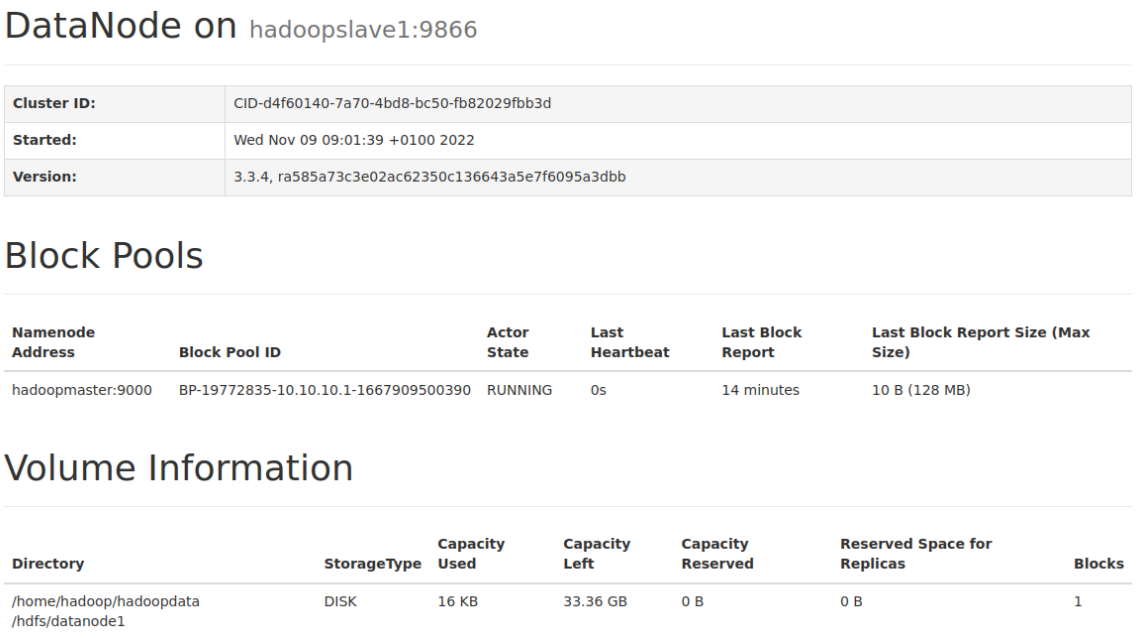
En lliuraments anteriors ja havíem vist breument com funcionen les interfícies web de gestió de HDFS i YARN en el nostre clúster Hadoop.

Així, en el lliurament 1 vàrem veure que en el port 9870 podíem accedir a la interfície del Name Node HDFS:



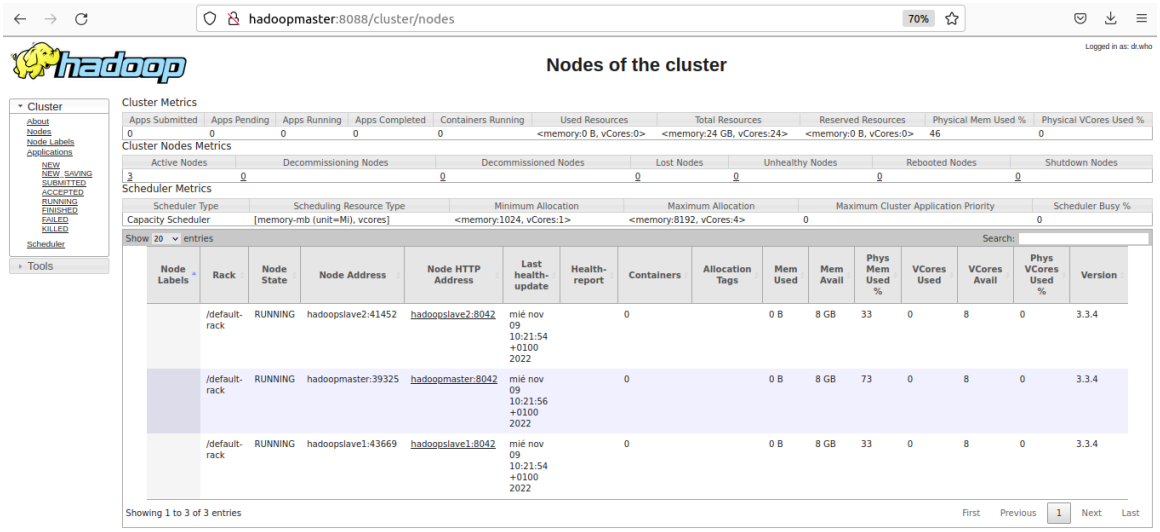
Imatge: *Interfície web del Name Node HDFS*

I que en el port 9864 tenim la interfície dels Data Nodes:



Imatge: *Interfície web del Data Node HDFS*

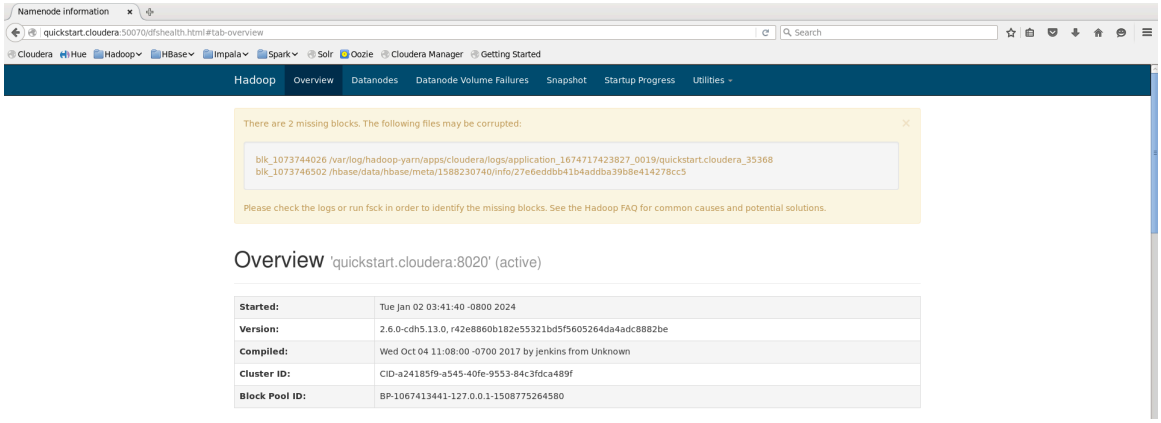
També vàrem veure que en el port 8088 podem trobar la interfície del ResourceManager de YARN, des de la qual podem accedir a la interfície del NodeManager de cada node (port 8042).



Imatge: *Interfície web de YARN (node mestre)*

6.2. Cas pràctic 1: eines internes en la Cloudera Quickstart VM

En aquest cas pràctic anam a treballar amb la màquina virtual Cloudera Quickstart. En aquest cas, tenim la interfície web del Name Node HDFS en el port 50070:



Imatge: *Interfície del Name Node HDFS en la Cloudera Quickstart VM*

Podem veure com, en aquest cas, ens està remarcant que tenim 2 blocs perduts, amb dos fitxers que poden estar corromputs. També veim que el dimoni del Name Node HDFS s'executa en el port 8020 i que està actiu.

Vegem ara el resum:

Summary

Security is off.

Safemode is off.

2,749 files and directories, 1,635 blocks = 4,384 total filesystem object(s).

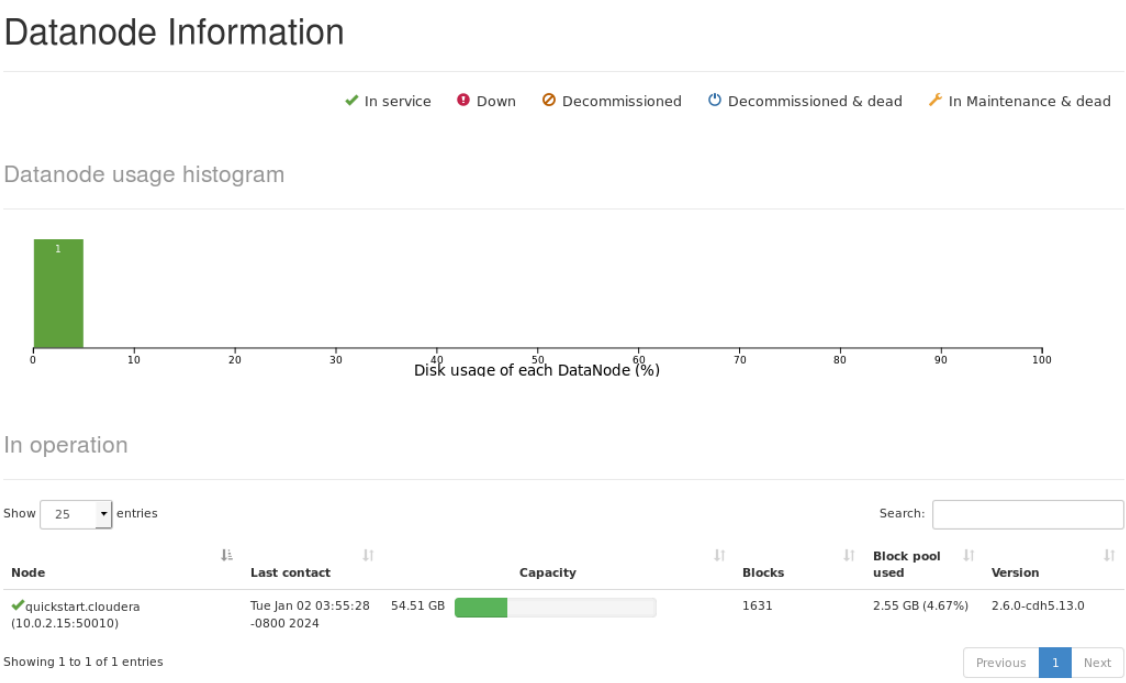
Heap Memory used 73.65 MB of 260.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 36.6 MB of 37.44 MB Commited Non Heap Memory. Max Non Heap Memory is 130 MB.

