

Apunts CE_5075 1.1

lloc: [Institut d'Ensenyaments a Distància de les Illes
Balears](#)

Curs: Big data aplicat

Llibre: Apunts CE_5075 1.1

Imprès per: Carlos Sanchez Recio

Data: dimecres, 2 d'octubre 2024, 20:32

Taula de continguts

1. Introducció

2. Processament paral·lel

3. Processament distribuït

3.1. Sistemes d'arxius distribuïts

4. Apache Hadoop

5. HDFS

5.1. Arquitectura d'HDFS

5.2. Accés a HDFS

5.3. FS Shell

6. Configuració d'un clúster Hadoop

6.1. Preparar l'entorn

6.2. Instal·lar Hadoop en el node mestre

6.3. Configurar Hadoop en el node mestre

6.4. Engegar Hadoop amb un únic node

6.5. Afegir dos esclaus al clúster

6.6. Provar el clúster

6.7. Interactuar amb la interfície gràfica

1. Introducció

En el primer lliurament del mòdul de Sistemes de Big Data hem vist què entenem per *big data* o dades massives. No ho tornarem a repetir aquí, però sí recordarem que la definició més emprada és la de les tres V, que enumera les característiques principals de les dades massives: **volum**, **varietat** i **velocitat**.

Tractar aquests grans volums de dades variades (estructurades i no estructurades) a una velocitat acceptable no és fàcil. No ho podem fer amb un únic ordinador, per molt potent que aquest sigui. Necessitam emprar molts d'ells treballant conjuntament. És per això que en els darrers anys s'han desenvolupat i popularitzat noves solucions per al processament distribuït. La més utilitzada avui en dia és la d'Apache Hadoop.

En aquest lliurament veurem els fonaments de la computació paral·lela i, sobretot de la computació distribuïda. A continuació farem una introducció a Apache Hadoop, fent especial èmfasi en un dels seus components principals: HDFS, el sistema d'arxius distribuït de Hadoop.

Acabarem veient com podem configurar un clúster Hadoop amb diversos nodes.

En el següent vídeo pots veure un resum del que tractarem en aquest lliurament.



Vídeo: Resum del Lliurament 1

2. Processament paral·lel

Tant el volum de dades com la velocitat de processament han seguit una trajectòria d'augment exponencial des de l'inici de l'era digital, però el primer ha augmentat a un ritme molt major que el segon. D'aquesta manera es fa imprescindible comptar amb noves eines, com ara la computació o processament paral·lel, per aconseguir salvar la bretxa generada.

Tradicionalment, els programes informàtics s'han escrit per a la seva computació en sèrie, on un conjunt d'instruccions s'executen, una darrera d'una altra, en una unitat central de processament, en un únic ordinador. La computació en paral·lel, per contra, utilitza simultàniament múltiples elements de processament per a resoldre un problema. Això s'aconsegueix mitjançant la divisió del problema en parts independents de manera que cada element de processament pugui executar la seva part de l'algorisme de manera simultània amb els altres. Els elements de processament són diversos i inclouen recursos com ara una computadora amb múltiples processadors, diversos ordinadors en xarxa, maquinari especialitzat, o qualsevol combinació dels anteriors.

Els algorismes i arquitectures de processament paral·lel s'estudien des dels anys 1950 com una manera de millorar el rendiment dels sistemes d'informació i, més recentment, tenint en compte el consum d'energia: l'objectiu és millorar el rendiment mantenint el consum d'energia sota control.

Idealment, si tenim una tasca que necessita 10 hores per executar-se en un processador, emprant-ne 10 processadors, n'hauria d'emprar només una hora. O vist d'una altra manera, doblant el número de processadors, doblam la velocitat. No obstant això, molt pocs algorismes paral·lels aconseguixen aquesta acceleració òptima. El potencial increment de la velocitat d'un algorisme en una plataforma de computació paral·lela ve donat per la **Llei d'Amdahl**, que va ser formulada per Gene Amdahl en els anys 1960, i que té en compte que qualsevol programa sol tenir porcions que no es poden paral·lelitzar. D'aquesta manera, l'acceleració màxima ve determinada pel percentatge no paral·lelitzable d'un programa, segons aquesta fórmula, on S és l'acceleració (*speedup*) i α el percentatge no paral·litzable:

$$S = \frac{1}{\alpha}$$

Per exemple, si la porció seqüencial d'un programa suposa un 20% (0,2) del temps d'execució, la S serà 5, és a dir, no podem aconseguir accelerar més de 5 vegades l'execució del programa. Per tant, no té cap utilitat utilitzar més de 5 processadors, seria el mateix emprar 5 processadors que 500.

Molt relacionada amb la Llei d'Amdahl és la **Llei de Gustafson**, que considera que la part no paral·lela d'un programa decreix quan la mida del problema augmenta. Per tant, qualsevol problema suficientment gran pot ser paral·lelitzat eficientment.



Existeixen moltes taxonomies per classificar les diferents arquitectures per al processament seqüencial i paral·lel que s'han anat dissenyant i produint al llarg del temps. La més referenciada és la que va definir Michael J. Flynn en 1966, coneguda com la **taxonomia de Flynn**, que està basada en dues magnituds: per un costat, si els diferents elements de procés operen utilitzant un conjunt únic o conjunts múltiples d'instruccions; i per l'altre, si aquestes instruccions utilitzen un conjunt de dades únic o múltiple.

Així, tenim aquesta classificació:

	Instruccions individuals	Instruccions múltiples
Dades individuals	SISD	MISD
Dades múltiples	SIMD	MIMD

Taula: Taxonomia de Flynn

Les arquitectures **SISD** (Instruccions individuals - Dades individuals) equivalen als sistemes totalment seqüencials, la tradicional arquitectura de Von Neumann.

Els sistemes **SIMD** (Instruccions individuals - Dades múltiples) processen en paral·lel més d'una seqüència de dades amb una única seqüència d'instruccions. Aquí el processament és síncron, l'execució de les instruccions

segueix essent seqüencial com en el cas anterior, tots els elements realitzen la mateixa instrucció, però sobre una gran quantitat de dades.

Els sistemes **MISD** (Instruccions múltiples - Dades individuals) executen diverses operacions simultàniament però sobre les mateixes dades. No tenen gaire utilitat pràctica.

En els sistemes **MIMD** (Instruccions múltiples - Dades múltiples), múltiples processadors independents operen sobre diferents seqüències de dades. Cada un dels processadors aplica una seqüència diferent d'instruccions a una seqüència d'entrada diferent, per obtenir una seqüència de sortida.

Per altra banda, la coneguda com a **Llei de Moore**, prediu que aproximadament cada 2 anys es duplica el nombre de transistors que es poden introduir en un microprocesador. Aquesta llei continua vigent encara avui dia. Fins a principis del segle XXI, aquest augment en el número de transistors es dedicava fonamentalment a millorar el rendiment mitjançant un augment de la freqüència de rellotge. Però a partir d'aquest moment, comencen a aparèixer problemes per l'efecte de la dissipació de la calor, que dificulta la refrigeració dels circuits superdensos. Aleshores, els transistors addicionals, que fins llavors s'empraven per a augmentar la freqüència de rellotge, passen a utilitzar-se per afegir hardware addicional que permeti la computació paral·lela. Així doncs, l'atenció a la millora del rendiment es va traslladar a l'ús de múltiples processadors independents en un mateix xip, donant lloc als **processadors multinuclis**.

L'aparició del Big Data requereix una reavaluació de la forma en què mesurar el rendiment dels supercomputadors. Mentre que abans el rendiment es mesura fonamentalment en els FLOPs (operacions de coma flotant per segon), amb el Big Data, l'entrada/sortida i l'amplada de banda d'emmagatzematge assumeixen un paper més important en la determinació del rendiment. Un supercomputador que realitza càlculs científics pot rebre escassos paràmetres d'entrada i un conjunt d'equacions que defineixen un model i després realitzar càlculs durant dies o setmanes, abans de retornar les respostes. En canvi, moltes aplicacions Big Data requereixen un flux constant de noves dades que s'introdueixen, s'emmagatzemen, es processen i s'ofereixen com a resultats, amb el que possiblement es posa a prova la memòria i l'amplada de banda d'E/S, capacitats que solen ser més limitades que la velocitat de càlcul.

Així doncs, en els últims anys, s'han introduït molts models abstractes de processament paral·lel, en un esforç per a representar amb precisió els recursos de processament, emmagatzematge i comunicació, juntament amb les seves interaccions, permetent als desenvolupadors d'aplicacions paral·leles visualitzar i aprofitar els avantatges disponibles, sense haver d'ocupar-se en detalls específics de la màquina. Amb un alt nivell d'abstracció podem distingir els enfocaments de processament paral·lel de dades i de control.

El **paral·lelisme de dades** implica la partició d'un gran conjunt de dades entre múltiples nodes de processament, cadascun d'ells operant sobre una part assignada de les dades, abans que els resultats parcials s'acabin combinant. Són l'equivalent a l'arquitectura SIMD de la taxonomia de Flynn. Sovint, en les aplicacions Big Data el mateix conjunt d'operacions ha d'executar-se en cada subconjunt de dades, per la qual cosa el processament SIMD és l'alternativa més eficient. En la pràctica, no obstant això, la sincronització d'un gran nombre de nodes de processament introdueix sobrecàrregues i ineficiències que redueixen els guanys de velocitat. Per això pot ser beneficiós dirigir els nodes de processament perquè executin un únic programa amb múltiples dades de manera asíncrona, amb una coordinació dispersa.

En els esquemes basats en el **paral·lelisme de control** diversos nodes operen independentment resolent subproblemes, sincronitzant-se entre si mitjançant l'enviament de missatges informatius i de sincronització. Són l'equivalent a les arquitectures MIMD de la taxonomia de Flynn. Aquesta independència permet emprar recursos heterogenis i recursos específics de l'aplicació sense interferències creuades ni que recursos més lents obstaculitzin el progrés dels més ràpids.

Resumint, en el paral·lelisme de dades, una mateixa operació és realitzada simultàniament sobre diferents dades. Cada conjunt de dades estarà associat a un processador diferent. En canvi, el paral·lelisme de control permet que diverses tasques siguin executades per diferents processadors al mateix temps.

3. Processament distribuït



DEFINICIÓ

Un **sistema distribuït** és una col·lecció d'ordinadors autònoms enllaçats per una xarxa d'ordinadors i suportats per un programari que fa que la col·lecció actuï com un servei integrat.

George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. Pearson Education, 2005.

Alguns dels aspectes més importants a tenir en compte dins d'un sistema distribuït actual són els següents:

- **Obertura:** El sistema ha de ser ampliable, és a dir, ha d'existir la possibilitat d'afegir nous recursos i serveis compartits i que aquests es puguin posar a la disposició dels diferents components que formen el sistema.
- **Concurrencia:** Els diferents components d'un sistema distribuït poden demandar accedir a un recurs simultàniament. És necessari que el sistema estigui dissenyat per a permetre-ho.
- **Escalabilitat:** Un sistema és escalable si manté la seva eficiència quan hi ha un increment significatiu en el nombre de recursos i el nombre d'usuaris.
- **Heterogeneïtat:** El sistema distribuït pot estar format per una varietat de diferents xarxes, sistemes operatius, maquinari, etc. Malgrat aquestes diferències, els components han de poder interactuar entre si.
- **Tolerància a fallades:** Qualsevol sistema pot fallar. En el cas d'un sistema distribuït malgrat la fallada d'un component ha de ser possible que el sistema continuï funcionant.
- **Transparència:** La transparència permet que uns certs aspectes del sistema siguin invisibles a les aplicacions, per exemple, la ubicació, l'accés, la concurrència, la gestió de les fallades, la replicació de la informació, etc.

Els **sistemes computacionals distribuïts** són sistemes de computació d'alt rendiment que estan formats per conjunts d'ordinadors interconnectats mitjançant una xarxa que ofereixen funcionalitats diverses, algunes d'elles pròpies de la supercomputació, com ara la computació paral·lela. Entre aquests sistemes cal destacar els denominats clústers, els basats en grid i les en arquitectures al núvol (*cloud*).

Clústers

Segons el Termcat, un **clúster** és un "conjunt d'unitats funcionals interconnectades per mitjà d'una xarxa que actuen com una sola unitat". Una altra definició és la de Thomas Sterling, pare del clúster Beowulf: "una classe d'arquitectura paral·lela que es basa a unir màquines independents integrades per mitjà de xarxes d'interconnexió, per obtenir un sistema coordinat, capaç de processar una càrrega".

Cadascun dels ordinadors que formen part d'un clúster s'anomena "node". Els nodes fan servir un mateix sistema operatiu i és un middleware l'encarregat d'abstrure i virtualitzar els diferents nodes del sistema, donant la visió a l'usuari d'un sistema operatiu únic. El sistema operatiu dels nodes és estàndard, així que és el middleware qui proveeix de llibreries que permeten la computació paral·lela.

Podem distingir dos tipus de clústers: els d'alta disponibilitat i els d'alt rendiment.

Els **clústers d'alta disponibilitat** són aquells destinats a la necessitat de suportar un error de maquinari o programari. Si un dels nodes cau, la resta es reparteixen les seves tasques i intenten reactivar-lo. Aquest seria el cas de servidors web, ordinadors que han de prestar servei les 24h del dia els 365 dies de l'any.

