Cloud computing i dades massives (*big data*)

PID_00286205

Remo Suppi Boldrito

Temps mínim de dedicació recomanat: 6 hores

Universitat Oberta de Catalunya

Remo Suppi Boldrito

Enginyer de Telecomunicacions. Doctor en Informàtica per la UAB. Professor del Departament d'Arquitectura de Computadors i Sistemes Operatius a la Universitat Autònoma de Barcelona.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats pel professor: Josep Jorba Esteve

Primera edició: febrer 2022

Av. Tibidabo, 39-43, 08035 Barcelona
Autoria: Remo Suppi Boldrito
Producció: FUOC

Índex

1. Introducció

Les dades massives (*big data*) és un terme que ha adquirit especial rellevància durant l'última dècada i el seu creixement i la seva utilització no paren de créixer.

Aquest concepte (també referit com a *dades a gran escala*) s'utilitza per a un conjunt de dades que, donada la seva quantitat/mida, no és possible analitzar/processar amb els mètodes tradicionals, i cal optar per eines i algoritmes alternatius que puguin extreure informació en temps acceptables.

Moltes són les notícies on la utilització de les dades massives és una realitat.

Per exemple: la utilització de les bases de dades de cerca com el corrector d'ortografia de Google que suggereix (fins i tot cerca amb anticipació abans que acabem d'escriure) la paraula adequada perquè altres milers de persones l'han buscat abans, les dades utilitzades per la lliga NFL per proveir de dades per a la presa de decisions, o l'NBA, o com el *big data* va ajudar a guanyar a Obama les eleccions als Estats Units, o aplicacions com PriceStats, que de la informació de preus de botigues en línia «obté» informació diària de l'evolució de la inflació en vint-i-dos països del món, o l'aplicació Illustreets, que sobre la base de l'*open data* disponible al Regne Unit permet generar diferents mapes, com The Integration Hub, per estimular les iniciatives i discussions sobre la integració de diferents col·lectius a la ciutat de Londres amb la finalitat d'evitar la segregació racial.

No obstant això, cal tenir en compte que les dades són dades i, a vegades, la seva interpretació pot donar lloc a dubtes o errors.

Per exemple, va ser el cas de la predicció d'epidèmies de grip que feia Google (Google Flu Trends). El 2008, els investigadors de la companyia van explorar la potencialitat de les dades massives de les cerques en temes vinculats amb la grip partint del fet que la persona malalta busca informació relacionada, la qual cosa els permetia obtenir prediccions instantànies de les epidèmies a escala mundial (*Nature*, 2008). Però el 2013 les prediccions van fracassar estrepitosament (amb un 140 % d'error en el pic) i alguns científics, a *Science*, van indicar que el problema no era en el valor de les dades massives, sinó en les problemàtiques derivades del seu ús i la seva interpretació. En aquest cas en particular van ser un cúmul de situacions, més enllà de les bones intencions dels investigadors de Google (encara que alguns científics li critiquen l'opacitat en el mètode i les dades), ja que l'algoritme era vulnerable a cerques no correlacionades, a no contemplar els canvis del comportament de les cerques en el temps, a la introducció de nous complements en l'algoritme de cerques sobre salut que no es van tenir en compte per a les prediccions i que els afectaven, i una llarga llista més. Amb això es va demostrar un aspecte crític de les dades massives on el problema no és en les dades sinó en la manera en què són tractades, filtrades, processades, etc. (vegeu l'article de *Wired*).

Les dades massives han generat la creació/adequació de tècniques i metodologies, i la creació de noves eines amb un creixement constant, ja que les necessitats derivades de la captura, emmagatzematge, cerca, compartició, anàlisi i visualització són totalment diferents a les conegudes fins ara i requereixen nous mètodes, procediments i estratègies per dur-les a terme.

Molts camps de la ciència, però també orientats al negoci, estan obtenint beneficis d'aquestes dades i les seves necessitats/requeriments són cada vegada més específics/diversos.

Entre els diferents segments es poden esmentar la ciència (genètica, biologia, medicina, enginyeria, física/astrofísica...), les anàlisis de negoci (banca, publicitat, borsa, compres, moda, frau, tendències de negoci...), la salut (epidèmies, incidències i contagis de malalties infeccioses...), o altres tan diversos com l'espionatge i el seguiment a la població o la lluita contra el crim organitzat.

Són molt interessants les reflexions d'Eric Schmidt (que ha sigut director executiu de Google i president d'Alphabet Inc.), que l'any 2011 va explicar que hi havia, en aquell temps, 295 exabytes d'informació en el món sobre la base de l'estudi de Hilbert i López *The world's technological capacity to store, communicate and compute information* (*Science*). L'estudi del període 1986-2007 establia com a punt d'inici de l'era digital l'any 2002 com aquell en el qual la informació digital emmagatzemada va superar la capacitat analògica d'emmagatzematge, i considerava que l'any 2007 el 94 % de tota la informació era digital.

El 2018, l'estudi *How Much Data Is Generated Every Minute?*, basat en el report de Dom, indica que «Over 2.5 quintillion bytes of data are created every single day, and it's only going to grow from there. By 2020, it's estimated that 1.7MB of data will be created every second for every person on earth», i assegura que el «90% of the world's data has been created in the last two years». Avui dia, tenim millors aproximacions a les dades generades i utilitzades, per exemple, l'informe *How Much Data Is Produced Every Day in 2021*, on recopilen informació de moltes fonts, però probablement la més sorprenent és que «**Cloud data storage around the world will amount to 200 Zettabytes by 2025**», la qual cosa serà molt superior als 4,4 ZB del 2019 i als 44 ZB del 2020.