Configured Capacity:	54.51 GB
DFS Used:	2.55 GB (4.67%)
Non DFS Used:	11.49 GB
DFS Remaining:	37.45 GB (68.71%)
Block Pool Used:	2.55 GB (4.67%)
DataNodes usages% (Min/Median/Max/stdDev):	4.67% / 4.67% / 4.67% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	2
Number of Blocks Pending Deletion	0
Block Deletion Start Time	Tue Jan 02 03:41:40 -0800 2024
Last Checkpoint Time	Tue Jan 02 03:41:42 -0800 2024

Imatge: *Resum de l'estat de HDFS*

Si entram en la pestanya dels Datanodes, veurem la informació de l'únic data node del sistema, que veim que s'executa en el port 50010:



Imatge: *Informació del Data Node*

Molt rellevant, pel que fa al monitoratge, és la informació que ens retorna l'eina **JMX** respecte a la màquina virtual Java on s'executa Hadoop.

DEFINICIÓ

Java Management Extensions (JMX) és una tecnologia que forma part de la Java Platform, Standard Edition (Java SE platform), que proporciona una manera simple i estàndard de gestionar recursos tals com aplicacions, dispositius i serveis. Gràcies a que JMX és dinàmic, pot ser emprada per a monitorar i gestionar recursos així com es van creant, instal·lant i implementant. També es pot emprar JMX per a monitorar i gestionar la màquina virtual Java (JVM).

Utilitzant la tecnologia JMX, un recurs donat és instrumentat per un o més objectes Java coneguts com a Managed Beans (beans gestionats), o MBeans. Aquests MBeans estan registrats en un servidor d'objectes, conegut com a *MBean server*. El *MBean server* actua com a agent de gestió i pot executar-se en la majoria de dispositius que han estat habilitats per al llenguatge de programació Java.

Font: The Java Tutorials, Overview of the JMX Technology: https://docs.oracle.com/javase/tutorial/jmx/overview/index.html

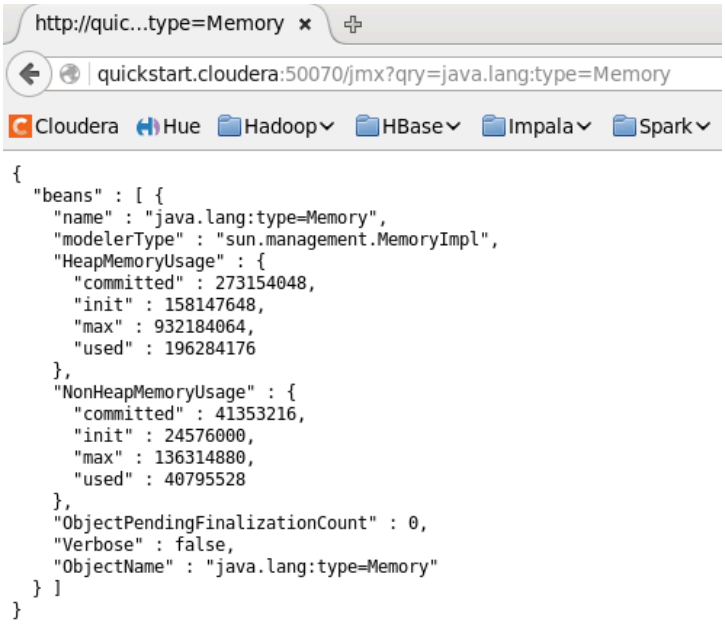
Podeu trobar més informació sobre la tecnologia JMX a: **https://www.oracle.com/java/technologies/javase/javamanagement.html**

Podem accedir a JMX des de **http://quickstart.cloudera:50070/jmx** :

Imatge: *JMX en la Cloudera Quickstart VM*

Com veim, JMX ens retorna un document JSON que conté un conjunt de MBeans (dins de l'array *beans*). Per exemple, podem veure que per al heap (espai emprat per a l'assignació dinàmica de memòria per als objectes Java en temps d'execució) s'han reservat 273 154 048 bytes (uns 260 MB), dels quals s'estan utilitzant 125 467 912 bytes (uns 120 MB).

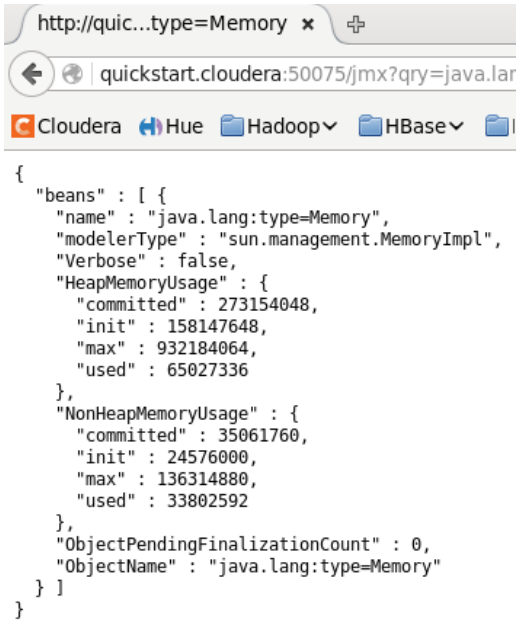
Podem filtrar els resultats de JMX. Per exemple, si volem només recuperar els beans amb java.lang:type=Memory, li passam la petició http://quickstart.cloudera:50070/jmx?qry=java.lang:type=Memory :



Imatge: *JMX amb una query*

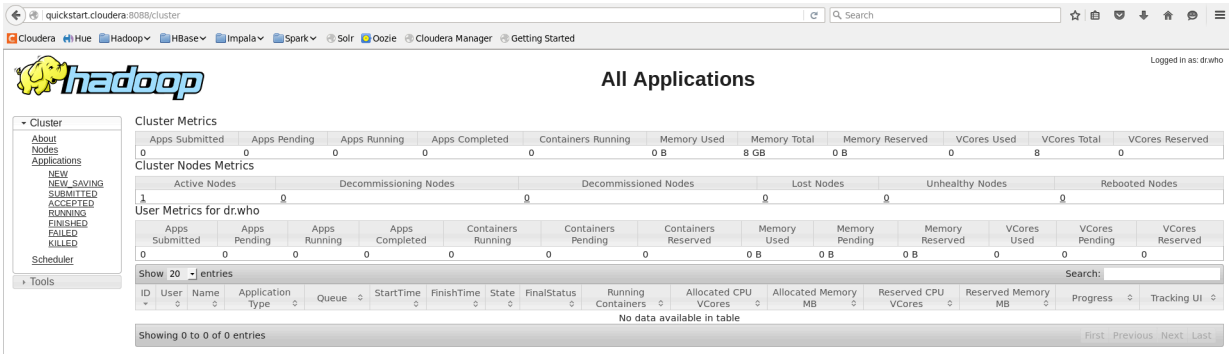
Veim que en aquest moments s'està emprant més memòria del heap que abans (l'espai reservat continua essent el mateix).

En el port 50075 podem accedir a la interfície web del Data Node. També hi podem emprar l'eina JMX. Per exemple, http://quickstart.cloudera:50075/jmx?qry=java.lang:type=Memory :



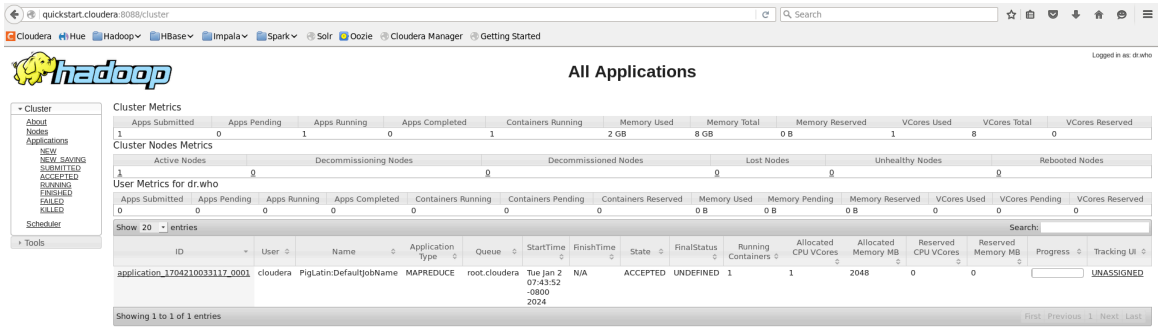
Imatge: *JMX en el Data Node*

Ens queda veure la interfície web de YARN. En http://quickstart.cloudera:8088/cluster podem accedir a la interfície del ResourceManager:



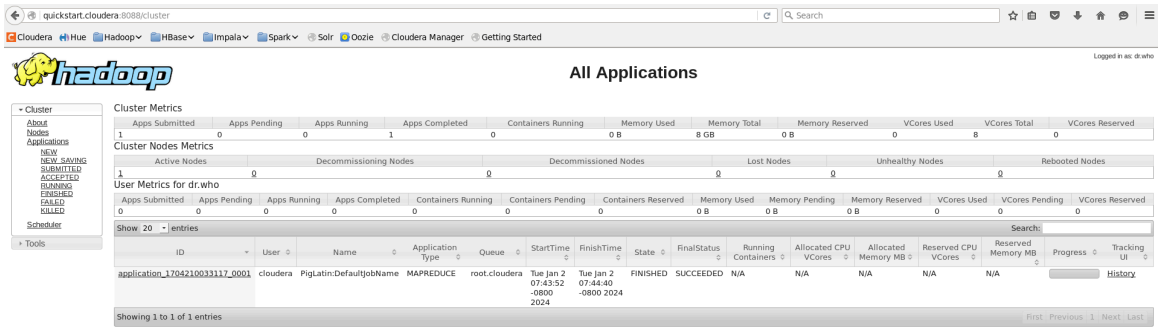
Imatge: *Interfície web del ResourceManager*

Podem veure que no hi apareix cap aplicació. Si ara llenç un treball amb Pig, vegem com hi apareix:



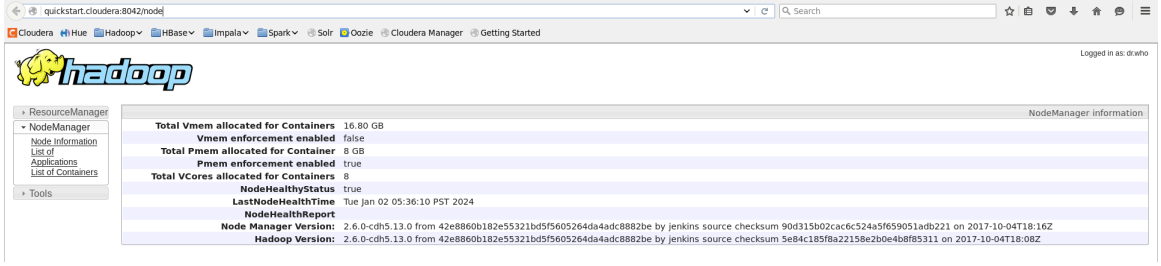
Imatge: *Interfície web del ResourceManager amb un treball Pig en marxa*

Veim que l'estat de l'aplicació és ACCEPTED, que l'estatus final encara és UNDEFINED i que se li ha assignat 2048 MB de memòria. Vegem-ho ara una vegada ha finalitzat l'execució del treball, on l'estat ha passat a ser FINISHED, l'estatus SUCCEEDED i ja no hi ha cap assignació de memòria:



Imatge: *Interfície web del ResourceManager amb un treball Pig finalitzada*

Per últim, en http://quickstart.cloudera:8042/node podem accedir a la interfície web del NodeManager:



Imatge: *Interfície web del NodeManager*

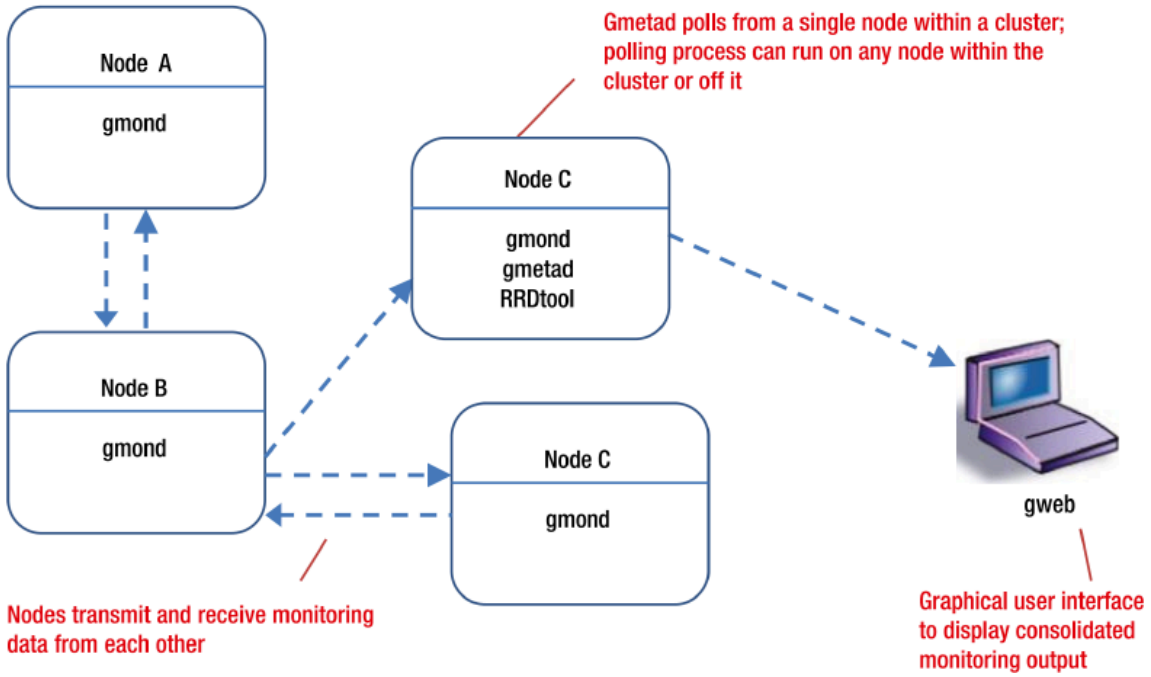
6.3. Ganglia

Ganglia va ser dissenyat en la Universitat de Califòrnia, Berkeley i va començar com un projecte de monitoratge de codi obert destinat a ser utilitzat amb grans sistemes distribuïts. L'arquitectura oberta de Ganglia facilita la integració amb altres aplicacions i recopilar estadístiques sobre les seves operacions. Ganglia pot rebre i processar mètriques de Hadoop amb facilitat i utilitzar-les eficaçment.

Per a un clúster monitorat, cada host executa un procés *daemon* anomenat **gmond** que recull i difon les dades locals de mètriques (com l'ús de la CPU, l'ús de la memòria, etc.) a tots els hosts dins del clúster. Un procés de sondeig **gmetad** pot consultar qualsevol dels hosts, llegir totes les dades de les mètriques i enviar-les a un servidor central de monitoratge. El host central pot mostrar les mètriques, agregar-les o resumir-les mitjançant una aplicació web anomenada **gweb**.

A grans trets, Ganglia té quatre components principals:

- gmond (Ganglia monitor deamon): s'executa en tots els nodes recollint les dades de les mètriques.
- gmetad (Ganglia meta data deamon): s'encarrega del sondeig de les dades de gmond.
- rrdtool: emmagatzema les dades que s'han sondejat mitjançant gmetad.
- gweb: visualització i anàlisi de les dades emmagatzemades.



Imatge: *Arquitectura de Ganglia per monitorar Hadoop*

A més, Ganglia també proporciona aquestes dues aplicacions de línia de comandes:

- gmetric: permet injectar mètriques personalitzades sobre els hosts que estan essent monitorats per Ganglia.
- gstat: permet executar queries sobre gmond per recuperar informació de mètriques.

Anam a veure a continuació cada un dels 4 components principals de Ganglia.

gmond

gmond s'ha d'instal·lar en cadascun dels hosts que es vulgui supervisar. Interactua amb el sistema operatiu del host per a adquirir mètriques de càrrega (per exemple, càrrega mitjana del node), de procés (per exemple, total de processos en execució) o de transferència (RPC) (per exemple, RpcAuthenticationFailures).

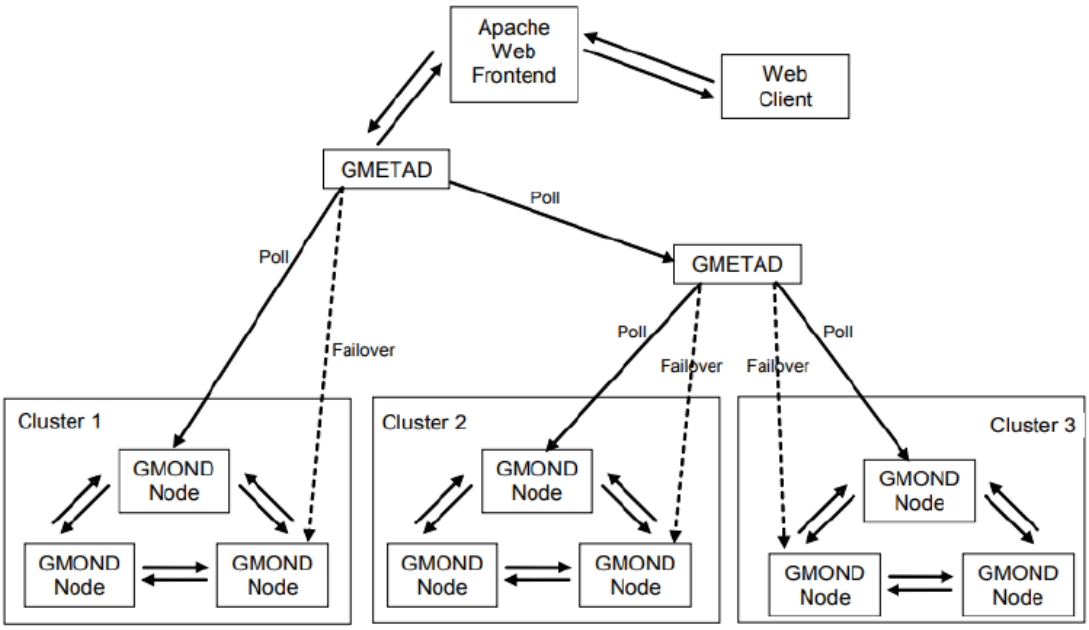
És modular i utilitza plugins específics del sistema operatiu per a realitzar els mesuraments. Atès que només s'instal·len els plugins necessaris en temps de compilació, gmond genera una sobrecàrrega insignificant.