Els **clústers d'alt rendiment** es basen en un conjunt de màquines configurades per aconseguir una capacitat de càlcul màxima al repartir-se la càrrega dels processos entre els nodes existents. Són utilitzats en la resolució d'algoritmes científics, reproducció d'imatges 3D, compilació de grans codis de programació, xifrat de contrasenyes, ... Els clústers d'alt rendiment funcionen amb una característica fonamental: el processament paral·lel, sobre el qual n'hem parlat en l'apartat anterior.

Grid computing

El concepte de Grid (malla) prové dels anys 1990, quan diversos investigadors van començar a plantejar una infraestructura de computació que proporcionàs accés sota demanda, segur, flexible als serveis de computació d'altres prestacions.

El nom de Grid es va prendre per analogia amb les xarxes elèctriques, amb les quals presenten alguns paral·lelismes, però també importants diferències. Inicialment el concepte de Grid es va aplicar com a un símil del fet que l'electricitat en si mateixa no va suposar una revolució, sinó més aviat la creació de la xarxa elèctrica que permetia un ús extensiu, barat i flexible de l'electricitat. Altres propietats, com l'escalabilitat en funció de la demanda són més pròpies de les xarxes elèctriques que de les xarxes de computació distribuïda.

Totes les implementacions de Grid Computing es basen en la comunicació a través d'Internet mitjançant diferents protocols.

Actualment existeixen dues aproximacions per a la implementació d'arquitectures en Grid: Arquitectures SOA i Arquitectures Peer-to-Peer.

Les **arquitectures SOA** (*Service-Oriented Architecture*, arquitectura orientada a serveis) estan basades en l'agregació de serveis als quals s'accedeix remotament de manera independent de les plataformes en les quals s'executen i del llenguatge en els quals estan implementats. Un servei és un programa autocontingut amb una funció i una interfície ben definida. Aquest tipus de serveis s'adapta molt bé als sistemes Grid ja que ofereix, entre altres avantatges, una alta portabilitat en permetre interactuar serveis de sistemes heterogenis a través d'interfícies que són independents de les plataformes on estan allotjats; interoperabilitat a través de protocols estàndard de xarxes; i permeten l'ús de clients lleugers que aïllen als consumidors de la complexitat dels serveis.

Les **arquitectures Peer-To-Peer** (o P2P) són agregacions de programes equivalents que s'executen sobre plataformes heterogènies i que comparteixen part de la seva memòria i capacitat de còmput. Aquestes plataformes presenten una alta tolerància a errors, atès que cada node té la mateixa funció que la resta. Aquest tipus d'arquitectures són més habituals en una mena de computació conegut com *Volunteer Computing*.

La tecnologia Grid permet d'una forma molt fàcil l'escalabilitat, ja que els nodes que actuen en ella són independents dels altres. Això permet que, si algun dels nodes falla, es poden delegar les tasques que estigués realitzant a la resta de nodes, la qual cosa permet seguir amb l'execució del programa i evitar aturades que puguin afectar l'efectivitat de l'obtenció dels resultats desitjats. A més, aquest tipus d'estructuració permet que es puguin escalar els recursos per a cada ordinador o node de manera independent en cas que fos necessari, sense haver d'afectar la resta d'ordinadors.

Cloud computing

La computació al nïgul, o *Cloud Computing*, és una de les tecnologies actuals de major implantació. El concepte de computació al nïgul suposa el següent pas evolutiu de la computació distribuïda superant els sistemes *legacy* i evolucionant cap als sistemes distribuïts a gran escala. L'objectiu d'aquest model informàtic és fer un millor ús dels recursos distribuïts, posar-los en comú per a aconseguir un major rendiment i poder abordar problemes de computació a gran escala.

La computació al nïgul no és un concepte completament nou per al desenvolupament i funcionament de les aplicacions web. Permet el desenvolupament més rendible de portals web escalables en infraestructures d'alta disponibilitat i a prova de fallades. En la computació al nïgul s'han d'abordar diferents reptes com: la virtualització, l'escalabilitat, la interoperabilitat, la qualitat del servei, la tolerància a errors i els models de nïgul.

Bàsicament, el nïgul pot definir-se mitjançant tres models diferents:

- **Nïgul públic:** Descriu la computació al nïgul en el sentit tradicional, on els recursos són facilitats de manera dinàmica sobre una base d'acte-servei a través d'aplicacions web/serveis web, des d'un proveïdor extern que comparteix recursos. Alguns exemples són Microsoft Azure, Google Cloud o, el més emprat de tots, Amazon Web Services (AWS).
- **Nïgul privat:** Les dades i processos es gestionen dins de l'organització sense les restriccions d'amplada de banda de la xarxa, els requeriments de seguretat i els requisits legals que suposa l'ús dels serveis del nïgul públic a través de xarxes públiques i obertes. Un exemple és Amazon Virtual Private Cloud (VPC).
- **Nïgul híbrid:** L'entorn està format per múltiples proveïdors interns i/o externs. Alguns exemples són RightScale, Asigra Hybrid Cloud Backup, Carpathia, Skytap i Elasta.

El *Cloud Computing* proporciona un entorn segur i senzill d'utilitzar, en el qual tant desenvolupadors com usuaris poden trobar una gran selecció de recursos que faciliten en gran manera la gestió de les aplicacions. Podem diferenciar entre tres grans grups de serveis de *cloud*: d'infraestructura, plataforma i programari.

Infraestructura (Infrastructure as a Service - IaaS)

És el sistema idoni per a desenvolupadors que desitgin encarregar-se de la gestió i administració de la seva infraestructura. Ofereix un major control que altres alternatives com PaaS, de manera que el desenvolupador és el responsable de tot allò relacionat amb el manteniment de la infraestructura, fins i tot de desplegar les seves aplicacions en funció de quines siguin les seves necessitats.

El millor exemple d'IaaS és Amazon Web Service (AWS). Es tracta d'una plataforma que ofereix una sèrie de serveis perquè els desenvolupadors puguin gestionar màquines virtuals al nívol, les quals també proporcionen espai d'emmagatzematge. Són els desenvolupadors els qui trien el sistema operatiu, Windows o Linux, així com la capacitat de memòria de cada màquina. El maquinari és 100% transparent, de manera que cada desenvolupador pot gestionar-lo de la manera que cregui més convenient.

Plataforma (Platform as a Service - PaaS)

Aquesta és l'alternativa idònia per a aquells desenvolupadors d'aplicacions que únicament volen preocupar-se de construir l'aplicació. La infraestructura la proporciona el proveïdor del servei i s'ocupa tant de la seva gestió com del seu manteniment.

A diferència d'IaaS, la construcció d'aplicacions i l'administració de la plataforma són molt senzilles. Les solucions PaaS gestionen de manera automàtica l'escalabilitat, fent ús d'un major nombre de recursos en cas que sigui necessari. Així i tot, els desenvolupadors han d'intentar que les seves aplicacions estiguin el millor optimitzades possibles per a no consumir massa recursos.

Un dels millors exemples de PaaS en l'actualitat és Google Colab, una plataforma per a ciència de dades basada en Python, i que veurem al llarg del curs, especialment en el mòdul de Programació d'Intel·ligència Artificial. Un altre exemple, també de Google, és Google App Engine, una plataforma en la qual els desenvolupadors poden crear i desplegar les seves aplicacions en Java o Python.

Programari (Software as a Service - SaaS)

Es defineix com SaaS a qualsevol servei que estigui basat en la web, com per exemple el Webmail de Gmail. En aquest cas els usuaris accedeixen al servei sense prestar la més mínima atenció al programari. Tant el desenvolupament com el manteniment i resta de gestions són responsabilitat única del proveïdor. Així, els usuaris tenen un control mínim sobre el servei en qüestió. Ells se situen en la capa més superficial d'aquest. Exemples de SaaS són Dropbox, Google Drive o les suites de Google Docs i de Microsoft Office 365 en la web.

3.1. Sistemes d'arxius distribuïts

El **sistema d'arxius (FS, File System)** és un subsistema d'un sistema operatiu que permet organitzar, recuperar i emmagatzemar dades dins arxius.

Una gran varietat d'aplicacions d'anàlisi de Big Data depenen d'entorns distribuïts per a analitzar grans quantitats de dades. A mesura que augmenta la quantitat de dades, la necessitat de proporcionar solucions d'emmagatzematge fiables i eficients s'ha convertit en una de les principals preocupacions dels administradors d'infraestructures de Big Data. Els sistemes i mètodes tradicionals d'emmagatzematge no són òptims a causa de les seves restriccions de preu o rendiment. És per això que s'han desenvolupat els **sistemes d'arxius distribuïts (DFS, Distributed File Systems)**. Un DFS és el sistema d'arxius d'un sistema distribuït, que permet l'emmagatzematge eficaç i fiable de grans volums de dades entre diversos ordinadors.

Una de les característiques principals d'un DFS és la **transparència**: ocultar als clients el fet que els processos i els recursos estan distribuïts físicament en la xarxa i proporcionar una visió comuna d'un sistema d'arxius centralitzat. És a dir, el DFS pretén ser invisible per als clients, que consideren que el DFS és similar a un sistema d'arxius local.

Un altre factor important és la **fiabilitat**. Atès que els arxius del DFS estan distribuïts per la xarxa, la possibilitat de fallades és major que en els sistemes d'arxius locals. La **replicació**, és la principal tècnica per a aconseguir una alta fiabilitat: els DFS fan diverses còpies d'un arxiu de dades (sovint després d'haver-los fragmentat en troços més petits) en diferents servidors. Quan un client sol·licita l'arxiu, accedeix de manera transparent a una de de les còpies. Dins de l'estratègia de replicació, la ubicació de les rèpliques juga un paper crític, tant pel que fa al rendiment, com a la fiabilitat del DFS. Així doncs, les rèpliques s'emmagatzemen en diferents servidors seguint el que s'anomena un esquema de col·locació.

El DFS més utilitzat avui en dia és **HDFS**, el DFS de Hadoop, sobre el qual parlarem més endavant.

4. Apache Hadoop

Apache Hadoop és un *framework* (entorn de treball) de codi obert que admet l'emmagatzematge i el processament distribuïts de grans conjunts de dades en clústers d'ordinadors que usen models de programació simples. Hadoop està dissenyat per a escalar verticalment des d'un únic ordinador fins a milers d'ordinadors agrupats en clústers, on cada màquina ofereix processament i emmagatzematge local. D'aquesta manera, Hadoop pot emmagatzemar i processar de manera eficient grans conjunts de dades que varien en grandària, des de gigabytes fins a petabytes de dades.

Apache Hadoop és la solució per a sistemes distribuïts més utilitzada avui en dia. Tant que computació distribuïda i Hadoop s'han arribat a convertir pràcticament en sinònims.

Hadoop va ser desenvolupat per Dough Cutting i Mike Cafarella en el marc d'un projecte de cercador web de codi obert anomenat Nutch. Per tal d'emmagatzemar i processar dades de manera distribuïda i automatitzada i aconseguir resultats de cerques a una major velocitat varen inspirar-se en dues solucions que ja emprava Google: el model de programació MapReduce i el Google File System (GFS).

L'any 2006 Dough Cutting passa a fer feina a Yahoo i se'n du el projecte Nutch amb ell. Dins Yahoo el proyecto Nutch va ser dividit en dos: el cercador web es va mantenir com a Nutch, i la part de processament distribuït es va convertir en el projecte Hadoop. El nom de Hadoop prové del nom de l'elefant de jugueta que tenia el fill de Dough Cutting, i per això el logo de Hadoop és un elefantet groc.

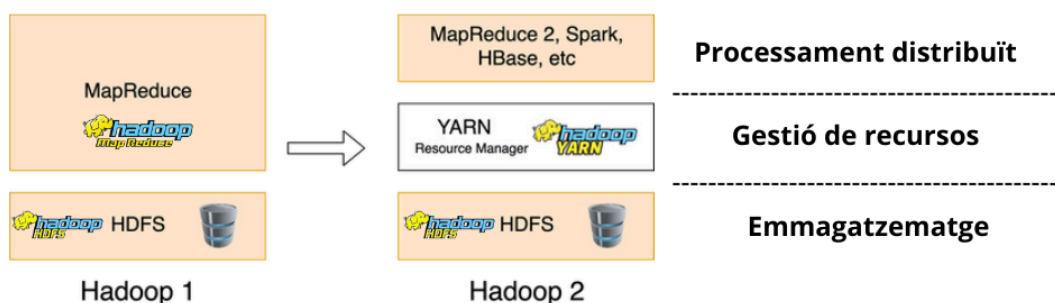
En 2008 Yahoo converteix Hadoop en un projecte de codi obert, que posteriorment va passar a ser gestionat per l'Apache Software Foundation. Des de 2012 el nom oficial és Apache Hadoop.

A més de Yahoo, moltes grans empreses utilitzen Hadoop per al processament distribuït, entre elles Facebook, Twitter o LinkedIn. A més, Hadoop està suportat a les principals plataformes del núvol: AWS, Azure i Google Cloud.