Però les dades només són dades, i el que es necessita és informació, per la qual cosa s'ha de filtrar, processar, creuar i analitzar per extreure la informació subjacent; per a això es necessita el ***cloud computing***, que és l'únic que pot proveir dels recursos necessaris per fer aquestes tasques i sense grans inversions. ***Big data*** i ***cloud computing*** són dos termes d'aquesta nova dècada que seran fonamentals per a la generació de nous negocis i noves oportunitats empresarials.

És interessant la projecció que fa l'informe del McKinsey Global Institute en el qual s'analitzen les possibilitats de negoci de les dades massives i la seva utilització, i com aquestes generen noves oportunitats empresarials que es poden quantificar en milers de milions de dòlars [Bdn].

Els experts de *Gartner* el 2011 van definir com a **característiques** de les dades massives les anomenades **3V** (volum, varietat, velocitat) i IBM, un altre dels grans actors en aquest tema, va estendre aquestes característiques a «veracitat»

i va generar així les 4V (infografia); amb tot i això, han estat ampliades a mesura que s'han anat consolidant les tecnologies [Bdd] i actualment les podem enumerar d'aquesta manera (ordre alfabètic):

1) Complexitat: aspecte que prevaldrà sobretot quan es disposi de grans volums de dades assemblades de diverses fonts.

2) Variabilitat: no totes les dades seran homogènies i hi haurà inconsistències que el sistema d'emmagatzematge i processament haurà de contemplar per a gestionar-les. Les eines utilitzades hauran de ser flexibles per adaptar-se a aquestes situacions i no generar errors quan el tipus/format/estructura de les dades no sigui l'esperat.

3) Varietat: formes i tipus de fonts de dades que tindran (o no) diferents estructures en comparació de les ja conegudes fins ara i on un sistema de processament haurà de ser capaç d'emmagatzemar i processar sense haver de realitzar un preprocés per estructurar o indexar la informació.

4) Velocitat: a la qual es produeixen, i a les quals serà necessari capturar, emmagatzemar, analitzar i extreure valor per aprofitar les finestres d'oportunitats basades en aquestes dades. Això constitueix un repte per a la tecnologia i el *cloud computing* és el principal actor que pot satisfer aquestes necessitats en aquest escenari.

5) Veracitat: qualitat de les dades capturades (predictibilitat i disponibilitat).

6) Volum: quantitat de dades utilitzada (que indicarà si es tracta de dades massives o no).

La potencialitat de nova informació, negocis i oportunitats generada per les dades massives ha evolucionat en paral·lel a la tecnologia necessària per tractar-les. Noms com Hadoop, MapReduce, NoSQL, Cassandra, *business intelligence* o *machine learning* han adquirit una popularitat derivada de les eines/mètodes utilitzats per tractar el *big data*, els quals poden treballar sobre tres tipus de dades massives d'acord amb la seva **estructura**:

1) Dades estructurades (*structured data*): dades com les utilitzades per la tecnologia actual, definides la seva longitud i el seu format i que es poden (generalment) emmagatzemar en taules (bases de dades relacionals/fulls de càlcul).

2) Dades no estructurades (*unstructured data*): mantenen una estructura similar a com han estat recollides, sense format específic; generalment són una mescla de tipus/formats de dades (dades en PDF, documents multimèdia, correus electrònics o documents de text).

3) Dades semiestructurades (*semistructured data*): dades que es troben entre les dues definicions anteriors; no tenen camps determinats, però inclouen marques per separar els diferents elements (per exemple, fitxers HTML, XML, JSON...).

Tota aquesta «revolució» ha permès reforçar i evolucionar tècniques i metodologies existents, però una part important d'aquest moviment és la nova «manera de pensar i de fer» que s'ha hagut de crear per tractar amb aquestes dades i de la qual han sorgit noves metodologies, tècniques i eines no conegudes fins ara, així com altres ja conegudes i en desús s'han reformulat i han cobrat un nou impuls per adequar-se a les noves dades/temps.

Entre les tècniques de cerca i anàlisi de dades que s'han hagut d'adaptar i reformular per tractar amb el *big data*, es poden esmentar:

- **associació**: trobar relacions entre diferents variables causals,
- **mineria de dades**: *data mining*, trobar comportaments predictius combinant mètodes estadístics i d'aprenentatge automàtic,
- **agrupació**: *clustering*, divisió de grans grups de dades en grups més petits per trobar similituds,
- **anàlisi de text**: *text analytics*, atès que gran part del conjunt de dades és en mode text s'han hagut de definir nous mètodes per extreure informació de grans volums de dades en text.

Per a això, en molts casos el tractament està orientat cap a tipus de dades no estructurades, per la qual cosa s'han hagut de buscar noves maneres d'emmagatzemar-les/cercar-les. En aquest sentit, les tècniques NoSQL (*not only SQL*) es refereixen a dades que no compleixen esquemes tradicionals d'entitat-relació de les bases de dades relacionals (tradicionals); permeten a més formes més flexibles i concurrents de manipular grans quantitats d'informació i de forma molt més ràpida. Aquest tractament no estàndard és utilitzat per emmagatzemar/recuperar/cercar la informació en diferents formes com, per exemple, per mitjà d'una clau-valor, d'un graf, o mètodes orientats a columnes entre les més importants, i això, al seu torn, s'ha traduït en noves eines que s'han hagut de desenvolupar (o evolucionar les existents), per exemple, en gestors de bases de dades (Cassandra, MongoDB, MariaDB, Hive, HBase, GraphDB...) i en nous mètodes per processar-les (Hadoop, Apache Stark...).