Els mesuraments de gmond no s'invoquen a petició d'un motor de sondeig extern, sinó que es realitzen d'acord amb un calendari definit per un arxiu de configuració local. Les mètriques mesurades són emeses mitjançant una adreça de multidifusió (per defecte el port UDP 8649).

Cada host de gmond també registra les mètriques que escolta dels altres hosts dins del clúster. Per tant, cada host en un clúster de Ganglia coneix el valor actual de cada mètrica registrada per la resta de hosts en el mateix clúster. Aquesta és la raó per la qual només és necessari sondejar un host per clúster per a obtenir les mètriques de tot el clúster, i qualsevol fallada d'un host individual no afectarà el sistema en absolut. A més, aquest disseny redueix exponencialment el nombre de hosts que necessiten ser *enquestats*, i per tant és fàcilment escalable per a grans clústers.

gmetad

gmetad és el procés de sondeig dins del sistema de monitoratge de Ganglia. Necessita una llista de noms de hosts que especifiqui almenys un host per clúster. D'aquesta manera, gmetad pot sol·licitat a qualsevol gmond del clúster un bolcat en format XML de les mètriques de tot el clúster, a través del port 8649. gmetad és el responsable d'emmagatzemar les mètriques rebudes emprant RDDTool i de publicar-les en una interfície web (gweb).



Imatge: *Rol de gmetad en l'arquitectura de Ganglia*

RRDTool

RRDtool és el component de Ganglia que s'utilitza per a emmagatzemar les dades de les mètriques sondejades per gmetad des de qualsevol dels hosts del clúster. Les mètriques s'emmagatzemen en forma de round-robin, és a dir, quan no queda espai per a emmagatzemar nous valors, els antics se sobreescriuen.

D'acord amb els requisits de retenció de dades, RRDtool agrega i emmagatzema els valors de les dades. Aquesta forma d'emmagatzematge de dades permet analitzar ràpidament les dades recents, així com mantenir anys de dades històriques utilitzant una petita quantitat d'espai en disc. A més, com tot l'espai de disc necessari s'assigna anticipadament, la planificació de la capacitat és molt senzilla.

Són els arxius RRD els que emmagatzemen les dades de les mètriques en gmetad. Els fitxers RRD s'emmagatzemen en la següent ruta per defecte:

/var/lib/ganglia/rrds/<nom_del_clúster>/<nom_del_node>/

on cada mètrica s'emmagatzema en un únic arxiu RRD.

gweb

gweb és la interfície de visualització de Ganglia. Proporciona accés instantani a qualsevol mètrica de qualsevol host en el clúster sense especificar cap detall de configuració.

Resumeix visualment tot l'estat del clúster mitjançant gràfics que combinen les mètriques per clúster i proporciona desplegables per a obtenir detalls addicionals.

gweb permet canviar el període de temps en els gràfics, suporta l'extracció de dades en diversos formats textuals (CSV, JSON, i més), i proporciona una interfície d'URL totalment funcional perquè es pugui incrustar els gràfics necessaris en altres programes a través d'URLs específiques. A més, gweb és un programa PHP, que s'executa sota el servidor web Apache i sol instal·lar-se en el mateix host que gmetad i RRDtool, ja que necessita accedir a les bases de dades RRD creades per gmetad.

6.4. Cas pràctic 2: Ganglia en el nostre clúster Hadoop

En aquest cas pràctic anam a treballar amb el clúster Hadoop que vàrem configurar en el Lliurament 1. Si teniu problemes de memòria, podeu reduir el clúster, però deixeu almenys un node mestre i un node esclau. En els apunts treballarem amb tres nodes, un mestre i dos esclaus. A més, configurarem una altra màquina virtual que pugui executar un entorn gràfic i que emprarem per connectar-nos amb el navegador a la interfície gràfica de Ganglia. Aquí treballarem amb un Ubuntu Desktop 24.04 LTS, però pot ser qualsevol altre sistema operatiu sempre que pugui executar un navegador web. És important que aquesta màquina tengui dues targets de xarxa, una de tipus NAT per a connectar a Internet i l'altra de tipus "Adaptador sólo-anfitrión" per poder connectar amb el clúster.

A continuació anam a instal·lar Ganglia en el nostre clúster. Començarem pel node mestre i després ho farem en els esclaus.

Instal·lació de Ganglia en el node mestre

Descarregam les eines necessàries:

```
$ yum update
$ yum install ganglia rrdtool ganglia-gmetad ganglia-gmond ganglia-web
```

Configuram l'usuari adminganglia per accedir a l'eina web de Ganglia:

```
$ htpasswd -c /etc/httpd/auth.basic adminganglia
$ setsebool -P httpd_can_network_connect 1
```

Si no tenim ja deshabilitat el firewall, l'hem d'aturar i deshabilitar ara (o bé obrir el port 8649):

```
$ systemctl stop firewalld
$ systemctl disable firewalld
```

Hem de modificar el fitxer **/etc/httpd/conf.d/ganglia.conf** amb el contingut següent (o substituir-lo per aquest fitxer), per configurar el context /ganglia del servidor web:

```
# Ganglia monitoring system php web frontend
Alias /ganglia /usr/share/ganglia
<Location /ganglia>
        AuthType basic
        AuthName "Ganglia web UI"
        AuthBasicProvider file
        AuthUserFile "/etc/httpd/auth.basic"
        Require user adminganglia
</Location>
```

Ara modificarem els fitxers de configuració del gmond i el gmetad. Substituirem el contingut del fitxer **/etc/ganglia/gmond.conf** pel contingut d'aquest fitxer. I també substituirem el contingut del fitxer **/etc/ganglia/gmetad.conf** pel contingut d'aquest altre fitxer.

Finalment, hem de configurar les mètriques de Hadoop perquè Ganglia les pugui sondejar. Hem de substituir el contingut del fitxer **/home/hadoop/hadoop/etc/hadoop/hadoop-metrics2.properties** pel contingut d'aquest fitxer.

Instal·lació de Ganglia en els nodes esclaus

En cada un dels nodes esclaus haurem de fer les següents passes.

Descarregam les eines necessàries:

```
$ yum update
$ yum install ganglia-gmond
```

Si no tenim ja deshabilitat el firewall, l'hem d'aturar i deshabilitar ara (o bé obrir el port 8649):

```
$ systemctl stop firewalld
$ systemctl disable firewalld
```

Hem de substituir el contingut del fitxer **/etc/ganglia/gmond.conf** pel contingut d'aquest fitxer.

I també hem de substituir el contingut del fitxer **/home/hadoop/hadoop/etc/hadoop/hadoop-metrics2.properties** pel contingut d'aquest fitxer.

Per acabar, hem de reiniciar i activar gmond:

```
$ systemctl restart gmond
$ systemctl enable gmond
```

Reiniciar el clúster i Ganglia

Ara que ja hem configurat el mestre i tots els esclaus, hem de tornar al node mestre i reiniciar el clúster, amb l'usuari **hadoop**:

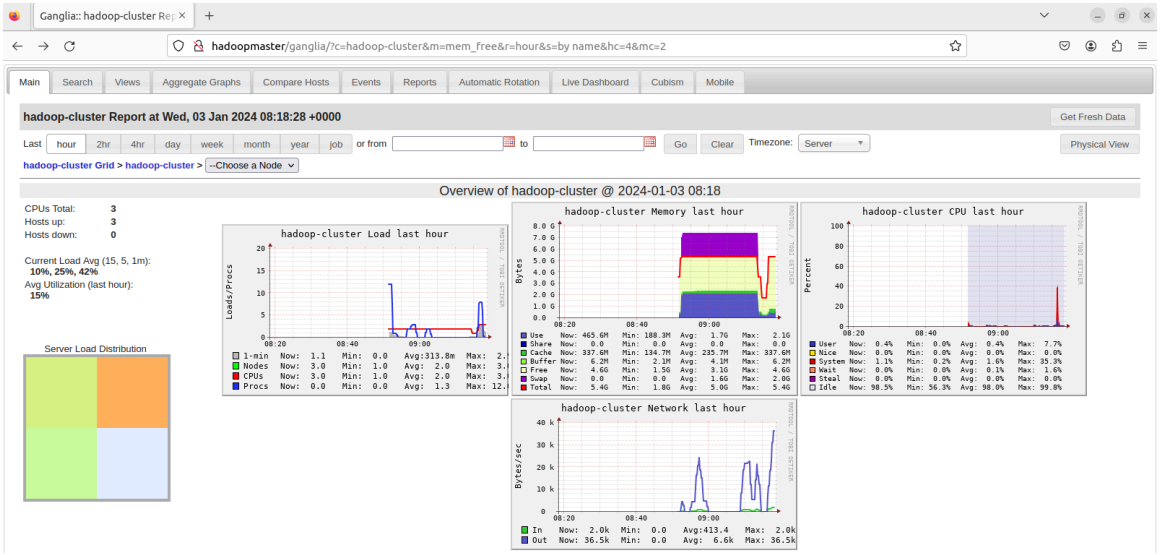
```
$ stop-all.sh
$ start-all.sh
```

Finalment, reiniciam i activam els serveis de Ganglia en el node mestre (amb l'usuari **root** o emprant sudo):

```
$ systemctl restart httpd gmetad gmond
$ systemctl enable httpd gmetad gmond
```