El nucli principal de Hadoop constava originalment de tres components principals (*core components*):

- **HDFS** (Hadoop Distributed File System): és el sistema d'arxius distribuït que proporciona accés d'alta capacitat de processament a dades d'aplicacions sense necessitat de definir esquemes amb anterioritat.
- **MapReduce**: és un model de programació per al processament de dades a gran escala. Mitjançant algorismes de processament distribuït i paral·lel, MapReduce fa que sigui possible executar la lògica de processament i ajuda a escriure aplicacions que transformen grans conjunts de dades en conjunts manejables.
- **Hadoop Common**: inclou les biblioteques i les utilitats que usen i comparteixen altres mòduls de Hadoop.

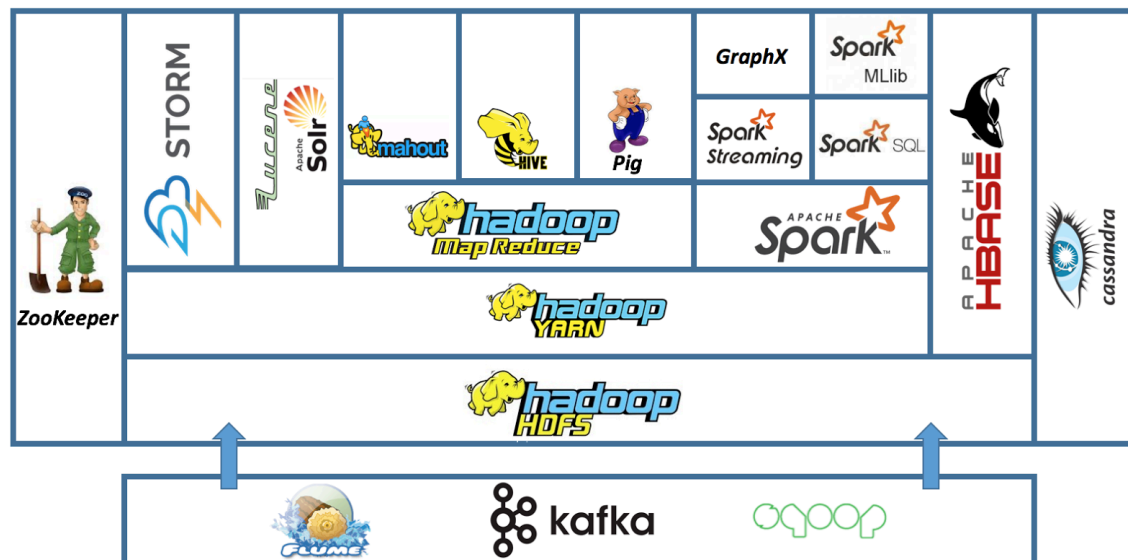
En la versió 2 de Hadoop s'introdueix un quart component principal al nucli, anomenat **YARN** (Yet Another Resource Negotiator): és un framework d'administració de recursos que permet programar tasques i assignar recursos en el sistema Hadoop. YARN permet a Hadoop suportar diversos motors d'execució, incloent MapReduce, però també d'altres com Apache Spark, que no està basat en Java sinó en Scala i que proporciona APIs per a diversos llenguatges: Java, Scala, Python i R. Parlarem amb més detall sobre YARN, i també sobre Apache Spark, en altres lliuraments d'aquest mòdul.



Imatge: Evolució de Hadoop 1 a Hadoop 2

Per altra banda, sobre aquest nucli principal s'han desenvolupat moltes més eines per a diferents propòsits: càrrega de dades, interfície SQL per a les dades, base de dades NoSQL, administració del clúster, *machine learning*, etc. La majoria d'aquestes eines són projectes de codi obert de l'Apache Software Foundation, però també n'hi ha que són software propietari. Tot junt, el nucli de Hadoop i el conjunt d'aquestes eines formen l'anomenat **ecosistema Hadoop**, que fan que Hadoop sigui tan potent i popular avui en dia.

La següent imatge mostra algunes (n'hi ha més!) de les eina de l'ecosistema Hadoop:



Imatge: Ecosistema Hadoop. Font: tutorials.freshersnow.com

De vegades s'utilitza el terme Hadoop per referir-se als components principals, però més sovint, a tot l'ecosistema.

5. HDFS

HDFS (Hadoop Distributed File System) és el sistema d'arxius distribuït (DFS) que constitueix el nucli de l'ecosistema Hadoop i s'ha convertit en el DFS més utilitzat actualment. Quan usam HDFS, les dades es reparteixen entre molts nodes, de vegades desenes de milers, que poden estar situats en diversos centres repartits per tot el món. Té moltes similituds amb altres sistemes de fitxers distribuïts existents, tot i que també té algunes diferències importants. HDFS proporciona un accés d'alt rendiment a les dades de l'aplicació i és adequat per a aplicacions que fan feina amb grans conjunts de dades. A més, és altament tolerant a fallades i està dissenyat per implementar-se en maquinari de baix cost.

HDFS es va crear originalment com a infraestructura per al projecte del motor de cerca web Apache Nutch, i avui en dia és un subprojecte Apache Hadoop.

5.1. Arquitectura d'HDFS

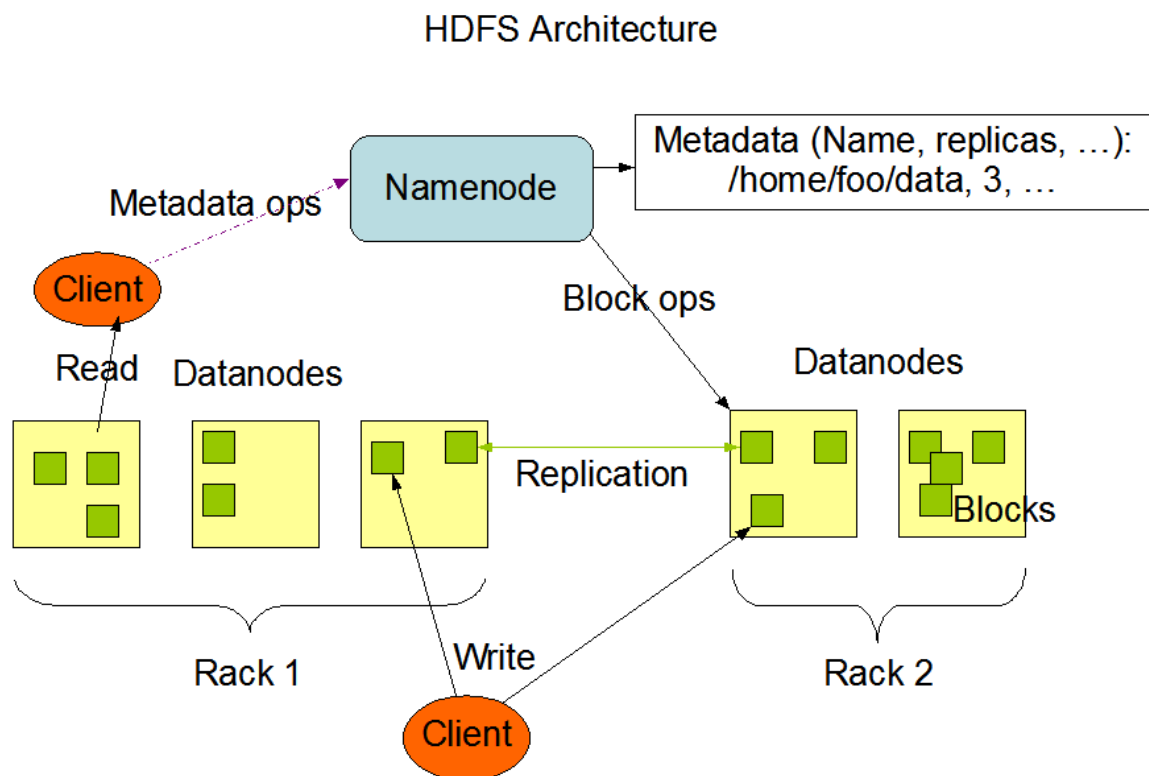
HDFS té una arquitectura mestre/esclau.

Un clúster HDFS consta d'un únic **NameNode**, un servidor mestre que gestiona les peticions dels ordinadors clients, organitza l'espai d'emmagatzematge i la ubicació de les dades. També controla les operacions bàsiques amb fitxers (com obrir-los o tancar-los) i controla l'accés a les dades per part dels clients. Per altra banda, hi ha una sèrie d'esclaus, anomenats **DataNodes**, normalment un per a cada node del clúster. Els DataNodes s'encarreguen d'emmagatzemar les dades i, per fer-ho, creen, esborren i copien blocs quan és necessari.

HDFS exposa un espai de noms del sistema de fitxers i permet que les dades de l'usuari s'emmagatzemin en fitxers. Internament, un fitxer es divideix en un o més blocs. I cada un d'aquests blocs s'emmagatzema, de manera replicada, en diversos DataNodes. És fonamental guardar diverses còpies de cada bloc, per al cas en què un node pugui fallar.

El NameNode segueix el rastre dels DataNodes que fallen mitjançant la recepció d'uns missatges que aquests li envien cada 3 segons, anomenats batecs (*heartbeats*). Si el NameNode deixa de rebre batecs d'un DataNode, assumeix que aquest ha deixat de funcionar. D'aquesta manera, quan s'hagi de llegir un bloc emmagatzemat al DataNode caigut, s'empraran la resta de DataNodes que tenen còpia del bloc. A més del batec, els DataNodes també envien periòdicament un informe de blocs (*blockreport*), una llista dels blocs que conté el DataNode.

Una de les funcions del NameNode consisteix en determinar quin DataNode convé utilitzar cada vegada, depenent de l'estat actual, la qual cosa assegura la rapidesa en l'accés a les dades i el seu tractament. A continuació, el client accedirà al bloc de dades emmagatzemat al DataNode seleccionat.



Imatge: Arquitectura d'HDFS. Font: hadoop.apache.org

Per altra banda, aquest model permet una escalabilitat horitzontal, ja que es poden anar afegint DataNodes nous a mesura que el volum creixent de dades ho requereix. Afegir nous DataNodes és barat i no exigeix canvis en els nodes preexistents.

Tots els blocs d'un fitxer (excepte el darrer) tenen la mateixa mida. Per defecte, els blocs són de 64 Mb en Hadoop 1.0 i de 128 Mb des de la versió 2.0. Per altra banda, el factor de replicació per defecte és 3: cada bloc està copiat a 3 DataNodes diferents. Aquestes dos paràmetres es poden modificar manualment. A més, una aplicació pot establir que un fitxer tingui un número determinat de rèpliques. El factor de replicació d'un fitxer pot ser especificat en el moment de la seva creació i modificat posteriorment. Un factor important de Hadoop és que els blocs s'escriuen només una vegada (son *write-once*), tot i que lògicament, es poden llegir moltes vegades.

Rack-awareness

Ja hem dit que un clúster HDFS està format per molts nodes. Però normalment, els nodes no estan tots dins la mateixa xarxa local, sinó que estan distribuïts en diferents llocs físics.

Un **rack** (prestatge) és una col·lecció física de nodes d'un clúster, que estan dins la mateixa xarxa. La connexió entre nodes de diferents racks ha de passar a través de switches. D'aquesta manera, l'ample de banda entre dos nodes del mateix rack és molt més gran que entre nodes de diferents racks.

El NameNode manté la informació sobre a quin rack pertany cada DataNode, és a dir, sap quin és el *rack id* de cada DataNode. Per això es diu que Hadoop és **rack-aware**. I aquesta informació és fonamental a l'hora de seleccionar on situar els blocs.

Si el factor de replicació és 3, una primera aproximació seria col·locar cada bloc en nodes de 3 racks diferents. Això incrementa la seguretat, ja que és tolerant a caigudes de racks sencers. Però, aquesta política incrementa el cost de les escriptures, ja que s'han de transferir blocs a múltiples racks.

Així que, per tal d'assegurar una bona tolerància a fallades a la vegada que un alt rendiment, HDFS segueix aquestes tres regles:

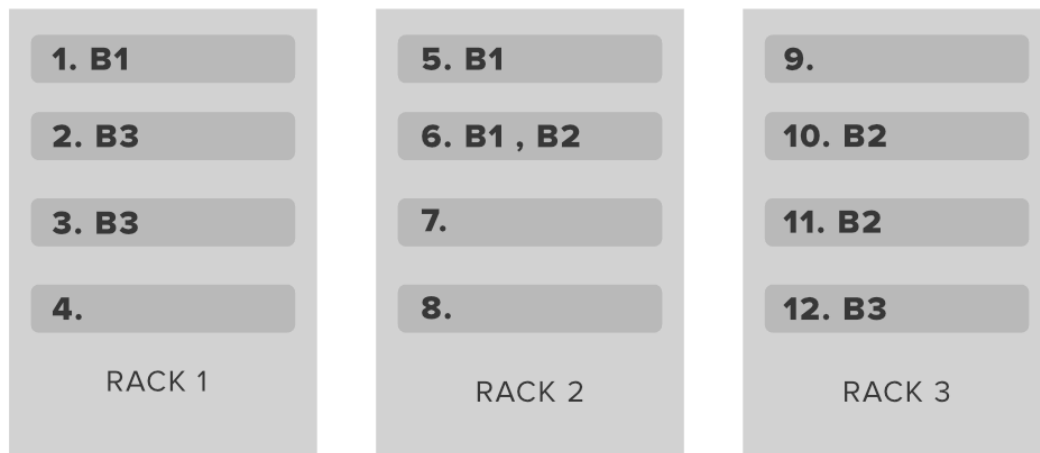
- No es permet mai més d'una rèplica d'un bloc en el mateix DataNode
- No es permeten més de 2 rèpliques d'un bloc en el mateix rack: un bloc es guardarà com a mínim a 2 racks
- Per a cada bloc, s'han d'utilitzar menys racks que el número de rèpliques que s'hi facin: almenys un rack tindrà més d'una rèplica

Hem de tenir en compte que el risc d'una fallada d'un rack sencer és molt menor que el de la fallada d'un node individual.