És interessant analitzar la línia de temps de les dades massives proposada per Verdict o en aquesta infografia, en la qual es poden trobar els esdeveniments i les fites més importants, així com la predicció de l'horitzó 2025/2022, respectivament. També és interessant l'opinió d'un expert com Kenneth Cukier (*The Economist*, Londres), en l'informe del qual «Los *big data* y el futuro de los negocios», publicat per OpenminfBBVA, exposa «que ningún ámbito de la ac-

tividad humana ni sector de la industria será inmune a la total reorganización que están a punto de traer los *big data* a medida que transforman la sociedad, la política y los negocios». I afirmant que «Más no es solo más. Más es nuevo. Más es mejor. Más es distinto».

Com es pot apreciar, en l'opinió d'experts, l'experiència del *big data* aplicada als negocis tindrà com a premissa «conèixer el client» com a evolució dels «antics» CRM (*customer relationship management*), ja que això permetrà obtenir no només la informació (interna) d'aquest, sinó creuar aquesta informació amb l'externa (xarxes socials, geolocalització, opinió, sentiments...), la qual cosa, al seu torn, generarà valor afegit, perquè s'estarà més a prop del que necessita i pensa el client per poder-li oferir productes/serveis amb un alt grau d'especificació i detall, encara que ajustat per al que necessita o considera necessari.

1.1. Arquitectura i paradigmes del *big data*

Una arquitectura per al processament de les dades massives, com en qualsevol entorn orientat a les dades, està formada per quatre nivells que interactuen (recollida de dades, emmagatzematge, processament i visualització), més un de global d'administració.

Aquesta arquitectura és l'habitual que es pot trobar en qualsevol entorn orientat a les dades (per exemple, *datamining*, *business intelligence* o *deep learning*), però, quan s'associa a aquests entorns el concepte de *big data*, cadascun d'aquests nivells ha evolucionat o s'han generat noves eines/algoritmes/entorns per adequar-se a les dades massives [Bad][Bdb].

En la **recollida** de les dades orientades al *big data* han sorgit una gran quantitat d'eines orientades a aquest efecte (*data ingestion*), que permeten el processament seqüencial habitual de dades emmagatzemades (*batch* o lots) per obtenir el tram o la seqüència següent (*chunk*) de dades des de l'última llegida; en tot cas, majoritàriament, i donat el volum d'aquestes, les eines han evolucionat per recollir eficientment fluxos de dades (*streams*) en temps real.

En l'**emmagatzematge** també han sorgit grans canvis per adaptar-se a les característiques de les dades; això s'ha reflectit en una gran quantitat de desenvolupaments de motors de bases de dades (NoSQL, documents, columnes, clau/valor, grafs) i sistemes de fitxers distribuïts per adequar-se a la realitat de processament i, així, disposar d'escalabilitat, fiabilitat i proximitat de les dades per al seu tractament (per exemple, Apache HDFS, BeeGFS, Disco DDFS, Google GFS, BaiduFS, GlusterFS, QuantcastFS, CephFS, GridGainin-memoryFS, LustreFS).

Per al **processament** han sorgit nous paradigmes que han caracteritzat tot l'entorn (i en cadascuna de les capes per adequar-les a les seves necessitats); es coneixen com a MapReduce (MR) o Massive Paralell Processing (MPP).

MapReduce és un paradigma de programació el nom del qual ve donat per les dues funcions que el componen (*map* i *reduce*); per mitjà d'un entorn *open-source*, anomenat Apache Hadoop, i és avui dia un dels paradigmes més utilitzats per al processament de les dades massives. Aquest paradigma, si bé no és la solució per a tots els problemes, treballa amb grans conjunts de dades (petabytes) i s'executa sobre sistemes de fitxers distribuïts (HDFS) i vinculats a eines com HBase, Hive, Impala o Cassandra (bases de dades). Aquest paradigma és apte per processar aquelles dades que poden treballar sobre tuples del tipus (clau, valor) i on la funció *map()* processa en paral·lel les tuples d'un domini i genera una llista de parells en un domini diferent, les quals s'agruparan sota la mateixa clau utilitzant un esquema de processament *master-worker* en forma d'arbre. Posteriorment, la funció *reduce()* s'aplica en forma paral·lela per a cada grup i genera una col·lecció de valors per a cada domini. Un exemple típic és el procediment per comptar el nombre de vegades que apareixen les paraules en un text. Les funcions serien:

```
map(string name, string document):
  foreach word w in document:
    generate-tupla(w, 1);

reduce(string word, iterator partialList):
  int total = 0;
  foreach value in partialList:
    total += int(value);
  generate(word,total);
```

On la funció *map()* divideix el document per paraules i genera les tuples (paraula, valor), per exemple, la famosa frase d'A. Turing «La ciència és una equació diferencial. La religió és una condició de frontera» quedarà (La,1), (ciència,1), (és,1), (una,1), (equació,1), (diferencial,1), (La,1), (religió,1), (és,1), (una,1), (condició,1), (de,1), (frontera,1). L'entorn agruparà totes les claus iguals ('La:1,1', 'és,1,1', 'una,1,1' i la resta 'clau,1') com a entrada a *reduce()*, que generarà l'agrupació i la suma dels valors de les claus en les quals, per a l'exemple, serà (La,2), (és,2), (una,2) i la resta (clau,1).