Interactuar amb la interfície web de Ganglia

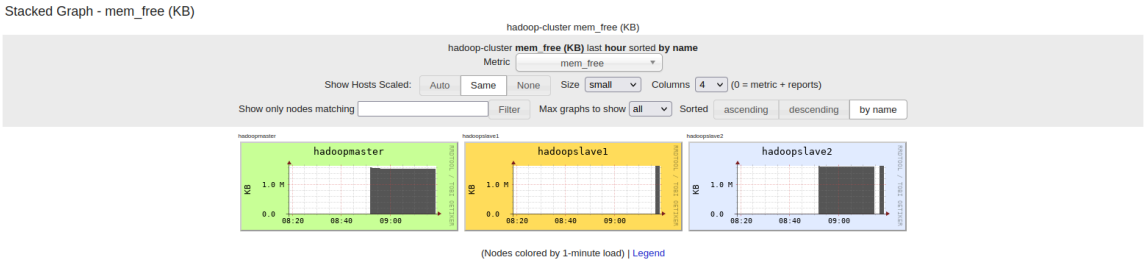
Podem accedir a la interfície web de Ganglia en el node mestre: **http://hadoopmaster/ganglia** :



Imatge: *Pàgina inicial de la interfície web de Ganglia*

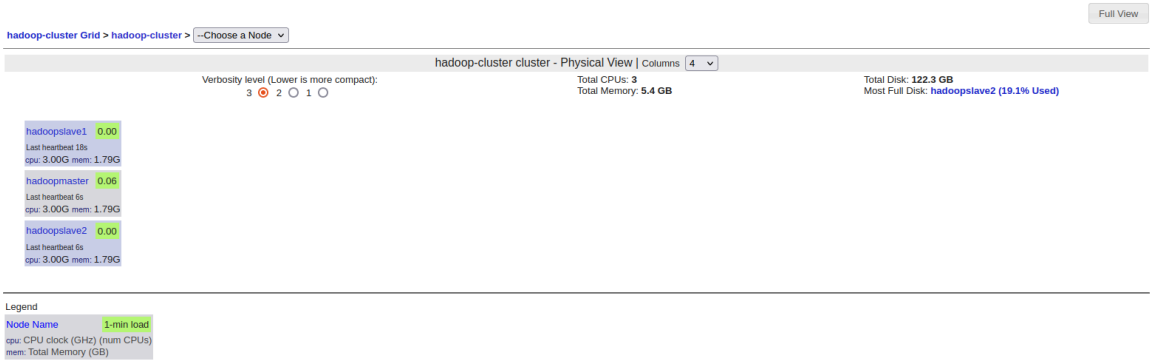
Aquí podem veure la càrrega i l'ús global de memòria CPU i xarxa del clúster durant la darrera hora. També, a la part de l'esquerra veim que tenim 3 hosts en marxa i 0 de caigut i que s'estan emprant 3 CPUs en el clúster (una cada host).

Si baixam fins al final de la pàgina podem veure la gràfica de cada node corresponent a una mètrica seleccionada. En concret, la següent imatge mostra la memòria lliure (mem_free):



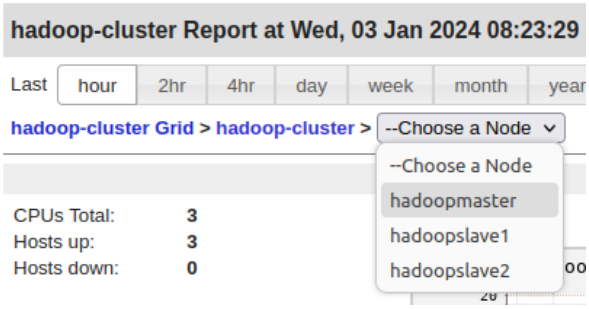
Imatge: *Mètrica mem_free en els 3 nodes del clúster*

Si tornam a la part superior, podem pitjar el botó "Physical View" per veure alguns detalls més sobre el clúster:



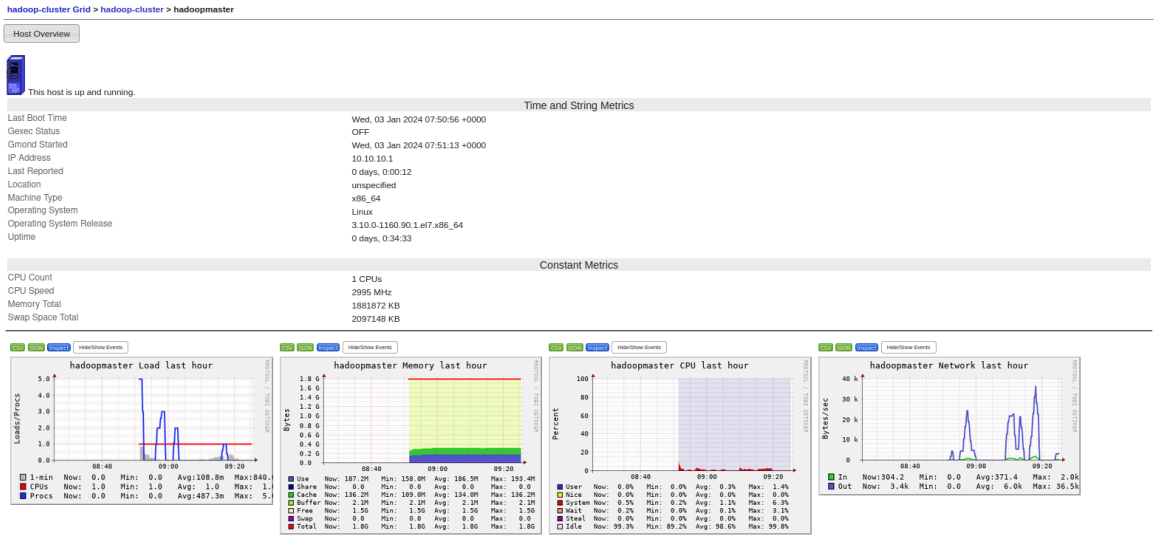
Imatge: *Vista física del clúster*

Si tornam ara a la "Host View", al menú superior podem seleccionar un node en concret, per exemple el mestre:



Imatge: *Selecció d'un node (hadoopmaster)*

Ara veim les gràfiques referides només al node seleccionat (i també hem desplegat la vista general del host):



Imatge: *Visió del host hadoopmaster*

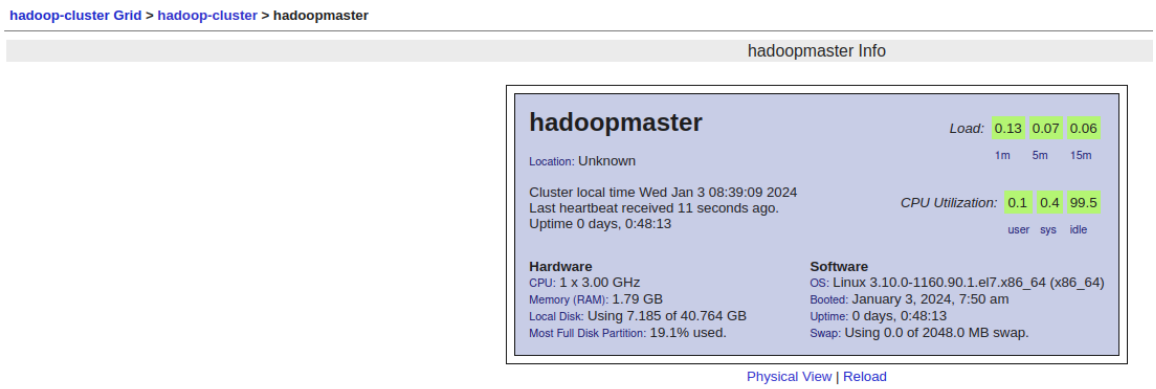
Si baixam fins a la part inferior d'aquesta pàgina podem veure els gràfics de moltes més mètriques referides a aquest host:

Imatge: *Gràfiques de les mètriques del host hadoopmaster*

Podem botar directament a algun grup de mètriques que ens interessi, per exemple, a les mètriques de memòria:

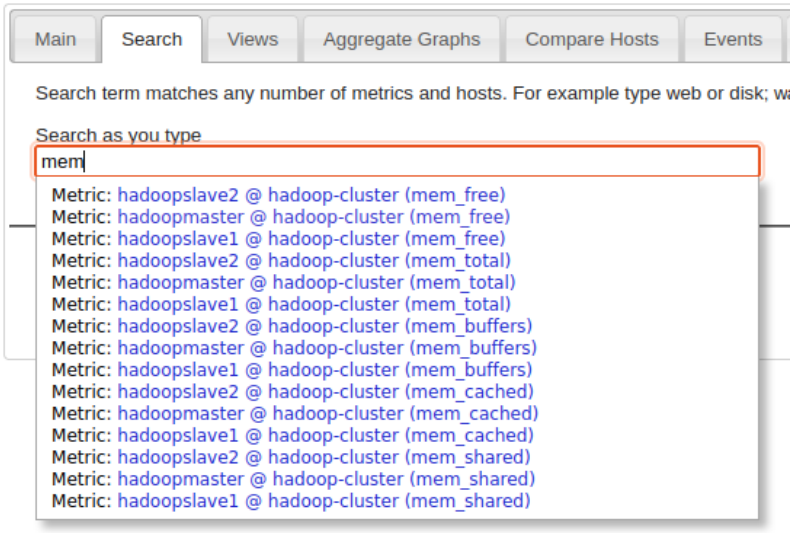
Imatge: *Gràfiques de les mètriques de memòria de hadoopmaster*

Si tornam a la part superior, podem pitjar el botó "Node View" per veure els detalls del node mestre:



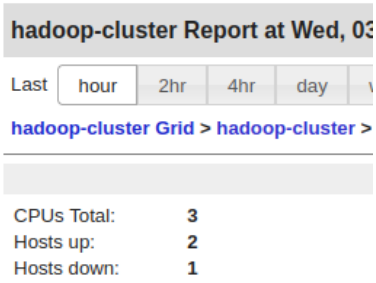
Imatge: *Vista del node hadoopmaster*

També podem emprar la pestanya "Search" del menú superior per cercar una mètrica en concret. Per exemple, si volem cercar mètriques relacionades amb la memòria:



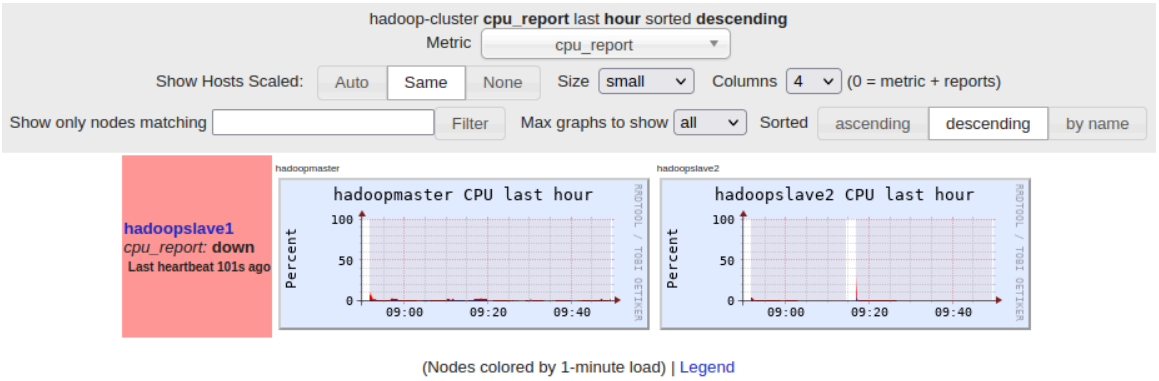
Imatge: *Cerca de mètriques relacionades amb la memòria*

Per últim, anam a fer la prova d'aturar un node, per exemple hadoopslave1. Tornem a la vista general del clúster i podem comprovar que ara només tenim 2 hosts en marxa i 1 caigut:



Imatge: *Un host caigut*

Si baixam a veure les mètriques podem observar com ens resalta que hadoopslave1 està caigut i que fa 101 segons que no s'ha rebut cap *heartbeat* seu:



Imatge: *Mètriques amb un host caigut*

6.5. Apache Ambari

Apache Ambari va ser creat per l'Apache Software Foundation com a part del projecte Hadoop amb l'objectiu d'oferir una solució eficaç i fàcil d'utilitzar per a la gestió de clústers Hadoop. Ambari facilita la configuració, la gestió i el monitoratge de clústers Hadoop, proporcionant una interfície d'usuari amigable que fa més accessible la complexa tasca de gestionar un clúster Hadoop.

La seva primera versió va ser llançada el 2011, com a resposta a la necessitat creixent d'una eina que pogués gestionar de manera eficient la complexitat creixent dels clústers Hadoop. Des del seu inici, Ambari ha evolucionat significativament, afegint noves funcionalitats i millores per a adaptar-se a les canviants demandes de les tecnologies de big data. La seva capacitat per integrar-se amb diverses plataformes i components de l'ecosistema Hadoop, juntament amb el suport continuat de la comunitat de desenvolupadors, ha fet d'Ambari una eina clau en el món del processament de dades a gran escala. El desenvolupament d'Ambari ha estat un reflex de l'evolució de Hadoop, adaptant-se constantment per a gestionar clústers més grans i més complexos, així com per a simplificar la seva administració.

Un dels punts forts d'Ambari és el monitoratge dels clústers Hadoop. Proporciona una visió detallada de l'estat del clúster, incloent mètriques sobre CPU, memòria, disc i xarxa. Els administradors poden veure el rendiment del clúster en temps real i també revisar dades històriques per a l'anàlisi de tendències.

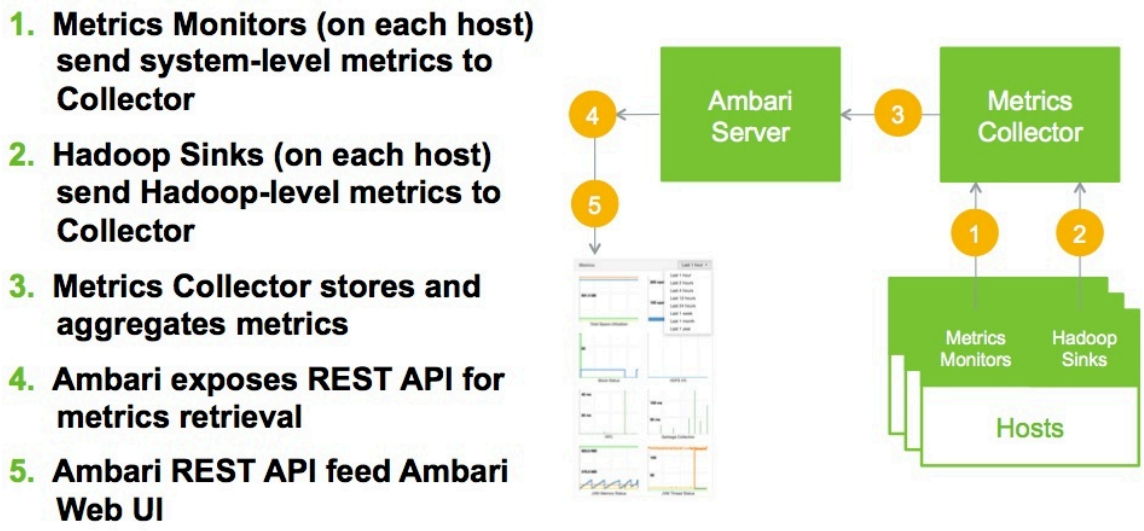
Aquestes són algunes de les principals funcionalitats d'Ambari pel que fa al monitoratge:

- Visió general del clúster: Ambari presenta un resum dels serveis del clúster, estat de salut i alertes.
- Gestió de serveis: permet als usuaris iniciar, aturar i reiniciar serveis Hadoop, així com visualitzar el seu estat.
- Visualització de mètriques: exhibeix gràfics de rendiment dels diferents components del clúster.
- Gestió d'alertes: notifica als administradors sobre qualsevol problema o irregularitat en el rendiment del clúster.
- Auditoria i registres: ofereix un accés fàcil als registres per a la diagnosi ràpida de problemes.

Ambari Metrics System (AMS)

Ambari Metrics System (AMS) és el sistema de recollida, agregació i visualització de mètriques dins de l'ecosistema Hadoop, i forma part del projecte Apache Ambari. L'objectiu principal d'AMS és proporcionar una manera unificada i centralitzada de recollir mètriques de tots els components d'un clúster Hadoop. Això inclou mètriques de CPU, memòria, disc, xarxa, així com mètriques específiques de components de Hadoop com HDFS, MapReduce o YARN.

AMS consisteix principalment en dos components: l'**Ambari Metrics Collector**, que és el repositori central on es guarden totes les mètriques, i els **Ambari Metrics Monitors**, que s'executen en cada node del clúster per recollir mètriques i enviar-les al Collector. A més, a través de la interfície d'usuari d'Apache Ambari, els usuaris poden visualitzar aquestes mètriques en forma de gràfics i taules, permetent una anàlisi fàcil del rendiment i la salut del clúster. Vegem la següent imatge que mostra la arquitectura d'AMS i el seu funcionament general:



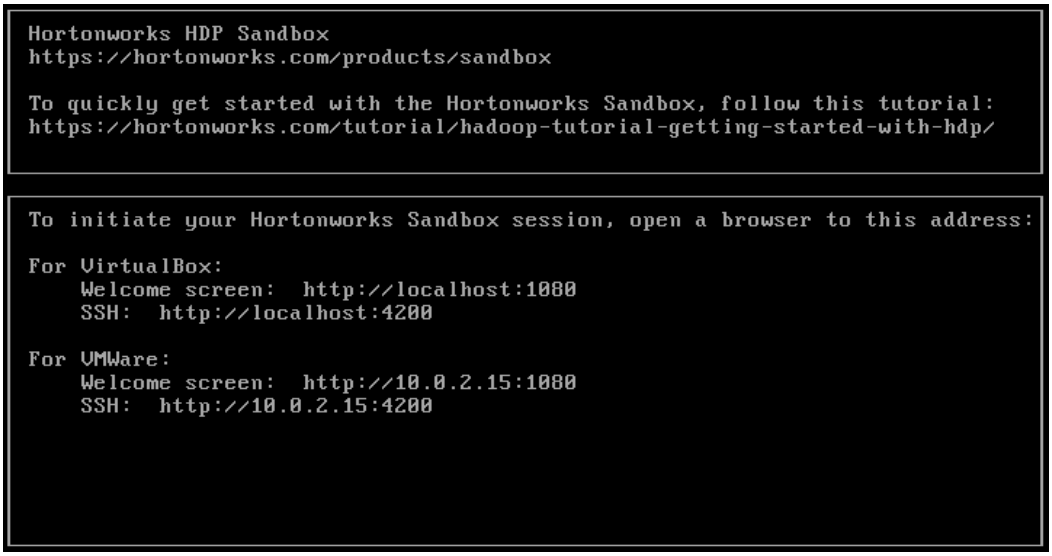
Imatge: *Arquitectura d'Ambari Metrics System (AMS). Font: https://cwiki.apache.org/*

D'altra banda, Ambari es pot integrar amb una varietat d'altres eines de monitoratge com Ganglia i Nagios, proporcionant una solució completa de monitoratge per a clústers Hadoop. En la configuració més habitual, Ganglia actua com a recollidor principal de mètriques de rendiment a nivell de hardware i xarxa, mentre que Ambari utilitza aquesta informació (i la que ell mateix recull de les mètriques d'Ambari) per proporcionar una interfície unificada i centralitzada en Hadoop per a la gestió i visualització de mètriques.