Si aplicam aquestes regles al cas habitual d'un factor de replicació de 3, tenim que HDFS col·loca una rèplica en el rack local, una altra en un node d'un altre rack, i la tercera en un altre node del rack local. És a dir, dues rèpliques en dos nodes diferents d'un mateix rack i una altra rèplica en un node d'un segon rack. Aquesta política millora el rendiment de les escriptures sense comprometre la fiabilitat de les dades ni el rendiment de les lectures.

Replica Placement via Rack Awareness:

==> Block 1 , Block 2 , Block 3



Imatge: Exemple de col·locació de blocs en nodes i racks. Font: www.geeksforgeeks.org

5.2. Accés a HDFS

Hi ha diverses maneres d'interactuar amb HDFS.

FS Shell

HDFS proporciona una interfície de línia d'ordres anomenada **FS Shell** que permet a l'usuari interactuar amb els fitxers a HDFS. La sintaxi d'aquest conjunt d'ordres és similar a la d'altres shells (per exemple, bash, csh), que els usuaris ja coneixen. En l'apartat següent veurem les ordres més habituals del FS Shell.

Interfície web

Una instal·lació típica de Hadoop inclou un servidor web que exposa HDFS a través d'un port TCP. Això permet a l'usuari navegar per l'espai de noms HDFS i veure el contingut dels directoris i fitxers utilitzant un navegador web.

API

Hadoop proporciona una API Java de manera nativa: <https://hadoop.apache.org/docs/stable/api/>

Part d'aquesta API (el paquet `org.apache.hadoop.fs` i els seus subpaquets) permet interactuar amb HDFS.

5.3. FS Shell

L'interpret d'ordres del Sistema de Fitxers (FS Shell) inclou diverses ordres que permeten interactuar directament amb HDFS, així com també amb altres sistemes de fitxers que Hadoop suporta, com ara Local FS, HFTP FS, S3 FS, i altres. Les ordres de FS Shell són invocades mitjançant:

```
hdfs dfs <args>
```

o també:

```
hadoop fs <args>
```

Per exemple, la següent ordre:

```
hdfs dfs -mkdir prova
```

que també es pot escriure:

```
hadoop fs -mkdir prova
```

crea un directori anomenat *prova*.

La diferència entre emprar *hadoop fs* o *hdfs dfs* és que el primer fa referència a un sistema de fitxers genèric, que pot ser HDFS, però també altres suportats per Hadoop (com Local FS, S3, etc.). En canvi, *hdfs dfs* serveix per interactuar específicament amb HDFS. Com que en aquest lliurament només xerrem de HDFS, emparem *hdfs dfs*, tot i que podríem emprar qualsevol dels dos, ja que en el nostre cas, serien equivalents.

Totes les ordres de FS Shell prenen URIs com a arguments. El format de les URIs és *esquema://autoritat/ruta*. Per a HDFS, l'esquema és *hdfs*, i per a Local FS l'esquema és *file*. Tant l'esquema com l'autoritat són opcionals i, si no s'especifiquen, s'utilitza l'esquema configurat per defecte.

Per exemple, un fitxer o directori HDFS com */pare/fill* es pot especificar com a *hdfs://namenode_host/pare/fill* (on *namenode_host* és la IP o nom de domini del NameNode) o simplement com */pare/fill* (si la configuració per defecte està establerta per apuntar a *hdfs://namenode_host*).

La majoria de les ordres de FS Shell es comporten com les ordres Unix corresponents, de la següent manera:

```
hdfs dfs -<ordre UNIX>
```

Per exemple, ja hem vist que per crear un directori, es fa servir *hdfs dfs -mkdir prova*, ja que *mkdir* és l'ordre corresponent de Unix. La informació d'error s'envia a *stderr* i la de sortida s'envia a *stdout*.

Vegem alguns exemples de les ordres emprades per a les operacions més habituals amb fitxers i directoris dins HDFS:

Crear un directori:

```
hdfs dfs -mkdir URI_directori
```

Per exemple:

```
hdfs dfs -mkdir prova
```

Copiar un fitxer del sistema d'arxius local a HDFS:

```
hdfs dfs -put path_fitxer_local URI_directori
```

Per exemple, la següent ordre copia el *fitxer.txt* que està al directori local */home/hadoop* al directori *prova* d'HDFS:

```
hdfs dfs -put /home/hadoop/fitxer.txt prova
```

En lloc de `-put` també es pot emprar `-copyFromLocal`, són idèntics

Copiar un fitxer d'HDFS al sistema d'arxius local

```
hdfs dfs -get URI_fitxer path_directori_local
```

Per exemple, la següent ordre copia el fitxer *fitxer.txt* del directori d'HDFS *prova* al directori local */home/hadoop*:

```
hdfs dfs -get prova/fitxer.txt /home/hadoop
```

En lloc de `-get` també es pot emprar `-copyToLocal`, són idèntics

Veure el contingut d'un directori:hdfs

```
hdfs dfs -ls URI_directori
```

Entre d'altres, tenim els paràmetres `-t` per ordenar per temps de modificació (el més recent primer), `-u` per ordenar per temps del darrer accés (el més recent primer), `-S` per ordenar per grandària, `-r` per invertir l'ordre.

Veure el contingut d'un fitxer (enviar una còpia del fitxer a stdout):

```
hdfs dfs -cat URI_fitxer
```

Moure un fitxer a un altre directori:

```
hdfs dfs -mv URI_fitxer URI_directori_destinació
```

Copiar un fitxer a un altre directori:

```
hdfs dfs -cp URI_fitxer URI_directori_destinació
```

Esborrar un fitxer:

```
hdfs dfs -rm URI_fitxer
```

Esborrar un directori recursivament:

```
hdfs dfs -rm -r URI_directori o hdfs dfs -rm -R URI_directori
```

Pots trobar un llistat complet de totes les ordres de FS Shell a <https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html>

6. Configuració d'un clúster Hadoop

En aquest capítol veurem com crear un clúster Hadoop.

Per fer-ho, no emprarem màquines físiques, sinó que farem servir màquines virtuals en Oracle VirtualBox. En concret, anam a crear un clúster de 3 nodes, un NameNode i dos DataNodes.

Hem de tenir clar que l'objectiu d'aquesta activitat és entendre com es crea i funciona un clúster Hadoop. Però no pretenem que aquest clúster pugui ser utilitzat en un entorn real de producció, on el número de nodes que formarien el clúster, i sobretot, els requeriments de RAM i processador de cada un dels nodes, haurien de ser molt superiors.

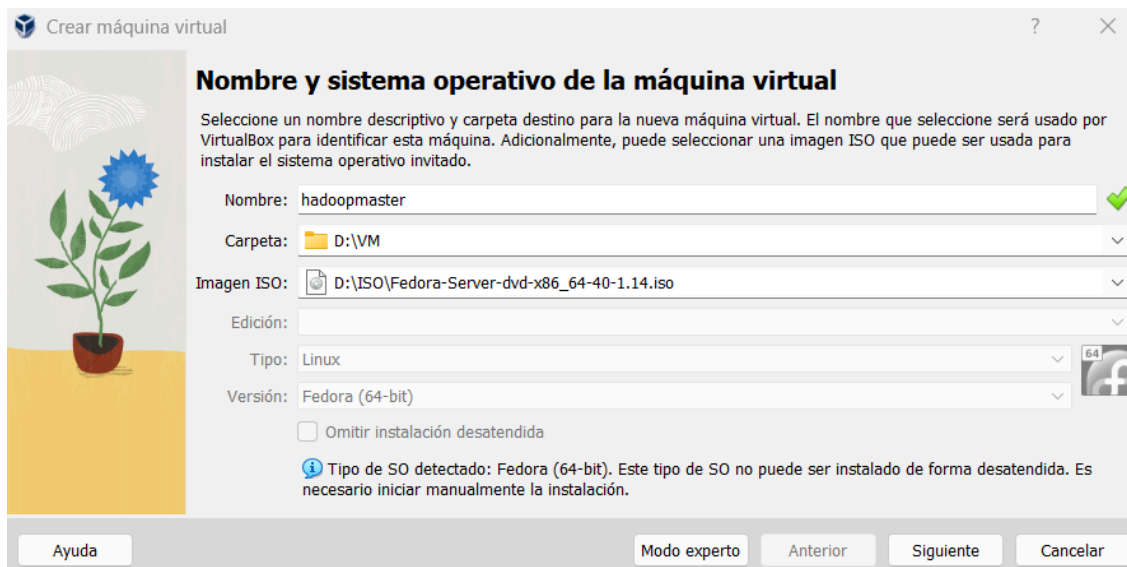
6.1. Preparar l'entorn

En primer lloc, si no el tenim ja, haurem de descarregar i instal·lar [Oracle VirtualBox](#). Es recomana emprar la darrera versió disponible, que actualment (setembre de 2024) és la 7.0.20.

I després anam a descarregar la ISO de la [darrera versió de Fedora Server](#), que actualment (setembre de 2024) és la 40.

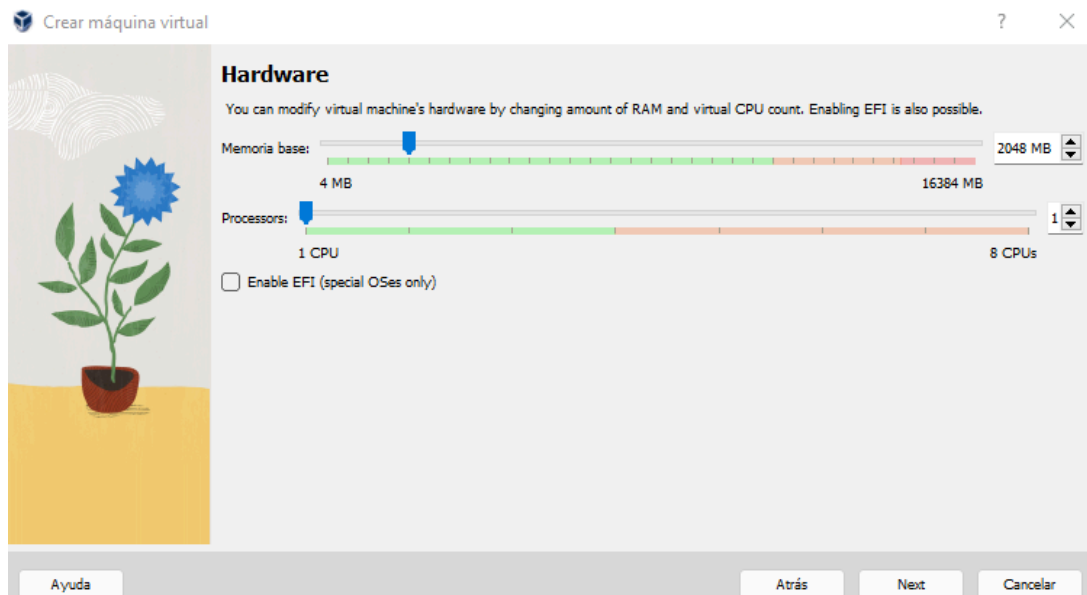
Ara ja podem crear una primera màquina virtual, que serà el node mestre del nostre clúster, el NameNode. Una vegada la tinguem configurada, en farem dos clons, que modificarem convenientment perquè passin a ser els dos nodes esclaus.

Així doncs, crearem una nova màquina virtual, a la qual li posarem el nom *hadoopmaster*. I seleccionarem la ISO de Fedora Server 40 que hem descarregat anteriorment.



Imatge: Creació de la màquina virtual per al node mestre

Per poder executar totes les màquines en el mateix PC, anam a assignar uns recursos mínims: només un processador i 2 Gb de RAM (2048 Mb). Si la memòria del teu ordinador no és suficient per suportar 2 Gb de RAM per a cada màquina, pots provar de reduir aquest número. Tal com hem comentat, aquests recursos serien insuficients per executar aplicacions en producció sobre el clúster, però ens bastarà per fer les nostres proves. En un entorn real es recomana disposar de 64 Gb de RAM per a un ús bàsic o normal, o 128 Gb per un ús avançat, i un mínim de 8 cores, tot i que és més habitual emprar-ne 16 (o fins i tot 24).