El paradigma MPP ofereix una solució tradicional adaptada al *big data* basada a dividir els grans conjunts de dades en seccions (*slices*) que seran més fàcils de gestionar; cadascun d'aquests seran assignats a un element de còmput per al seu processament. El dispositiu de còmput els processarà i, quan acabi el sistema, combinarà els resultats parcials per donar un resultat final (equivalent a una seqüència *fork-join* en un model *master-workers*). Atès que els elements de còmput (processadors) treballaran sobre el seu segment de dades assignat

de la BD i es comunicaran a través de missatges quan generin els resultats, no hi ha interacció entre ells; això es coneix com «feblement acoblats» (altres autors ho anomenen *shared nothing*).

Entre les característiques principals d'aquests sistemes es poden enumerar les seves possibilitats de sintonització i escalat il·limitat (en principi al nombre de nodes disponible), amb el consegüent increment del seu rendiment (pràcticament en forma lineal) i sense colls d'ampolla. Hi ha una gran quantitat de bibliografia sobre la comparació MR i MPP (per exemple, *Hadoop vs. MPP*, *Introduction to MPP*); no obstant això, moltes vegades la solució no prové d'un o altre paradigma, i atès que es complementen bé quant a les prestacions i els tipus de dades que obtenen millors rendiments, és habitual que s'utilitzin arquitectures mixtes on generalment s'aplica primer l'MR sobre dades no estructurades i, posteriorment, l'MPP per processar i visualitzar les dades estructurades generades pel primer.

El projecte Apache™ Hadoop® és una plataforma *open-source* per al còmput distribuït, de confiança i escalable, i utilitzada per una gran quantitat d'empreses/institucions de renom que implementa MapReduce. Aquesta plataforma s'inspira en la feina de GoogleFS/MapReduce i el seu desenvolupament es va iniciar dins del projecte Apache Nutch i després amb el projecte Hadoop (2006); Yahoo! és un dels seus grans contribuïdors (i D. Cutting un dels creadors d'aquesta companyia, que el va batejar així per l'elefant de joguina del seu fill). El codi inicial va ser de 5k línies per a HDFS i 6k línies per a MapReduce, i la primera versió Hadoop 0.1.0 va ser publicada l'abril del 2006; l'última disponible (estable) és del juny del 2021, la versió 3.3.1 [Ahd].

Apache Hadoop és un entorn per al processament distribuït de grans conjunts de dades a través de clústers de còmput; utilitza models de programació senzills i té la possibilitat d'escalar des de servidors individuals a milers de processadors. Bàsicament, Hadoop (a partir de la versió 2.0) està format per quatre mòduls:

1) Hadoop YARN: marc per a la programació de tasques i gestió de recursos del clúster.

2) Hadoop MapReduce: sistema basat en YARN per al processament paral·lel de grans conjunts de dades.

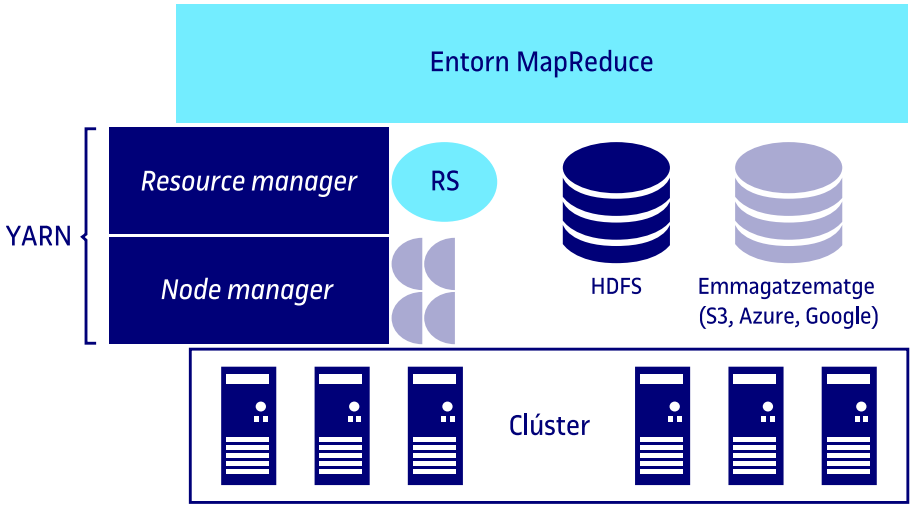
3) Hadoop Distributed File System (HDFS): sistema de fitxers distribuït que proporciona accés d'alt rendiment a les dades de l'aplicació.

4) Hadoop Common: les utilitats comunes que suporten els altres mòduls de Hadoop.

No obstant això, Hadoop es pot relacionar/integrar/ampliar de manera fàcil i ràpida amb els projectes següents (només alguns dels més habituals/referenciats):

- **Ambari**: eina web per a l'aprovisionament, la gestió i el monitoratge de clústers Hadoop amb HDFS, MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig i Sqoop.
- **Avro**: serialització de dades.
- **Cassandra**: base de dades escalable sense punts de fallada únics.
- **Chukwa**: recopilació de dades en sistemes distribuïts.
- **HBase**: base de dades distribuïda i escalable.
- **Hive**: magatzem de dades que proporciona resum de dades i consultes *ad hoc*.
- **Mahout**: entorn d'aprenentatge automàtic i *data mining*.
- **Pig**: llenguatge de flux de dades d'alt nivell.
- **Spark**: motor de càlcul en memòria i general per a les dades de Hadoop.
- **Tez**: entorn de programació de flux de dades generalitzat per a grafs dirigits.
- **ZooKeeper**: servei de coordinació d'alt rendiment per a aplicacions distribuïdes.