6.6. Cas pràctic 3: Ambari en el HortonWorks Sandbox HDP

En aquest cas pràctic anam a treballar amb el Hortonworks HDP Sandbox per a Oracle VirtualBox, que recordem que vàrem mencionar en el Lliurament 2 i que es pot descarregar des de https://archive.cloudera.com/hwx-sandbox/hdp/hdp-3.0.1/HDP_3.0.1_virtualbox_181205.ova. Una vegada hem afegit la màquina virtual a VirtualBox (amb doble clic sobre l'arxiu .ova), la primera vegada que l'arranquem tardarà una llarga estona mentre s'extreu i es carrega el sandbox. Les següents vegades ja arrancarà molt més ràpidament.

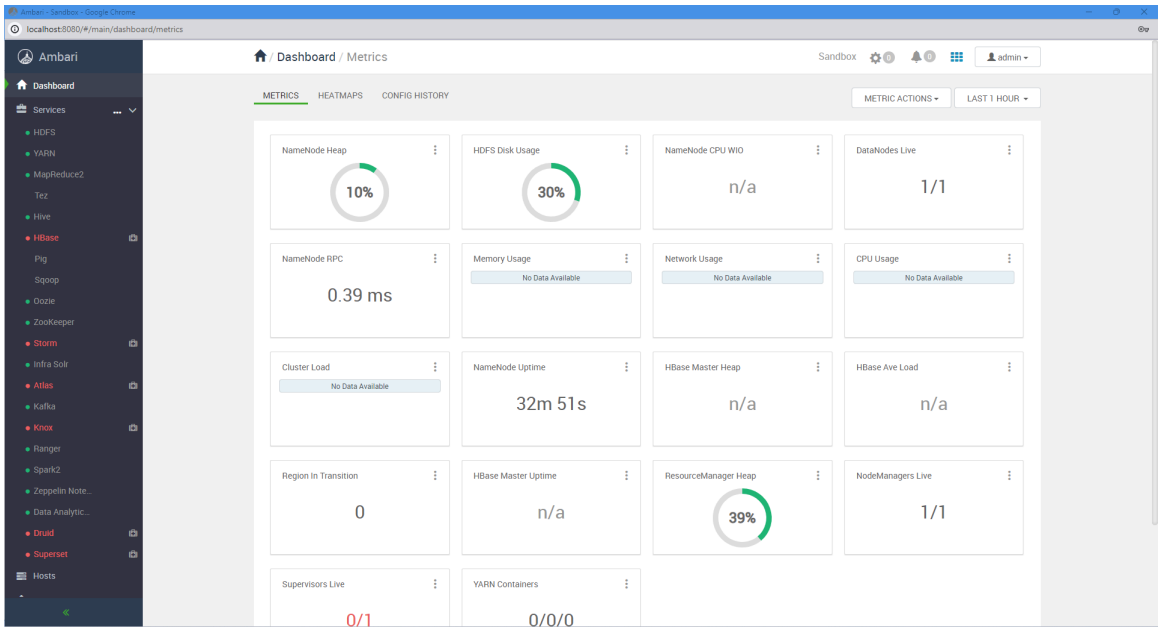
Una vegada ja s'ha arrancat la màquina virtual, podem interactuar amb ella des del navegador de la màquina anfitriona, mitjançant una interfície web (http://localhost:1080) o una consola SSH (http://localhost:4200).



Imatge: *Hortonworks Sandbox HDP arrancat*

Nosaltres treballarem amb la interfície web, però abans, hem de configurar l'usuari i contrasenya d'administrador. Per fer-ho, hem d'entrar amb la consola (http://localhost:4200) amb l'usuari ***root*** i contrasenya ***hadoop***. Ens demanarà canviar la contrasenya. A continuació hem d'executar l'ordre ***ambari-admin-password-reset*** (podeu fer *paste* amb el botó dret del ratolí i seleccionant *Paste from browser*). Això canviarà la contrasenya de l'usuari ***admin***, que serà el que emprarem per a entrar en la interfície web.

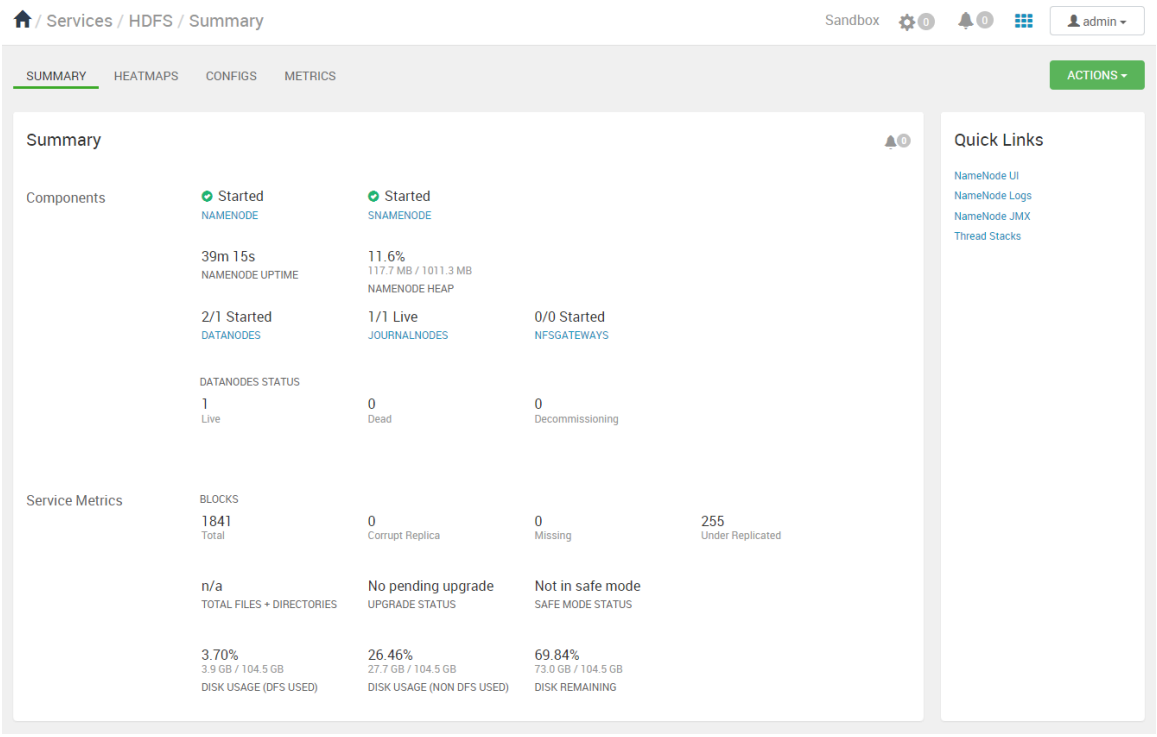
Entrem ja a la interfície web (http://localhost:1080) i farem clic en el botó "LAUNCH DAHSBOARD", la qual cosa ens obrirà una nova finestra amb la interfície web d'Apache Ambari, després d'introduir l'usuari ***admin*** amb la seva contrasenya.



Imatge: *Dashboard d'Apache Ambari*

Al panell central veim un resum de les mètriques principals del clúster. Per exemple, veim que la memòria Heap del NameNode està al 10% i el disc HDFS està al 30% de capacitat. Veim també que el clúster té un DataNode actiu.

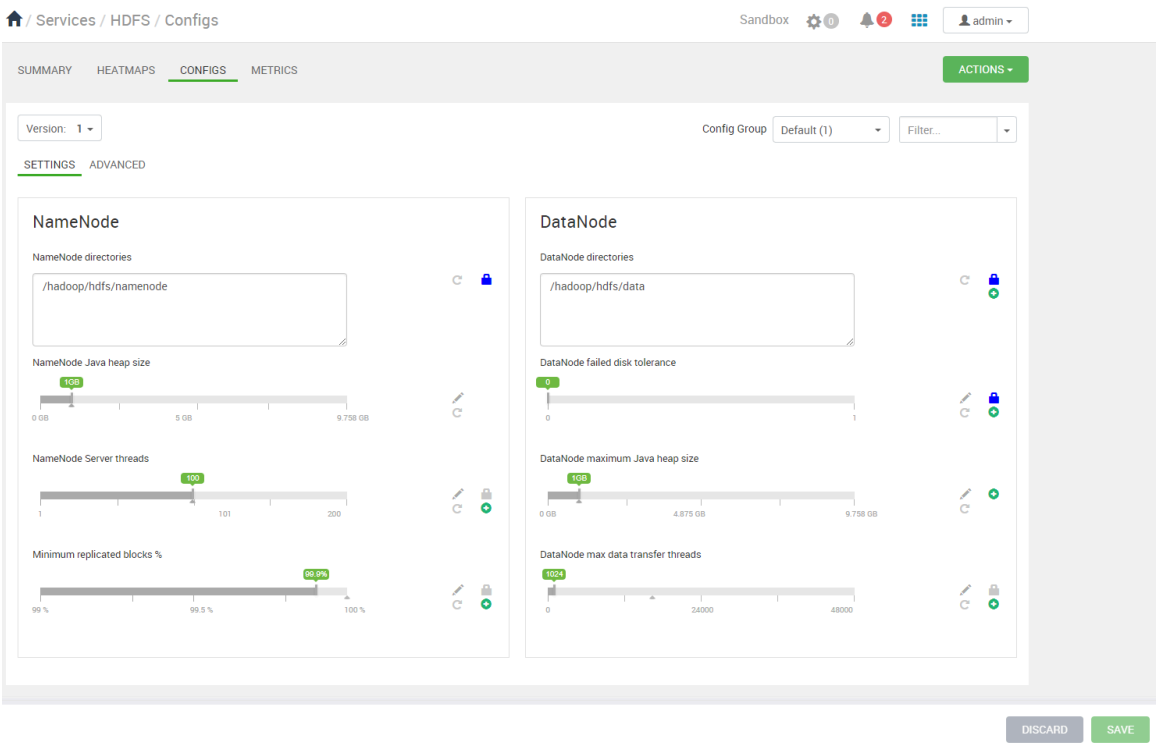
Al panell de l'esquerra, en l'apartat de *Services*, veim el llistat dels serveis disponibles en el nostre sandbox. Els que tenen el punt verd indiquen que estan en marxa, mentre que els del punt vermell estan aturats. Si seleccionam un d'ells, per exemple HDFS, podem veure un resum de les mètriques relacionades amb aquest servei.