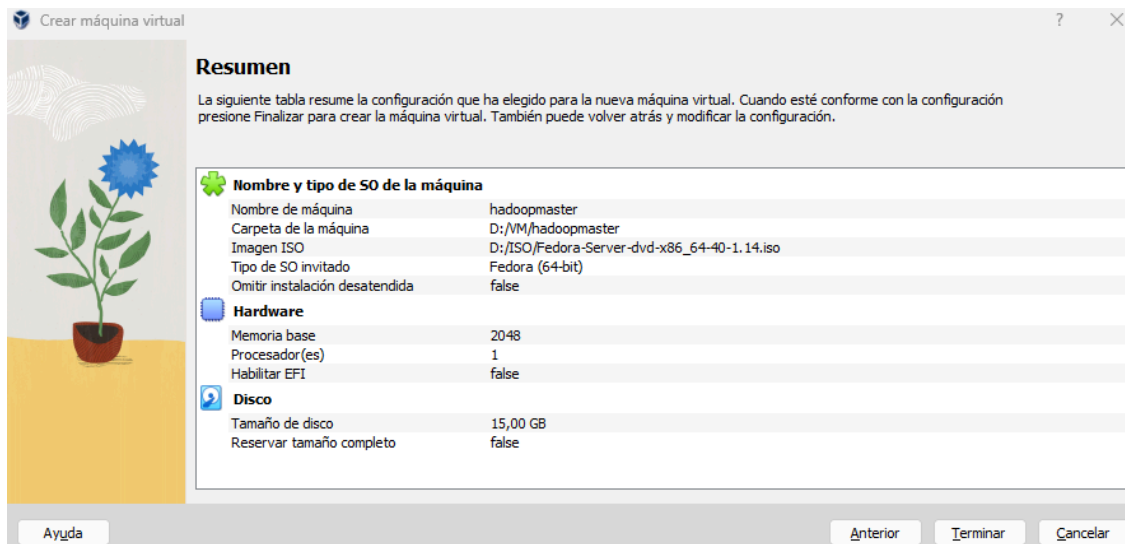
Imatge: Configuració de RAM i processadors

Per a l'espai de disc, definirem un disc dur virtual de 15 Gb, el valor que ens dona per defecte. De nou, en un entorn real, es recomana un mínim de 500 Gb, o preferiblement 1 Tb.



Imatge: Configuració de disc dur

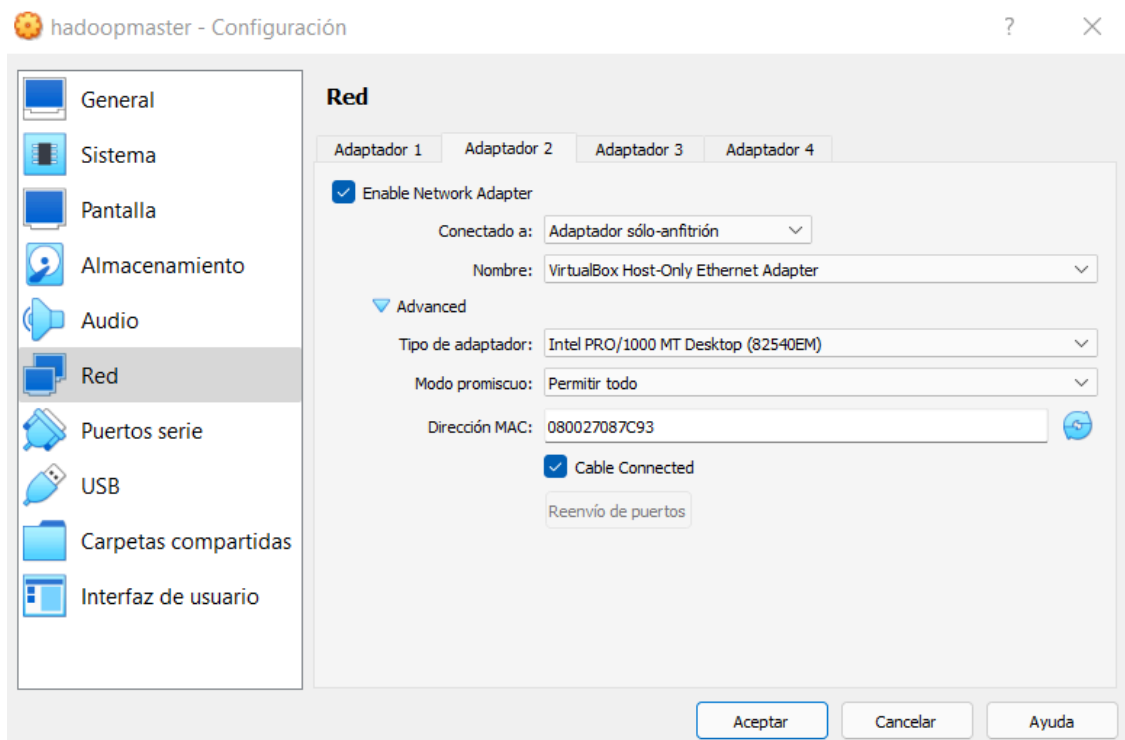
Ara apareix el resum de les característiques que hem seleccionat:



Imatge: Resum de característiques

I ara ja podem crear la màquina (botó "Terminar").

Abans d'iniciar-la, hem de configurar els adaptadors de xarxa, que en tindrem dos. Un de tipus NAT, que ens donarà accés a l'exterior, i un altre de tipus "Adaptador sólo anfitrión" ("Host-only adapter"), per a comunicar-se els tres nodes entre sí. En aquest segon adaptador, hem de seleccionar el nom "VirtualBox Host-Only Ethernet Adapter" i hem d'activar el mode promiscu (seleccionant "Permitir todo").



Imatge: Configuració del segon adaptador de xarxa

I a continuació ja podem iniciar la màquina i instal·lar-hi el sistema operatiu. En el procés d'instal·lació (podeu emprar l'idioma que vulgueu), haureu de seleccionar el disc virtual que hem definit prèviament com a destinació de la instal·lació. I també activar el compte de root i assignar-li una contrasenya. Tot i que no és una bona pràctica en un entorn real, per simplificar la gestió podem permetre l'accés per SSH a l'usuari root amb contrasenya.

Una vegada ha arrancat la màquina, instal·larem el JDK de Java. Aquí ho farem amb l'Open JDK 8 (tot i que no hauria problema en emprar una versió més nova):

```
yum install java-1.8.0-openjdk-devel -y
```

Podem comprovar que ha quedat ben instal·lat executant:

```
java -version
```

Per a les passes següents, podem seguir fent feina des del terminal de la mateixa màquina, o bé obrir-hi una connexió SSL. Per fer-ho, mirarem la IP que s'ha assignat al segon adaptador de xarxa (*enp0s8*) amb *ip a*:

```
root@localhost:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:40:d9:ab brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 85279sec preferred_lft 85279sec
    inet6 fe80::a00:27ff:fe40:d9ab/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7c:28:d4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.118/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s8
        valid_lft 379sec preferred_lft 379sec
    inet6 fe80::a00:27ff:fe7c:28d4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Imatge: Adreces IP

En aquest cas, veim que se'ns ha assignat l'adreça 192.168.56.118. Si tenim una altra màquina virtual amb un adaptador de xarxa "sólo anfitrión" amb el mateix nom, podem connectar-nos a la nostra màquina per SSH:

```
toni@ubuntu1:~$ ssh root@192.168.56.118
root@192.168.56.118's password:
Web console: https://localhost:9090/ or https://10.0.2.15:9090/

Last login: Mon Sep  9 12:31:43 2024 from 192.168.56.112
root@localhost:~#
```

Imatge: Accés per SSH al node mestre

6.2. Instal·lar Hadoop en el node mestre

Per instal·lar Hadoop, és recomanable crear un nou usuari (li direm *hadoop*, amb privilegis de sudo), que l'utilitzarem com un "super usuari" de Hadoop.

```
adduser hadoop
passwd hadoop
usermod -aG wheel hadoop
```

Canviarem a l'usuari hadoop:

```
su - hadoop
```

Primer, anam a generar una clau per a SSH i l'afegirem a les claus autoritzades, per permetre connexions SSH a la màquina. Ho farem amb les següents ordres (quan generam la clau, podem assignar-li una paraula de pas o no, com vulguem):

```
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```

Ara anam a descarregar els binaris de Hadoop. Per fer-ho, miram a <https://hadoop.apache.org/releases.html> quina versió volem. Podem descarregar la darrera disponible, actualment la 3.3.4, que es troba a <https://d1cdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz>. Descarregam el fitxer i el descomprimim:

```
wget https://d1cdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz
tar xzf hadoop-3.4.0.tar.gz
```

A contiunació, canviarem el nom del directori de Hadoop, de *hadoop-3.4.0* a *hadoop*. No és necessari, però facilita una mica algunes coses.

```
mv hadoop-3.4.0 hadoop
```

Ara hem d'afegir algunes variables d'entorn en el fitxer *~/.bashrc*. L'editarem amb *vi* i afegirem al final:

```
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

Per modificar l'entorn actual, hem d'executar:

```
source ~/.bashrc
```

Podem executar la següent ordre per assegurar-vos que els canvis s'han fet correctament:

```
ls $HADOOP_HOME
```


El resultat hauria de ser aquest:

```
Uhadoop@localhost ~]$ ls $HADOOP_HOME
bin  include  libexec  licenses-binary  NOTICE-binary  README.txt  share
etc  lib      LICENSE-binary  LICENSE.txt      NOTICE.txt     sbin
```

Imatge: Directori \$HADOOP_HOME

Abans de continuar configurant Hadoop, anam a canviar el nom de la màquina. Li direm **hadoopmaster**. Per fer-ho, podem editar el fitxer **/etc/hostname**, o bé executar l'ordre:

```
sudo hostnamectl set-hostname hadoopmaster
```

Podem comprovar que el canvi s'ha fet correctament, executant l'ordre següent, que ens hauria de retornar *hadoopmaster*:

```
hostnamectl
```

6.3. Configurar Hadoop en el node mestre

En aquest apartat modificarem alguns fitxers de configuració de Hadoop, ubicats en el directori `$HADOOP_HOME/etc/hadoop`. Primer, seguint amb l'usuari *hadoop*, accedirem al directori:

```
cd $HADOOP_HOME/etc/hadoop
```

I començarem editant el fitxer **hadoop-env.sh**. Podem emprar l'editor *vi*, o bé instal·lar *nano* (amb `sudo yum install nano`). Hem d'afegir la següent línia al final del fitxer, amb la ubicació de l'Open JDK:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

Al fitxer **core-site.xml**, hem d'afegir la següent propietat dins de l'element *configuration* per establir la URI d'HDFS (`hdfs://hadoopmaster:9000`, on *hadoopmaster* és el nom de la nostra màquina):

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://hadoopmaster:9000</value>
  </property>
</configuration>
```

Al fitxer **hdfs-site.xml**, hem d'afegir les següents propietats per establir el factor de replicació (3) i els directoris del namenode i datanode:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
```

```
  <property>
    <name>dfs.name.dir</name>
    <value>/home/hadoop/hadoopdata/hdfs/namenode</value>
  </property>
```

```
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/hadoopdata/hdfs/datanode</value>
  </property>
</configuration>
```

Al fitxer **mapred-site.xml**, hem d'afegir la següent propietat per establir YARN com el framework de mapreduce:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Finalment, al fitxer **yarn-site.xml**, hem d'afegir les següents propietats relacionades amb YARN:

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
```

```
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
```

```
  <property>
    <name>yarn.resourcemanager.schedule.address</name>
    <value>hadoopmaster:8030</value>
  </property>
```

```
  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>hadoopmaster:8031</value>
  </property>
```

```
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>hadoopmaster:8032</value>
  </property>
```

```
  <property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>hadoopmaster:8033</value>
  </property>
</configuration>
```

6.4. Engegar Hadoop amb un únic node

Ara que ja hem configurat el node mestre, podem arrencar-hi Hadoop, de manera que funcioni amb un únic node, que contindrà tant el NameNode com un DataNode.

Abans, però, hem de donar format al sistema d'arxius HDFS. Això crearà totes les metadades necessàries per al funcionament del NameNode. Hem d'executar la següent ordre, amb l'usuari *hadoop*:

```
hdfs namenode -format
```

I ara ja sí que podem executar el script (amb l'usuari *hadoop*) que arrenca tots els dimonis de Hadoop:

```
start-all.sh
```

```
[hadoop@hadoopmaster ~]$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [hadoopmaster]
Starting datanodes
Starting secondary namenodes [hadoopmaster]
Starting resourcemanager
Starting nodemanagers
```

Imatge: Arrencar Hadoop



L'ús de *start-all.sh* (i *stop-all.sh* per aturar Hadoop) està *deprecated* en les darreres versions de Hadoop. En el seu lloc, es recomana iniciar els dimonis de HDFS i YARN de forma separada amb *start-dfs.sh* (i *stop-dfs.sh* per aturar-lo) i *start-yarn.sh* (i *stop-yarn.sh* per aturar-lo).

Per simplicitat, aquí seguirem utilitzant *start-all.sh* i *stop-all.sh*.

Si ara miram els processos Java amb *jps*, podem veure que s'estan executant un **NameNode** i un **DataNode**:

```
[hadoop@hadoopmaster hadoop]$ jps
3941 SecondaryNameNode
4277 NodeManager
4645 Jps
3737 DataNode
4157 Resourcemanager
3615 NameNode
```

Imatge: Processos Java al node mestre (únic node del clúster)



Si algun d'aquests processos no ha arrancat, és probable que tinguem algun error copiant les propietats dels fitxers de configuració. Podem veure els fitxers de log que estan a **\$HADOOP_HOME/logs** per trobar una pista del que està succeint.