En relació amb l'arquitectura de Hadoop i de MapReduce, es pot consultar la documentació d'E.Coppa [Hao], però en resum les capes i els mòduls de Hadoop queden representats a la figura següent:



On:

a) Entorn MapReduce: capa de programari que permet l'execució d'aplicacions amb aquest paradigma.

b) YARN (*Yet Another Resource Negotiator*): és la part responsable de gestionar els recursos (CPU, memòria, etc.) que utilitzaran les aplicacions; es compon de dos grans mòduls:

- *Resource Manager*, RM (un per clúster), que actua com a supervisor i coneix on estan situats els *workers* i de quants recursos disposa. Inclou una sèrie de serveis, però el més important és el *Resource Scheduler* (RS), que decideix com i a qui assignar-los.

- *Node Manager* (> 1 per clúster) és el *worker* de la infraestructura i manté informat el *Resource Manager* sobre el seu estat; gestiona la memòria i *vcores*, que poden ser assignats sota la gestió d'RS, que decidirà com s'utilitza aquesta capacitat a través de fraccions de la capacitat de l'NM anomenades *containers*.

c) HDFS Federation: és la part de l'entorn responsable de proveir emmagatzematge permanent, fiable i distribuït; és utilitzat per emmagatzemar les entrades i sortides de l'aplicació (no les intermèdies). Es pot complementar amb altres opcions d'emmagatzematge *cloud* (per exemple, S3, Google *Storage Objects*, etc.).

d) *Clúster*: és el conjunt de nodes de la infraestructura.

És important destacar que la infraestructura YARN i l'HDFS estan totalment desacoblats; sobre aquests es podrien executar altres entorns, encara que, de moment, és l'únic que està implementat.

En el cicle de vida de l'aplicació, gestionat per YARN, intervenen tres mòduls: *job submitter* (client), *resource manager* (supervisor o màster) i *node manager* (*worker*), per la qual cosa el cicle de treball d'una aplicació seria:

1) el client envia l'aplicació a l'RM,

2) aquest sobre la base de la informació d'RS,

3) assigna el contenidor,

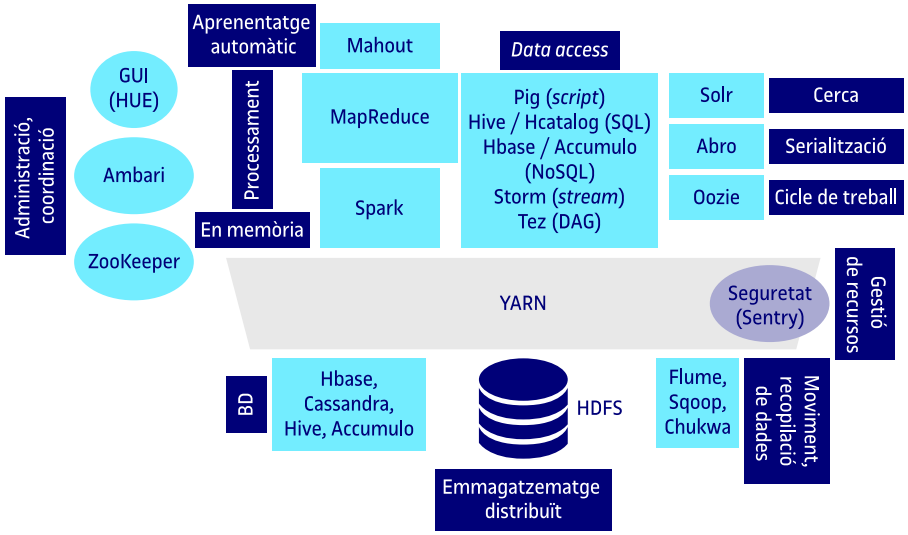
4) que, a més, contacta amb el *node manager*,

5) que, al seu torn, llança el contenidor

6) i aquest executa l'aplicació.

A la documentació [Hao] esmentada abans es recomana analitzar l'anatomia d'una aplicació MR i com interactuen cadascuna de les seves fases amb cadascun dels mòduls, així com els detalls de cadascun d'ells, les tasques que es generen i la granularitat del model de computació derivat de submòduls de l'RM i NM [Ydo].

La figura següent mostra un entorn Hadoop amb els seus actors més importants i el seu rol (un dels possibles) dins de la plataforma.



Els grans proveïdors de IaaS/PaaS/SaaS disposen d'entorns basats en MR, com Amazon EMR (Elastic MapReduce), Google Cloud DataProc, Azure HDInsight i IBM Hadoop (Open Platform Hadoop Docker). Quan es va actualitzar aquest contingut (desembre del 2021), el mercat de les distribucions específiques ha quedat monopolitzat per Cloudera, ja que unes altres de les dues plataformes existents i molt utilitzades en anys anteriors, com HortonWorks o MapR, han estat absorbides per altres companyies (Cloudera el 2018 i HP el 2019, respectivament). Addicionalment, i un dels problemes associats a Cloudera, és que, malgrat tenir gran part del programari *open-source*, l'octubre del 2021 va publicar una nova «metodologia» per accedir per subscripció a tot el seu programari.