Imatge: *Resum de mètriques del servei HDFS*

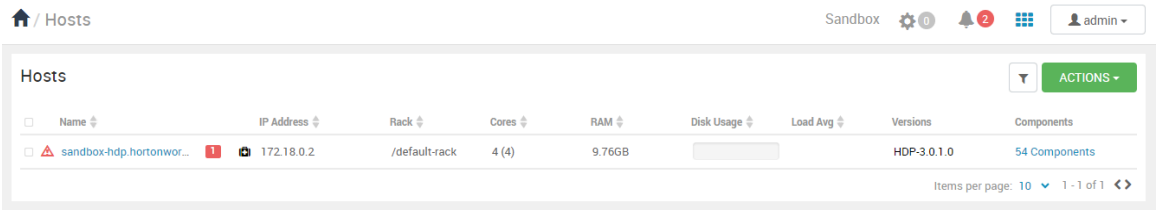
Veim també com a la dreta ens apareixen els enllaços a les interfícies web del NameNode, arxius de log d'HDFS, JMX del NameNode i informació sobre els threads actius.

A més, Ambari també permet modificar paràmetres de configuració dels serveis. Per exemple, seguint en HDFS, anem a la pestanya "CONFIGS":



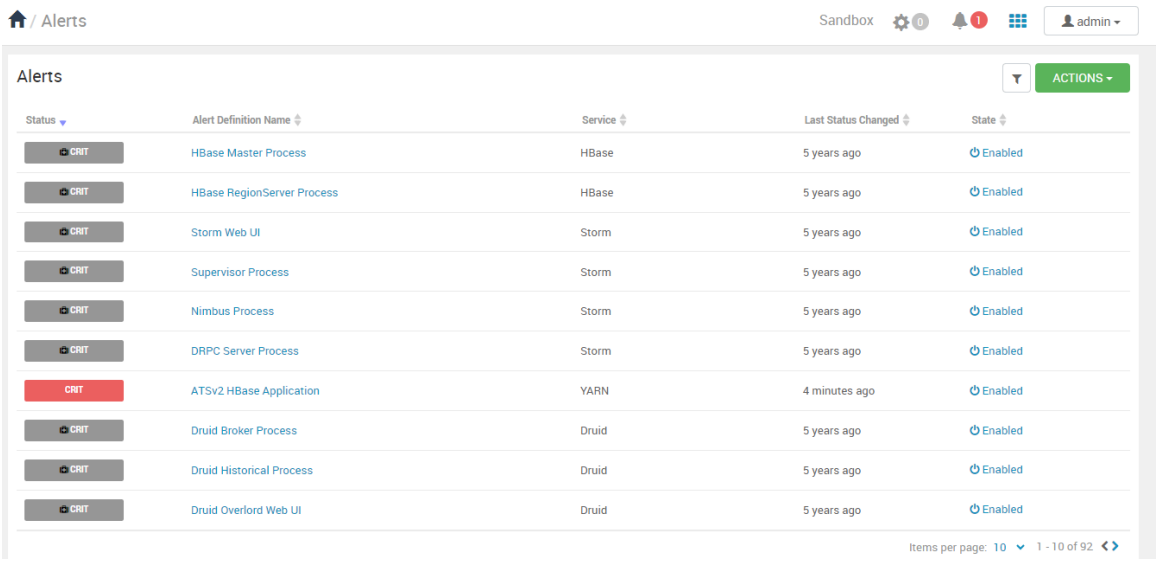
Imatge: *Configuració dels serveis*

Si tornam a mirar el panell de l'esquerra, tenim un altre apartat "Hosts". En el cas del nostre sandbox, només tenim un host (que ens indica que tenim avisos crítics), però quan treballam amb un clúster, aquest panell resulta especialment útil.



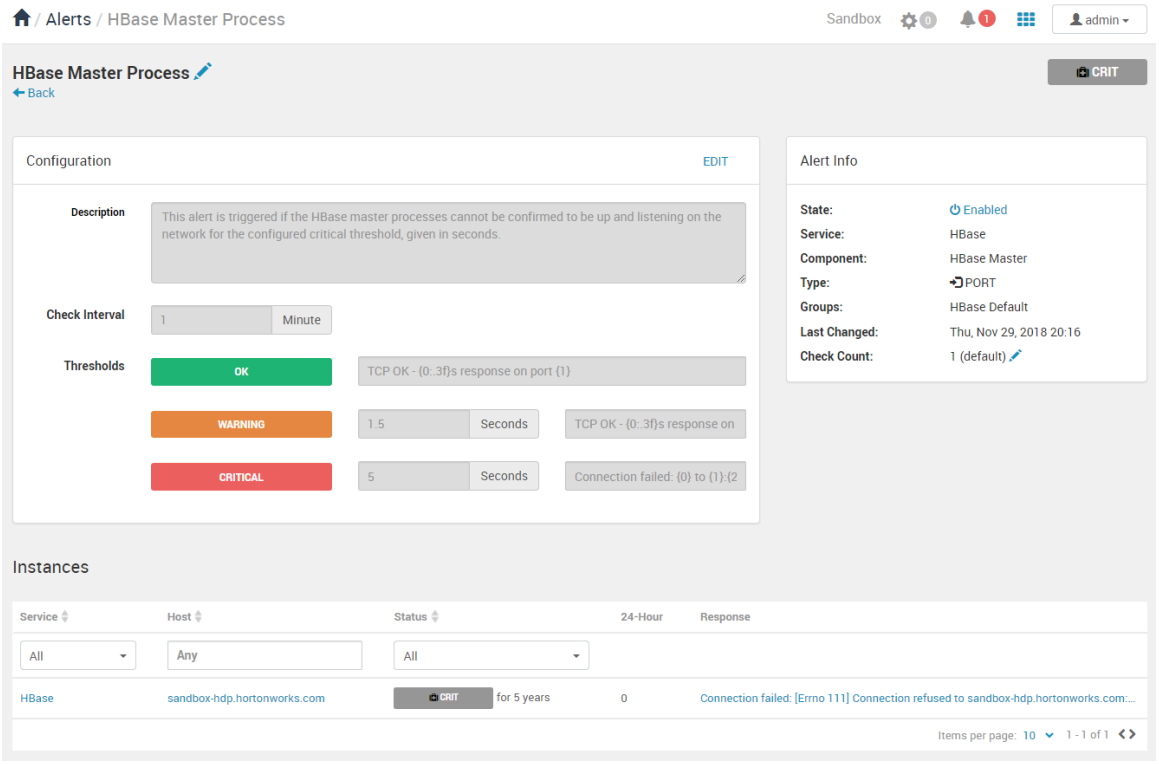
Imatge: *Hosts*

Per acabar, en el panell de l'esquerra també podem accedir a les alertes del sistema:



Imatge: *Alertes*

Fent clic sobre qualsevol d'elles, podem veure els detalls, en aquest cas la que ens dona un missatge crític indicant que no s'ha pogut contactar amb el procés mestre de HBase (la base de dades NoSQL de l'ecosistema Hadoop):



Imatge: *Alerta crítica sobre el procés mestre de HBase*

7. Bibliografia

Aquests apunts estan basats parcialment en els capítols 10, 11 i 12 dels apunts de Big data aplicat de la Universitat de Castella-La Mancha.

Altres fonts web que també s'han utilitzat per a la seva elaboració són:

- Monitoring Hadoop Cluster (v2.x.x) using Ganglia
- Setup Real-Time Monitoring using Ganglia on Centos 7
- How to monitor Hadoop metrics i How to collect Hadoop metrics (sèrie de 4 articles sobre monitorització en Hadoop)
- Ganglia Wiki Page
- Pàgina del projecte Apache Ambari i Wiki sobre Ambari Metrics System

Així mateix, tot i que escrits ja fa al voltant d'una dècada, aquests són probablement els llibres més reconeguts en l'àmbit de la monitorització de Hadoop i sobre Ganglia:

- *Practical Hadoop Security*. Bhushan Lakhe. Apress, 2014.
- *Monitoring Hadoop*. Gurmukh Singh. Packt Publishing, 2015.
- *Monitoring with Ganglia*. Alex Dean, Robert Alexander, Dave Josephsen, Vladimir Vuksan, Bernard Li, Brad Nicholes, Jeff Buchbinder, Frederiko Costa, Matt Massie, Peter Phaal i Daniel Pocock. O'Reilly, 2012.