Un problema freqüent, especialment si hem modificat els directoris per al NameNode i DataNode, és que el directori del NameNode quedi en un estat inconsistent. En aquest cas, el més senzill és esborrar tot el contingut del sistema de fitxers HDFS. Lògicament això no es fa en un sistema en producció perquè estaríem perdent totes les dades!

Per fer-ho, primer aturam Hadoop (amb l'usuari *hadoop*):

```
stop-all.sh
```

A continuació, si ja hem escrit alguna cosa al HDFS, hauríem d'esborrar tots els fitxers i directoris que tinguem al directori del DataNode (recordem que ho hem especificat al fitxer *hdfs-sites.xml* i hem posat */home/hadoop/hadoopdata/hdfs/datanode*).

I després haurem de formatejar el NameNode, cosa que elimina totes les metadades d'HDFS:

```
hadoop namenode -format
```

Per últim, tornam a engegar Hadoop:

```
start-all.sh
```

Un altre procés que sempre trobarem al node mestre és el **SecondaryNameNode**. Tot i que pel seu nom pot semblar que es tracta d'un node de backup per si cau el NameNode, no és exactament això.

En el node mestre hi ha dos fitxers que es fan servir per gestionar les metadades que representen l'estructura de fitxers i directoris d'HDFS. Per un costat tenim el fitxer de "Edit logs" que guarda un registre de tots els canvis que s'han fet a l'espai de noms d'HDFS. Per l'altra banda, tenim també el fitxer *FsImage*, que guarda una instantània (snapshot) de les metadades HDFS en un determinat moment. El procés NameNode va actualitzant contínuament aquests dos fitxers. El que fa el procés SecondaryNameNode és guardar la darrera còpia d'aquests dos fitxers, per si hagués un error amb els originals, poder recuperar-la.

També podem veure dos processos més, ambdós relacionats amb YARN: **NodeManager** i **ResourceManager**. El primer és l'encarregat de gestionar un node d'un clúster Hadoop, així que el trobarem tant al mestre com als esclaus. El segon és específic del node mestre i és l'encarregat de comunicar-se amb els diferents NodeManagers del clúster i assignar els recursos (nodes) a les peticions dels clients. Podeu trobar més detalls a <https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html>, tot i que en tornarem a xerrar en el pròxim lliurament.

Ara que ja tenim el clúster en marxa (amb un únic node), podem fer algunes proves, executant algunes comandes d'HDFS. Comencem creant un nou directori *prova*:

```
hdfs dfs -mkdir prova
```

Veurem que això ens dona un error:

```
hadoop@hadoopmaster:~/hadoopdata/hdfs$ hdfs dfs -mkdir prova
mkdir: `hdfs://hadoopmaster:9000/user/hadoop': No such file or directory
```

Imatge: Error creant el primer directori

El problema és que encara no existeix el directori */user/hadoop* on s'ha de crear el nou directori *prova*. Per solucionar-ho, només la primera vegada que cream un directori en HDFS, farem servir el *flag -p*, que indica que tots els directoris no existents es crearan també:

```
hdfs dfs -mkdir -p prova
```

Ara anam a pujar un fitxer a la nostra màquina i després el copiarem dins d'aquest directori *prova* d'HDFS. Primer anam a descarregar un fitxer CSV amb les dades dels estats d'Estat Units (des del directori arrel de l'usuari *hadoop*: */home/hadoop*)

```
wget https://raw.githubusercontent.com/plotly/datasets/master/2014_usa_states.csv
```

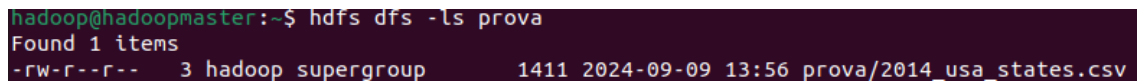
I ara el copiam al directori test d'HDFS mitjançant l'ordre *put*:

```
hdfs dfs -put 2014_usa_states.csv prova
```

I comprovam que s'ha copiat, mirant el contingut del directori test:

```
hdfs dfs -ls prova
```

I aquest és el resultat:



```
hadoop@hadoopmaster:~$ hdfs dfs -ls prova
Found 1 items
-rw-r--r--   3 hadoop supergroup      1411 2024-09-09 13:56 prova/2014_usa_states.csv
```

Imatge: *Contingut del directori prova*

El 3 que apareix indica que el factor de replicació és 3.

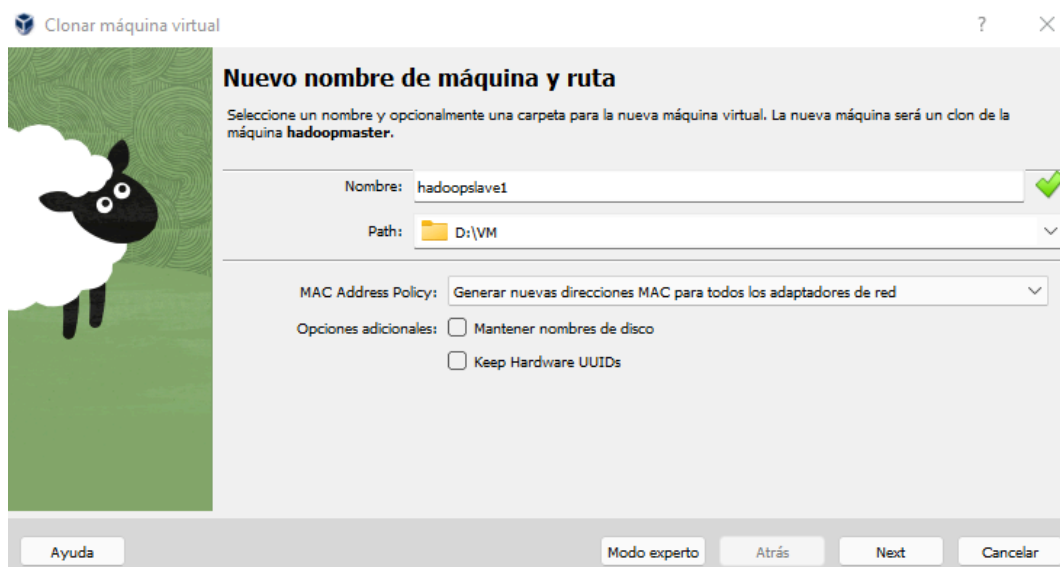
6.5. Afegir dos esclaus al clúster

Com hem dit al principi d'aquest capítol, volem configurar un clúster amb 3 nodes: un mestre i dos esclaus. Una vegada que tenim configurat el mestre, farem dos clons i modificarem els seus fitxers de configuració perquè formin part del clúster com a esclaus. Però abans de clonar la màquina virtual, anam a fer una darrera modificació en el mestre: configurarem el fitxer `/etc/hosts` per reflectir els tres nodes que volem que tenguim el clúster. Així que, amb l'usuari `root`, afegirem les següents tres línies al final del fitxer:

```
10.10.10.1 hadoopmaster
10.10.10.2 hadoopslave1
10.10.10.3 hadoopslave2
```

Com podem veure el nostre clúster estarà format per *hadoopmaster* (10.10.10.1), *hadoopslave1* (10.10.10.2) i *hadoopslave2* (10.10.10.3). Podríem emprar qualsevol altra configuració de xarxa, per exemple amb adreces 192.168.*.*

I ara ja podem fer les còpies de la màquina virtual *hadoopmaster*. Aturarem la màquina virtual i amb el botó dret seleccionam l'opció "Clonar". Al primer clon li posarem el nom *hadoopslave1* i a la política d'adreces MAC, hem de triar l'opció "Generar nuevas direcciones MAC para todos los adaptadores de red".



Imatge: Clonació (*hadoopslave1*)

A la pantalla següent marcam l'opció "Clonación completa" i fem clic a "Terminar".

I a continuació repetim el procés per a *hadoopslave2*.

Ara ja podem arrencar les màquines i, amb l'usuari `root`, anam a configurar el nom de host i el segon adaptador de xarxa, perquè les màquines puguin interactuar entre elles.

En les dues màquines esclau, hem d'editar el fitxer `/etc/hostname` per posar-hi *hadoopslave1* i *hadoopslave2*, respectivament. Recordau que també es pot fer mitjançant l'ordre

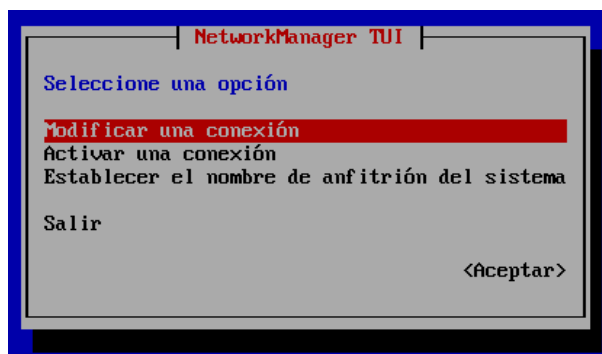
```
hostnamectl set-hostname hadoopslave1
```

Ara, en cada una de les 3 màquines instal·larem l'eina **nmtui** per configurar el segon adaptador de xarxa (el corresponent al "adaptador sólo anfitrión") de cada màquina:

```
yum install NetworkManager-tui
```

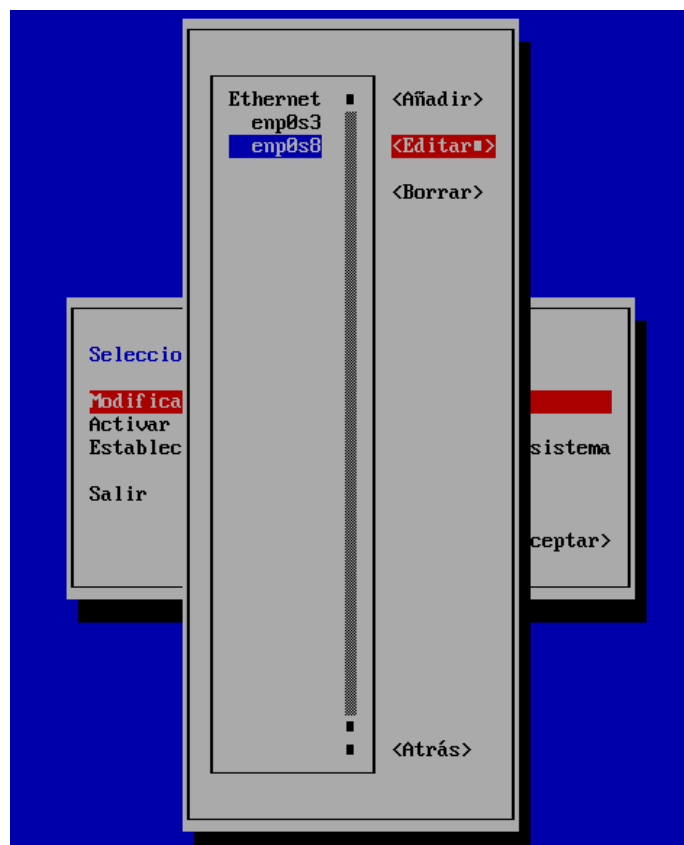
I, una vegada instal·lada, l'executem:

```
nmtui
```



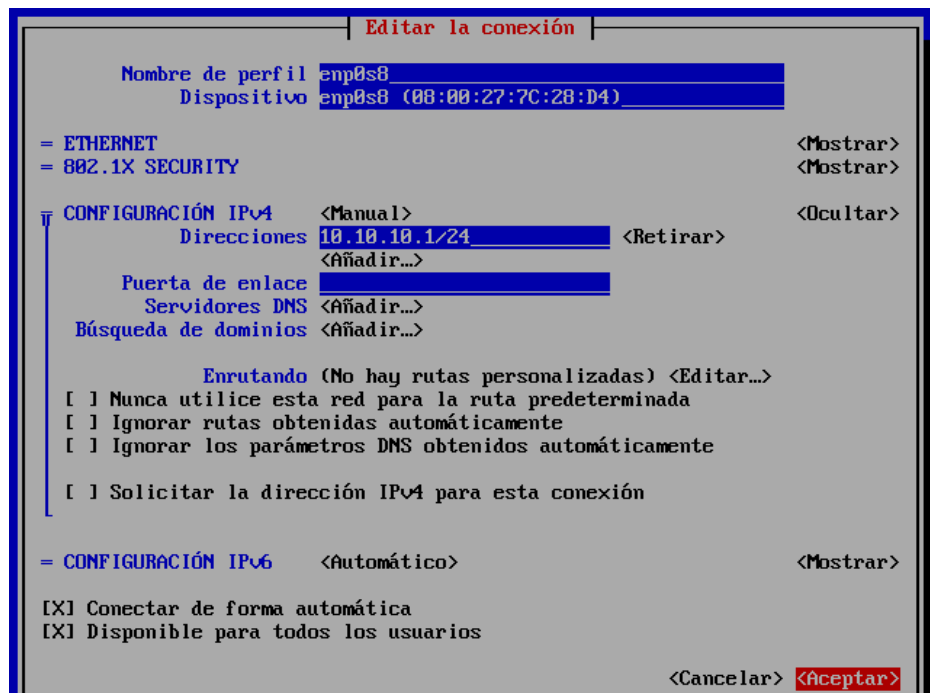
Imatge: Network Manager TUI

Seleccionam "Modificar una conexión" i hem d'editar la targeta "enp0s8" (la "enp0s3" és que es fa servir per a l'adaptador NAT):



Imatge: Editar la segona targeta de xarxa (amb nmtui)

En la pantalla que apareix a continuació hem de seleccionar "Manual" a la configuració d'IPv4, a continuació "Mostrar" també a la configuració IPv4 i afegim una adreça (10.10.10.1/24 per a *hadoopmaster*, 10.10.10.2/24 per a *hadoopslave1* i 10.10.10.3/24 per a *hadoopslave2*). Ens assegurem que quedi marcada l'opció "Conectar de forma automática" i acabam amb "Aceptar".