D'acord amb les recomanacions de Gartner a Hadoop Distributions Reviews 2021 | Gartner Peer Insights (es recomana també analitzar la categoria Data Science and Machine Learning [ML] Platforms, atès que algunes d'aquestes plataformes estan classificades en diferents àmbits), es poden trobar diferents plataformes, a més de les esmentades, com les següents:

- SparkED (Alteryx) amb llicències per a educació
- Dataiku-DSS, amb una versió gratuïta de la seva plataforma (disponible per a tots els SO, *cloud* i com a MV també).
- KNIME Analytics Platform (*open-source*) amb extensions per a Spark.
- MapR: si bé la plataforma (tant la tecnologia com la propietat intel·lectual) va ser venuda a HP el 2019 per les dificultats financeres, es pot (desembre del 2021) descarregar i instal·lar la versió6.1de MapR o la versió sandbox per a *virtualbox/vmware*, o l'última versió 6.2 (*docker*) des d'HPE.
- Oracle Big Data Lite Virtual Machine (inclou Cloudera 5.x i tots els productes específics d'Oracle), si bé no és apta per a fer proves de concepte (a causa de la mida –la descàrrega són +22 GB–).
- Serveis de Hadoop as a Service (HaaS), com Qubole (30 dies gratuït).

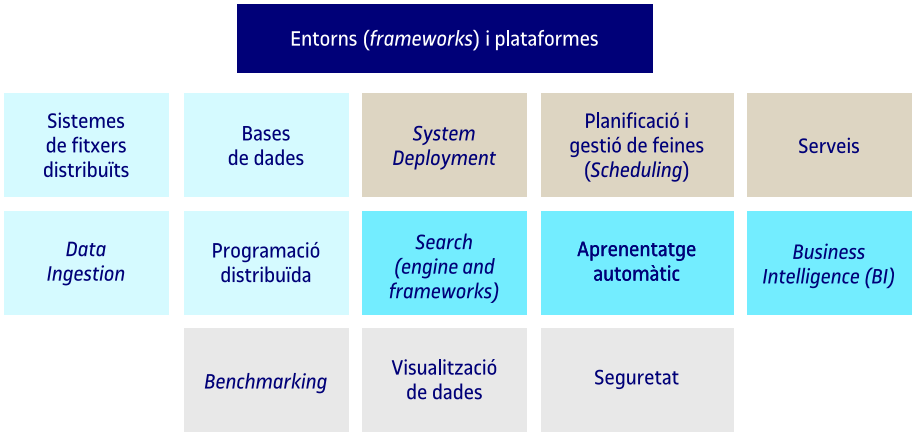
D'altra banda, hi ha algunes distribucions com Pivotal HD que ja són a *End of Availability* (*EoA*) (febrer del 2017); però es pot trobar alguna MV disponible a GitHub.

També és important considerar que no tot el tractament de les dades massives se centralitza sobre Hadoop, sinó que hi ha al *cloud* altres eines com, per exemple, Azure Intelligence + Analytics, AWS Big Data, Gloogle Cloud Big Data Products, entre altres.

1.2. Eines

La «revolució» del *big data* ha generat una gran quantitat de mètodes i eines (o evolucions de les ja existents) que han sorgit en diverses categories per gestionar volums massius de dades. És molt interessant el mapa d'eines i entorns (*landscape*), en la del 2016, generat per FirstMark [Bds] i l'actualització el 2021 (de Red Hot: The 2021 Machine Learning, AI and Data (MAD) Landscape) i l'extensa llista d'AweSome-Bigdata [Awd], on es pot trobar **tot** (o una major part) del que conforma l'ecosistema del *big data* i *cloud computing*.

Sense ànim de repetir el mapa o la llista, a continuació es mostra una gran divisió per categories i s'esmenten algunes de les més importants/habituals (llista no exhaustiva i majoritàriament *open-source*, però s'hi inclouen algunes amb llicències propietàries/*freemium*/ús no comercial).



1.2.1. Entorns (*frameworks*) i plataformes

- Apatxe Hadoop, entorn per al processament distribuït que inclou MapReduce (*parallel processing*), YARN (*job scheduling*) i HDFS (*distributed file system*).
- Cloudera, plataforma Hadoop *open-source*.
- Hortonworks (integrada a Cloudera).
- MapR, plataforma per a dades massives basada en Hadoop (avui propietat d'Hewlett Packard Enterprise), però en la versió 6.1 de la qual és possible descarregar des de https://mapr.com/download/ el sanbox, o la 6.2 (*docker*) des d'HPE.
- Tigon, entorn per al processament d'altes prestacions de fluxos de dades (*stream*) en temps real que utilitza Hadoop i Hbase.
- Pachyderm, plataforma d'emmagatzematge construïda sobre Docker i Kubernetes per proveir processament de dades i anàlisis.
- Solucions empresarials per a dades massives (només algunes d'aquestes): Amazon, Google, IBM Hadoop, Microsoft, Oracle, SAS Hadoop, TeraData, Vertica HP.

1.2.2. Sistemes de fitxers distribuïts

- Apache HDFS, sistema per emmagatzemar fitxers de mida gran en múltiples màquines.
- BeeGFS, Disc DDFS, Google GFS, Baidu File System, GlusterFS, Quantcast File System QFS, *distributed file systems*.
- Ceph Filesystem, sistema de fitxers compatible Posix de mida il·limitada.
- GridGain, sistema de fitxer *in-memory* compatible amb Hadoop.
- Lustre file system, *distributed filesystem* d'altes prestacions.

1.2.3. Bases de dades

RDBMS

- MySQL/MariaDB, popular base de dades en les versions d'Oracle (amb llicència per a alguns casos) i la versió *open-source* compatible, dissenyada i implementada pels mateixos desenvolupadors d'MyQSL.
- PostgreSQL BD, molt popular i amb excel·lents prestacions *open-source*.