Imatge: Afegir adreça IP a la segona targeta (amb nmtui)

Després de guardar els canvis a les tres màquines, les haurem de reiniciar.

Per comprovar que la configuració de xarxa és correcta, farem ping entre les màquines (millor utilitzant el *hostname*) i veurem que es responen correctament.

Ara ja podem entrar amb l'usuari *hadoop* a cada una de les tres màquines. En les dues màquines esclaus, hem de canviar el directori associat al datanode, perquè no sigui igual que el del mestre. Editarem el fitxer **\$HADOOP_HOME/etc/hadoop/hdfs-site.xml** i hem de canviar la propietat *dfs.data.dir*. A *hadoopslave1* posarem */home/hadoop/hadoopdata/hdfs/datanode1* i a *hadoopslave2* */home/hadoop/hadoopdata/hdfs/datanode2*. En el cas del mestre, no cal canviar-lo.

Només ens queda dir-li al mestre quins són els nodes del clúster (*workers*). En la màquina *hadoopmaster*, seguint amb l'usuari *hadoop*, hem d'editar el fitxer **\$HADOOP_HOME/etc/hadoop/workers**. El contingut del fitxer ha de quedar així:

```
hadoopmaster
hadoopslave1
hadoopslave2
```

```
hadoopmaster
hadoopslave1
hadoopslave2
```

Imatge: Contingut del fitxer *workers*

I ara ja podem arrencar el clúster des del node mestre amb:

```
start-all.sh
```

```
[hadoop@hadoopmaster hadoop1$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [hadoopmaster]
hadoopmaster: Warning: Permanently added the ECDSA host key for IP address '10.10.10.1' to the list
of known hosts.
Starting datanodes
hadoopslave2: Warning: Permanently added 'hadoopslave2,10.10.10.3' (ECDSA) to the list of known host
s.
hadoopslave1: Warning: Permanently added 'hadoopslave1,10.10.10.2' (ECDSA) to the list of known host
s.
Starting secondary namenodes [hadoopmaster]
Starting resource manager
Starting node managers
```

Imatge: Arrencada del clúster

Si feim un `jps` als tres nodes podrem veure com tots els processos estan en marxa. En el mestre tenim *NameNode* i *DataNode*, a més de *SecondaryNameNode*, *NodeManager* i *ResourceManager*. En cada un dels esclaus, tenim *DataNode*, a més de *NodeManager*.

a) <i>hadoopmaster</i>	b) <i>hadoopslave1</i>	c) <i>hadoopslave2</i>
<pre>[hadoop@hadoopmaster hadoop1\$ jps 1809 NameNode 2370 ResourceManager 2499 NodeManager 2137 SecondaryNameNode 2861 Jps 1934 DataNode</pre>	<pre>[hadoop@hadoopslave1 hadoop1\$ jps 1920 Jps 1691 DataNode 1790 NodeManager</pre>	<pre>[hadoop@hadoopslave2 hadoop1\$ jps 1687 DataNode 1785 NodeManager 1916 Jps</pre>

Figura: Processos dels nodes del clúster

6.6. Provar el clúster

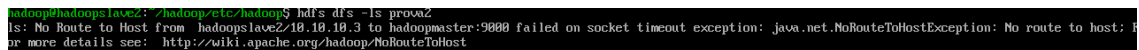
Ara ja podem crear un directori al clúster, des de *hadoopmaster* amb l'usuari *hadoop*:

```
hdfs dfs -mkdir prova2
```

En principi, aquest directori *prova2* hauria d'estar accessible des de qualsevol dels nodes del clúster. Per tant, anem a mirar el contingut del directori *prova2* des d'un dels nodes esclaus, per exemple *hadoopslave2*:

```
hdfs dfs -ls prova2
```

I aquí veurem que ens dona un error indicant que no pot connectar a *hadoopmaster:9000*:



```
hadoop@hadoopslave2: /hadoop/etc/hadoop$ hdfs dfs -ls prova2
ls: Is: No Route to Host from hadoopslave2/10.10.10.3 to hadoopmaster:9000 failed on socket timeout exception: java.net.NoRouteToHostException: No route to host: P
for more details see: http://wiki.apache.org/hadoop/NoRouteToHost
```

Imatge: Error de connexió amb el namenode

El motiu de l'error és que el firewall està bloquejant alguns ports i per tant, està tallant peticions entre el namenode i els datanodes. El que hauríem de fer és obrir tots els ports que fa servir Hadoop. Per simplificar, aquí anam directament a desactivar el firewall:

Primer comprovam que estigui funcionant:

```
sudo firewall-cmd --state
```

I hauria de retornar *running*.

Per aturar el firewall, hem d'executar:

```
sudo systemctl stop firewalld
```

Però quan reiniciem la màquina, es tornaria a posar en marxa. Així que, a més de l'ordre anterior, n'hem d'executar també aquestes dues:

```
sudo systemctl disable firewalld
```

```
sudo systemctl mask --now firewalld
```

La segona evita que el firewall pugui ser arrencat per altres serveis.

Això ho hem de fer a les tres màquines.

Podem comprovar que podem fer un telnet al port 9000 de *hadoopmaster* des d'un dels esclaus (després d'instal·lar telnet amb *sudo yum install telnet*) fent

```
telnet hadoopmaster 9000
```

Si s'obri un terminal de telnet és que la configuració és correcta (i la podeu tancar).

Anam ara a esborrar el directori *prova2* (que s'ha creat malament) i crearem un nou directori *prova3*, per comprovar que tot ja funciona correctament:

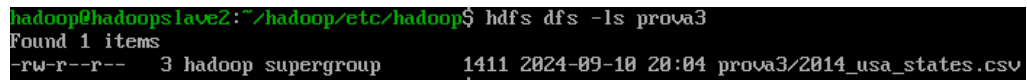
```
hdfs dfs -rm -r prova2
hdfs dfs -mkdir prova3
```

Anam a copiar-hi ara el fitxer d'estats d'Estats Units.

```
hdfs dfs -put 2014_usa_states.csv prova3
```

Per acabar, des d'un dels nodes esclaus, anem a veure el contingut del directori *prova3*:

```
hdfs dfs -ls prova3
```



```
hadoop@hadoopslave2:~/hadoop/etc/hadoop$ hdfs dfs -ls prova3
Found 1 items
-rw-r--r-- 3 hadoop supergroup      1411 2024-09-10 20:04 prova3/2014_usa_states.csv
```

Imatge: Contingut del directori *prova3* des de *hadoopslave2*

6.7. Interactuar amb la interfície gràfica

Tant HDFS, com també YARN, ens ofereixen una interfície gràfica web, que es pot utilitzar a través d'un navegador. Anam a veure com funciona, però per fer-ho, necessitarem disposar d'un entorn gràfic on executar un navegador. Tenim dues opcions:

Opció 1

Instal·lam un entorn gràfic GNOME en una de les màquines del node, preferiblement el mestre (pot ser convenient assignar-li més RAM a la màquina virtual):

```
yum groups install "GNOME Desktop"
```

I, una vegada instal·lat, obrim l'entorn gràfic amb:

```
startx
```

Opció 2

Utilitzam una altra màquina virtual amb un entorn gràfic, per exemple amb Ubuntu. Haurem d'assignar-li una segona targeta de xarxa i configurar-la com hem fet amb els nodes del clúster. A continuació, hem d'incloure les 3 màquines del clúster al fitxer `/etc/hosts` (on també hem assignat la IP 10.10.10.4 a la màquina amb Ubuntu, anomenada `ubuntu1`):

```
10.10.10.1 hadoopmaster
10.10.10.2 hadoopslave1
10.10.10.3 hadoopslave2
10.10.10.4 ubuntu1
```

Imatge: Fragment del fitxer `/etc/hosts`

Ara ja podem obrir un navegador i connectar-nos a:

- Interfície per al namenode HDFS: <http://hadoopmaster:9870>
- Interfície per als datanodes: <http://hadoopmaster:9864>, <http://hadoopslave1:9864> i <http://hadoopslave2:9864>
- Interfície YARN: <http://hadoopmaster:8088>

Començarem veient la interfície d'HDFS del NameNode: <http://hadoopmaster:9870>

← → ↺

hadoopmaster:9870/dfshealth.html#tab-overview

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities ▾

Overview 'hadoopmaster:9000' (✓active)

Started:	Tue Sep 10 20:24:09 +0200 2024
Version:	3.4.0, rbd8b77f398f626bb7791783192ee7a5dfaee760
Compiled:	Mon Mar 04 07:35:00 +0100 2024 by root from (HEAD detached at release-3.4.0-RC3)
Cluster ID:	CID-15373077-e16a-46bf-8e55-d0895680302b
Block Pool ID:	BP-1066275894-10.10.10.1-1725992599916

Summary

Security is off.

Safemode is off.

5 files and directories, 1 blocks (1 replicated blocks, 0 erasure coded block groups) = 6 total filesystem object(s).

Heap Memory used 34.85 MB of 43.04 MB Heap Memory. Max Heap Memory is 475.63 MB.

Non Heap Memory used 60.7 MB of 62.14 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Imatge: Pàgina d'inici de la interfície web d'HDFS

Podem veure que tenim tres nodes actius en el clúster. El sistema té una capacitat de 110,92 Gb, de la qual ara mateix només estam utilitzant 56 Kb.

En la pàgina de Datanodes podem veure la informació dels tres datanodes que tenim actius (el del node mestre i els dos dels esclaus).

Datanode Information

✓ In service

⬇ Down

🔄 Decommissioning

🛑 Decommissioned

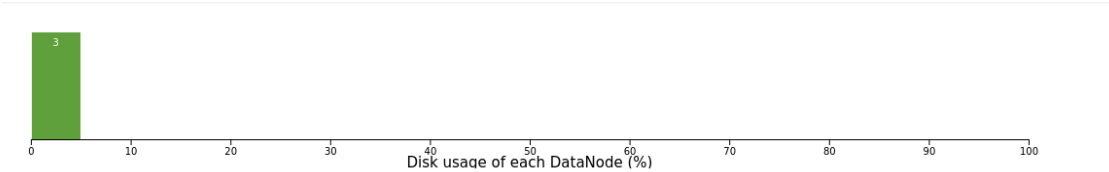
⚰ Decommissioned & dead

🔧 Entering Maintenance

🔥 In Maintenance

🔪 In Maintenance & dead

Datanode usage histogram



In operation

DataNode State: All ▾

Show: 25 ▾ entries

Search:

Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Block pool usage StdDev	Version
✓/default-rack/hadoopmaster:9866 (10.10.10.1:9866)	http://hadoopmaster:9864	1s	5m	5.4 KB	5.06 GB	13.93 GB	1	5.4 KB (0%)	0%	3.4.0
✓/default-rack/hadoopslave1:9866 (10.10.10.2:9866)	http://hadoopslave1:9864	0s	5m	5.4 KB	5.07 GB	13.93 GB	1	5.4 KB (0%)	0%	3.4.0
✓/default-rack/hadoopslave2:9866 (10.10.10.3:9866)	http://hadoopslave2:9864	2s	5m	5.4 KB	5.05 GB	13.93 GB	1	5.4 KB (0%)	0%	3.4.0

Showing 1 to 3 of 3 entries

Previous

1

Next

Imatge: Datanodes del clúster

Si clicam en l'adreça HTTP de cada un dels datanodes anirem a la seva interfície web. Per exemple, la següent imatge mostra la del datanode de hadoopslave1:

DataNode on hadoopslave1:9866

Cluster ID:	CID-15373077-e16a-46bf-8e55-d0895680302b
Started:	Tue Sep 10 20:24:12 +0200 2024
Version:	3.4.0, rbd8b77f398f626bb7791783192ee7a5dfaec760

Block Pools

Namenode Address	Namenode HA State	Block Pool ID	Actor State	Last Heartbeat Sent	Last Heartbeat Response	Last Block Report	Last Block Report Size (Max Size)
hadoopmaster:9000	active	BP-1066275894-10.10.10.1-1725992599916	RUNNING	1s	1s	8 minutes	0 B (128 MB)

Volume Information

Directory	StorageType	Capacity Used	Capacity Left	Capacity Reserved	Reserved Space for Replicas	Blocks
/home/hadoop/hadoopdata/hdfs/datanode1	DISK	5.4 KB	8.86 GB	0 B	0 B	1

Imatge: Interfície web del datanode de hadoopslave1

La interfície web dels datanodes té una pestanya amb diverses utilitats. La més emprada és probablement la que ens permet veure tots els logs de Hadoop:

Name ↑	Last Modified	Size
hadoop-hadoop-datanode-hadoopmaster.log	Sep 9, 2024 1:56:52 PM	276,621 bytes
hadoop-hadoop-datanode-hadoopmaster.out	Sep 9, 2024 1:36:02 PM	822 bytes
hadoop-hadoop-datanode-hadoopmaster.out.1	Sep 9, 2024 1:27:21 PM	822 bytes
hadoop-hadoop-datanode-hadoopmaster.out.2	Sep 9, 2024 1:17:13 PM	822 bytes
hadoop-hadoop-datanode-hadoopslave1.log	Sep 10, 2024 8:32:38 PM	567,907 bytes
hadoop-hadoop-datanode-hadoopslave1.out	Sep 10, 2024 8:24:10 PM	822 bytes
hadoop-hadoop-datanode-hadoopslave1.out.1	Sep 10, 2024 8:17:40 PM	822 bytes
hadoop-hadoop-datanode-hadoopslave1.out.2	Sep 10, 2024 7:35:28 PM	822 bytes
hadoop-hadoop-datanode-hadoopslave1.out.3	Sep 10, 2024 6:31:12 PM	822 bytes
hadoop-hadoop-namenode-hadoopmaster.log	Sep 9, 2024 1:56:52 PM	132,557 bytes
hadoop-hadoop-namenode-hadoopmaster.out	Sep 9, 2024 1:35:59 PM	822 bytes
hadoop-hadoop-namenode-hadoopmaster.out.1	Sep 9, 2024 1:27:18 PM	822 bytes
hadoop-hadoop-namenode-hadoopmaster.out.2	Sep 9, 2024 1:17:10 PM	822 bytes
hadoop-hadoop-nodemanager-hadoopmaster.log	Sep 9, 2024 2:06:24 PM	138,323 bytes
hadoop-hadoop-nodemanager-hadoopmaster.out	Sep 9, 2024 1:36:28 PM	2,333 bytes
hadoop-hadoop-nodemanager-hadoopmaster.out.1	Sep 9, 2024 1:27:44 PM	2,333 bytes
hadoop-hadoop-nodemanager-hadoopmaster.out.2	Sep 9, 2024 1:17:38 PM	2,333 bytes
hadoop-hadoop-nodemanager-hadoopslave1.log	Sep 10, 2024 8:24:48 PM	292,057 bytes
hadoop-hadoop-nodemanager-hadoopslave1.out	Sep 10, 2024 8:24:40 PM	2,333 bytes
hadoop-hadoop-nodemanager-hadoopslave1.out.1	Sep 10, 2024 8:18:09 PM	2,333 bytes
hadoop-hadoop-nodemanager-hadoopslave1.out.2	Sep 10, 2024 7:35:59 PM	2,333 bytes
hadoop-hadoop-nodemanager-hadoopslave1.out.3	Sep 10, 2024 6:31:41 PM	2,333 bytes
hadoop-hadoop-resourcemanager-hadoopmaster.log	Sep 9, 2024 1:46:29 PM	163,996 bytes
hadoop-hadoop-resourcemanager-hadoopmaster.out	Sep 9, 2024 1:36:27 PM	2,349 bytes
hadoop-hadoop-resourcemanager-hadoopmaster.out.1	Sep 9, 2024 1:27:43 PM	2,349 bytes
hadoop-hadoop-resourcemanager-hadoopmaster.out.2	Sep 9, 2024 1:17:38 PM	2,349 bytes
hadoop-hadoop-secondarynamenode-hadoopmaster.log	Sep 9, 2024 1:37:16 PM	199,410 bytes
hadoop-hadoop-secondarynamenode-hadoopmaster.out	Sep 9, 2024 1:36:08 PM	822 bytes
hadoop-hadoop-secondarynamenode-hadoopmaster.out.1	Sep 9, 2024 1:34:29 PM	22,836 bytes
hadoop-hadoop-secondarynamenode-hadoopmaster.out.2	Sep 9, 2024 1:26:43 PM	30,174 bytes
SecurityAuth-hadoop.audit	Sep 9, 2024 1:17:08 PM	0 bytes
userlogs/	Sep 9, 2024 1:17:32 PM	6 bytes

Imatge: Logs de hadoopslave1

Si tornam a la interfície del namenode (hadoopmaster:9870), també tenim una sèrie d'utilitats, entre elles la dels logs. Però aquí la més important és la que ens permet interactuar d'una manera gràfica amb el sistema d'arxius. En la següent imatge podem veure el contingut del directori *prova3* que hem creat en l'apartat anterior:

Browse Directory

/user/hadoop/prova3

Go!

Show

25

entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	1.38 KB	Sep 10 20:28	3	128 MB	2014_usa_states.csv	

Showing 1 to 1 of 1 entries

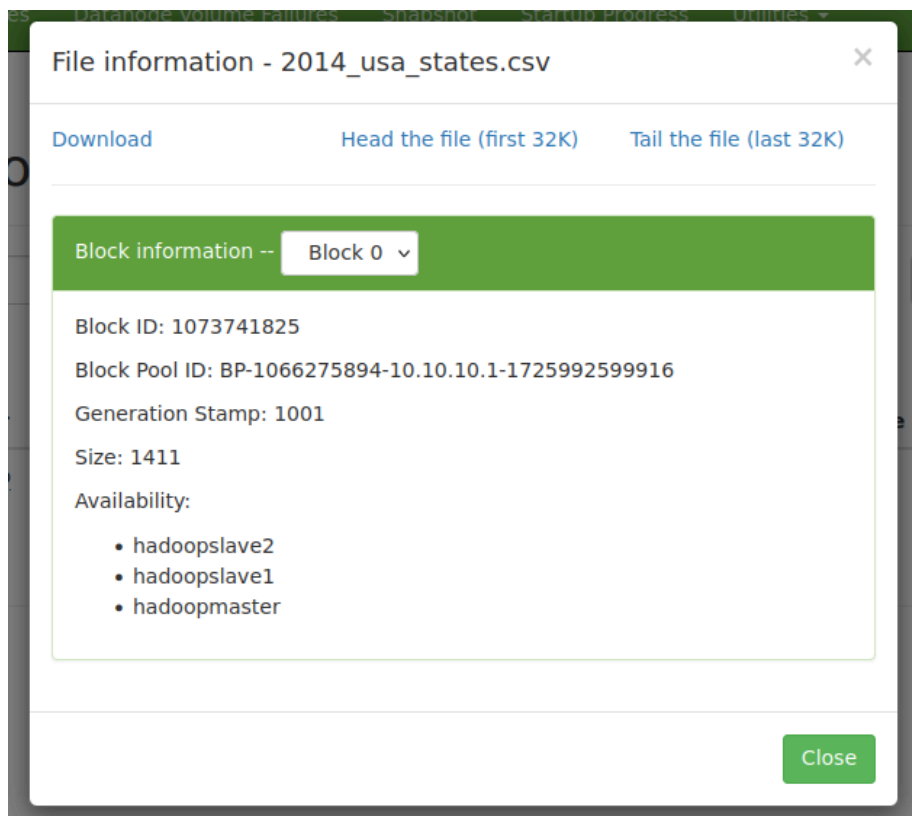
Previous

1

Next

Imatge: Contingut del directori *prova3*

Si feim clic sobre el nom del fitxer *2014_usa_states.csv* podrem veure la informació dels seus blocs i com estan repartits en el clúster. En aquest cas, com que el fitxer és menor de 128 Mb, només té un bloc (block 0), i podem observar que està disponible en els tres nodes del clúster.

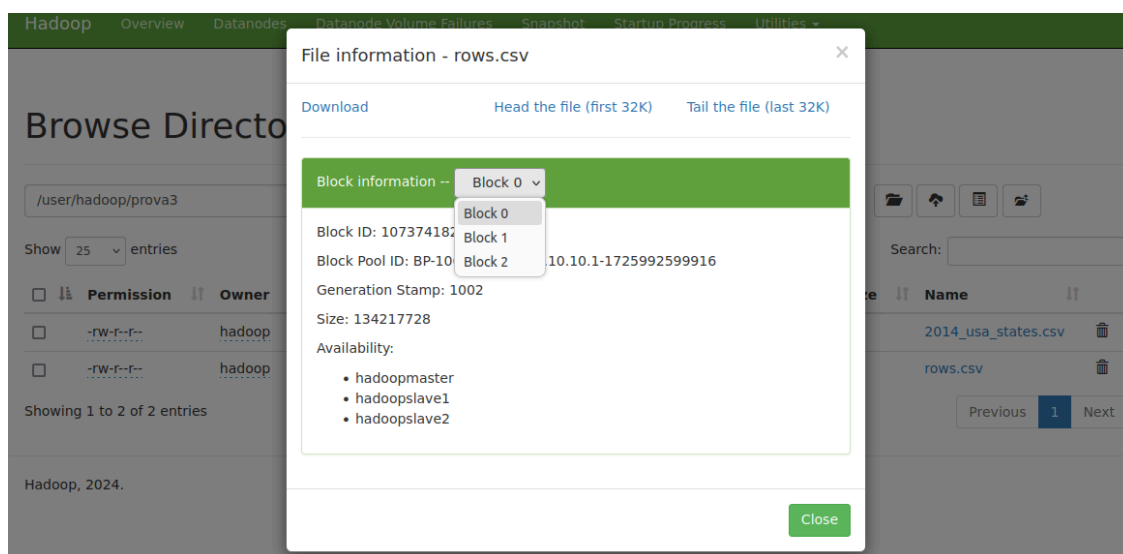


Imatge: Blocs del fitxer 2014_usa_states.csv

Ara veurem què passaria amb un arxiu més gros. Tornem a *hadoopmaster*, descarregarem el dataset dels indicadors de malalties cròniques d'Estats Units des del portal de dades obertes del Govern americà: <https://data.cdc.gov/api/views/g4ie-h725/rows.csv> i a continuació el pujarem al directori *prova2* d'HDFS:

```
wget https://data.cdc.gov/api/views/g4ie-h725/rows.csv
hdfs dfs -put rows.csv prova3
```

Si ara tornem a la interfície web, podem veure el fitxer *rows.csv* al directori *prova2* i quan hi feim clic, podem veure que el fitxer està dividit en 3 blocs (els dos primers de 128 Mb, el tercer d'uns 60 Mb), cada un dels quals està disponible als tres nodes del clúster (perquè el factor de replicació és 3).



Imatge: Distribució dels blocs d'un fitxer

En la tasca del lliurament, on haureu d'afegir un node més al clúster, veureu que no tots els blocs estan a tots els nodes, ja que tindrem 4 nodes i un factor de replicació de 3.

Per acabar, veurem també la interfície web de YARN: <http://hadoopmaster:8088>. No entrarem en detalls, perquè YARN no és objecte d'aquest lliurament. Ens quedarem, però, en què podem veure que tenim 3 nodes actius i, en la pantalla de "Nodes", podem trobar-ne més detalls:

Cluster

Nodes

Node Labels

Applications

NEW

SHOWING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

KILLED

Scheduler

Tools

Nodes of the cluster

Cluster Metrics

Apps Submitted	0	Apps Pending	0	Apps Running	0	Apps Completed	0	Containers Running	0	User Resources	<memory:0 B, vCores:0>	Total Resources	<memory:24 GB, vCores:24>	Reserved Resources	<memory:0 B, vCores:0>	Physical Mem Used %	41	Physical VCores Used %	0
----------------	---	--------------	---	--------------	---	----------------	---	--------------------	---	----------------	------------------------	-----------------	---------------------------	--------------------	------------------------	---------------------	----	------------------------	---

Cluster Nodes Metrics

Active Nodes	3	Decommissioning Nodes	0	Decommissioned Nodes	0	Lost Nodes	0	Unhealthy Nodes	0	Rebooted Nodes	0	Shutdown Nodes	0
--------------	---	-----------------------	---	----------------------	---	------------	---	-----------------	---	----------------	---	----------------	---

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority	Scheduler Busy %	RM Dispatcher EventQueue Size	Scheduler Dispatcher EventQueue Size
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0	0	0	0

Show: 20 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	Phys Mem Used %	VCores Used	VCores Avail	Phys VCores Used %	Version
/ default-rack		RUNNING	hadoopmaster:46741	hadoopmaster:8042	Tue Sep 10 20:42:40 +0200 2024		0		0 B	8 GB	57	0	8	0	3.4.0
/ default-rack		RUNNING	hadoopslave1:38965	hadoopslave1:8042	Tue Sep 10 20:42:36 +0200 2024		0		0 B	8 GB	33	0	8	0	3.4.0
/ default-rack		RUNNING	hadoopslave2:42989	hadoopslave2:8042	Tue Sep 10 20:42:39 +0200 2024		0		0 B	8 GB	32	0	8	0	3.4.0

Showing 1 to 3 of 3 entries

First Previous 1 Next Last

Imatge: Nodes a la interfície web de YARN