Document Data Model

- Crate Data, emmagatzematge de dades massives en temps real *open-source*.
- MongoDB, gestor de bases de dades NoSQL orientat a documents.
- RavenDB, BD transaccional orientada a documents *open-source*.
- RethinkDB, BD orientada a documents per a *realtime-web*.

Key Map/Value Data Model

- Aerospike, BD NoSQL *in-memory* i *open-source*.
- Apache Accumulo, emmagatzematge de dades distribuït basades en *key/value* construïdes sobre Hadoop.
- Apache Cassandra, emmagatzematge de dades distribuït orientat a columna.
- Apache HBase, emmagatzematge de dades distribuït orientat a columna.
- Baidu Tera, BD a *internet-scale*.
- Google BigTable, BD NoSQL orientat a columna de Google.
- Hypertable, emmagatzematge orientat a columna *open-source*.
- InfiniDB utilitza una interface MySQL i *massive parallel processing* per paral·lelitzar *queries*.
- Redis, emmagatzematge *in memory* orientat a *key/value*.
- Tephra, transaccions per a HBase.

Graph Data Model

- Apache Giraph, implementació de Pregel sobre Hadoop.
- Apache Spark Bagel, implementació de Pregel sobre Spark.
- DGraph, BD escalable i distribuïda orientada a grafs.
- EliasDB, BD «lleugera» orientada a grafs sense llibreries de tercers.
- Google Cayley, BD orientada a grafs *open-source*.

Columnar Databases

- MonetDB, DB orientada a columnes.
- Parquet, emmagatzematge orientat a columnes per a Hadoop.
- Google BigQuery, *datawarehouse* al *cloud* per a *big data*.
- Amazon Redshift, emmagatzematge basat en columnes per a *cloud* d'Amazon.

- IndexR, emmagatzematge *open-source* basat en columnes per a anàlisis en temps real de dades massives.

Processament SQL-*like*

- Apache HCatalog, gestió de l'emmagatzematge i taules per a Hadoop.
- Apache Hive, *datawarehouse SQL-like* per a Hadoop.
- Apache Phoenix, capa SQL per a HBase.
- Cloudera Impala, entorn per a anàlisi interactiva.
- Spark Catalyst, entorn d'optimització de *queries* per a Spark i Shark.
- SparkSQL, entorn per utilitzar dades estructurades sobre Spark.

1.2.4. *Data ingestion*

- Apache Chukwa, sistema de recol·lecció de dades.
- Apache Flume, servei per manejar gran quantitat de dades de *log*.
- Apache Kafka, sistema de missatges de dades amb mètode de *publish-subscribe*.
- Apache Sqoop, eina de transferència de dades entre Hadoop i un repositori de dades estructurat.
- Embulk, carregador de dades *open-source* que ajuda a la transferència en BD, repositoris, formats i serveis *cloud*.
- Fluentd, eina per recollir esdeveniments i *logs*.
- Heka, sistema de processament *open-source* de *streams*.
- Logstash, eina per manejar esdeveniments i *logs*.

1.2.5. Programació distribuïda

- Apache Flink, entorn de processament de *streams* distribuït i d'altes prestacions (Stratosphere).
- Apache MapReduce, model de programació per al processament de dades massives utilitzant un algoritme paral·lel/distribuït sobre un clúster/*cloud*.
- Apache Pig, llenguatge d'alt nivell per descriure anàlisis de dades per a Hadoop.
- Apache Spark, còmput distribuït *in-memory*.
- Apache Storm, processament de *streams* (de Twitter).
- Apache Samza, processament de *streams* basats en Kafka & YARN.
- Concurrent Cascading, processament/analítica de dades sobre Hadoop.
- Datasalt Pangool, entorn alternatiu a MapReduce.
- Onyx, còmput distribuït en el *cloud*.
- OpenMPI, entorn de programació per pas de missatges.
- Pydoop, MapReduce i HDFS API per a Hadoop a Python.

1.2.6. *System deployment*

- Apache Ambari, entorn per a gestió de Hadoop.
- Apache Bigtop, entorn de desenvolupament per a Hadoop.

- Apache Helix, entorn de gestió de clústers.
- Apache Mesos, gestor de clústers.
- Apache YARN, gestió de clústers.
- Apache Tez, entorn per a l'execució de grafs dirigits complexos i construïts sobre YARN.
- Brooklyn, entorn que simplifica el desplegament i la gestió d'aplicacions en clústers.
- Buildoop, similar a Apache BigTop.
- Marathon, entorn Mesos per a execucions/serveis de llarga durada.

1.2.7. Planificació i gestió de tasques (*scheduling*)

- Apache Aurora, servei de planificació que s'executa sobre Apache Mesos.
- Apache Falcon, entorn de gestió de dades.
- Apache Oozie, planificador de tasques de *workflow*.

1.2.8. Serveis

- Apache Avro, sistema de serialització de dades.
- Apache Karaf, *runtime* OSGi (*Open Services Gateway Initiative*) que s'executa sobre qualsevol entorn OSGi.
- Apache Thrift, entorn per construir protocols binaris.
- Apache Zookeeper, servei centralitzat per a la gestió de processos.

1.2.9. *Machine learning*

- convnetjs, aprenentatge profund a Javascript.
- Deeplearning4j, aprenentatge profund per a Java, Scala, Clojure.
- Mahout, llibreria d'aprenentatge automàtic per a Hadoop.
- MOA, aprenentatge automàtic sobre *big data stream* en temps real.
- Spark MLlib, aprenentatge automàtic per a Spark.
- WEKA, entorn d'aprenentatge automàtic.

1.2.10. *Search* (*engine and frameworks*)

- Apache Lucene, llibreria de cerca.
- Apache Solr, plataforma de cerca basada en Apache Lucene.
- Elassandra, projecte derivat d'Elasticsearch sobre Apache Cassandra.
- ElasticSearch, plataforma de cerca i analítica basada en Apache Lucene.
- Sphinx Search Server, entorn de cerca de text.

1.2.11. *Business intelligence* (BI)

- Qlik, plataforma de *business intelligence and analytics* (versió bàsica gratuïta).
- Redash, SpagoBI, plataformes *open-source* per a BI.

- Jethrodata, plataforma interactiva d'anàlisis de dades massives.

1.2.12. *Benchmarking*

- Apache Hadoop Benchmarking, test per analitzar prestacions de Hadoop.
- Berkeley SWIM Benchmark, exemples reals de dades massives.
- PUMA Benchmarking, conjunt de tests per a aplicacions MapReduce.

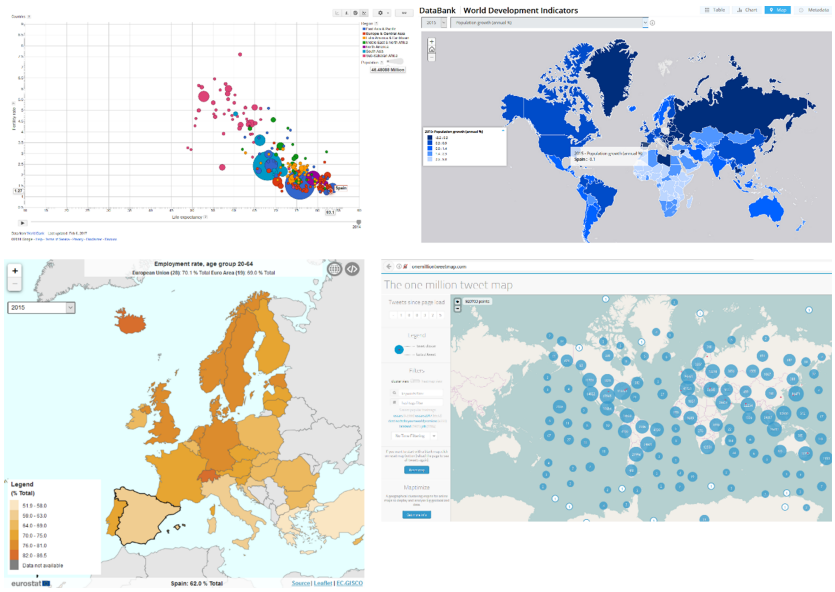
1.2.13. Seguretat

- Apache Eagle, monitoratge en temps real.
- Apache Knox Gateway, accés únic segur a clústers Hadoop.

1.2.14. Visualització de dades

- Llibreries (només alguns exemples): Bokeh, Chart.js, Chartist.js, Crossfilter, Cubism, DC.js, D3, D3Plus, D3.compose, DyGraphs, Envisionjs, Leaflet, NVD3, Plotly.js, Plot.ly, Sigma.js
- Plataformes: Freeboard, Fusion Tables (en línia), Gephi, Grafana, Graphite, Lumify, Mondrian, Redash

Exemples de com la visualització mostra les dades massives es poden veure a les gràfiques que presenta Google en un servei llançat el 2010 anomenat Public Data Explorer per representar diferents fonts de dades obertes (*open data*) com, per exemple, del World Bank, Eurostat i unes altres grans fonts de dades. A continuació, es mostren quatre exemples de visualització de dades massives on els tres primers són l'esperança de vida processada per Google, el creixement de la població mundial del WB i la taxa d'ocupació d'Eurostat sobre dades preprocesades i les respectives interfícies, mentre que l'últim és un exemple de dades massives però amb processament en temps real sobre els tuits emesos en el món (amb la possibilitat de filtrar-los i saber de què s'està parlant a escala mundial), en el mapa de *The one million tuit map* de Maptimize.



Com a exemples de dades massives es pot consultar:

- Amazon Public DataSets grans (i famosos) *datasets* (per exemple, el genoma humà).
- AwesomePublicDatasets, referència als grans repositoris de dades.
- Commoncrawl, informació sobre 3.140 milions de pàgines web i al voltant de 250 TiB (WebDataCommons amb Hyperlink Graphs).

2. Distribucions Hadoop i casos d'ús

En aquest apartat, es mostrarà un exemple de processament de dades massives basat en Hadoop utilitzant el repositori font. Si bé hi ha algunes distribucions rellevants que encapsulen les diferents eines de l'entorn Hadoop (segons https://www.gartner.com/reviews/market/hadoop-distributions, per exemple, Cloudera –HortonWorks–, MapR), la situació ha canviat des que es va escriure el document original (2016) i aquesta actualització (desembre de 2021). Els exemples es deixen com a documentació i casos d'ús, però no s'han actualitzat, ja que el 2016 aquestes distribucions tenien versions obertes i es podien utilitzar sense llicència; no obstant això, ara (a més de l'adquisició d'Hortonworks per part de Cloudera i la de MapR per HPE) han canviat les llicències i Cloudera+Hortonworks s'obté per subscripció de pagament.

2.1. Instal·lació de Hadoop sobre una màquina virtual Ubuntu

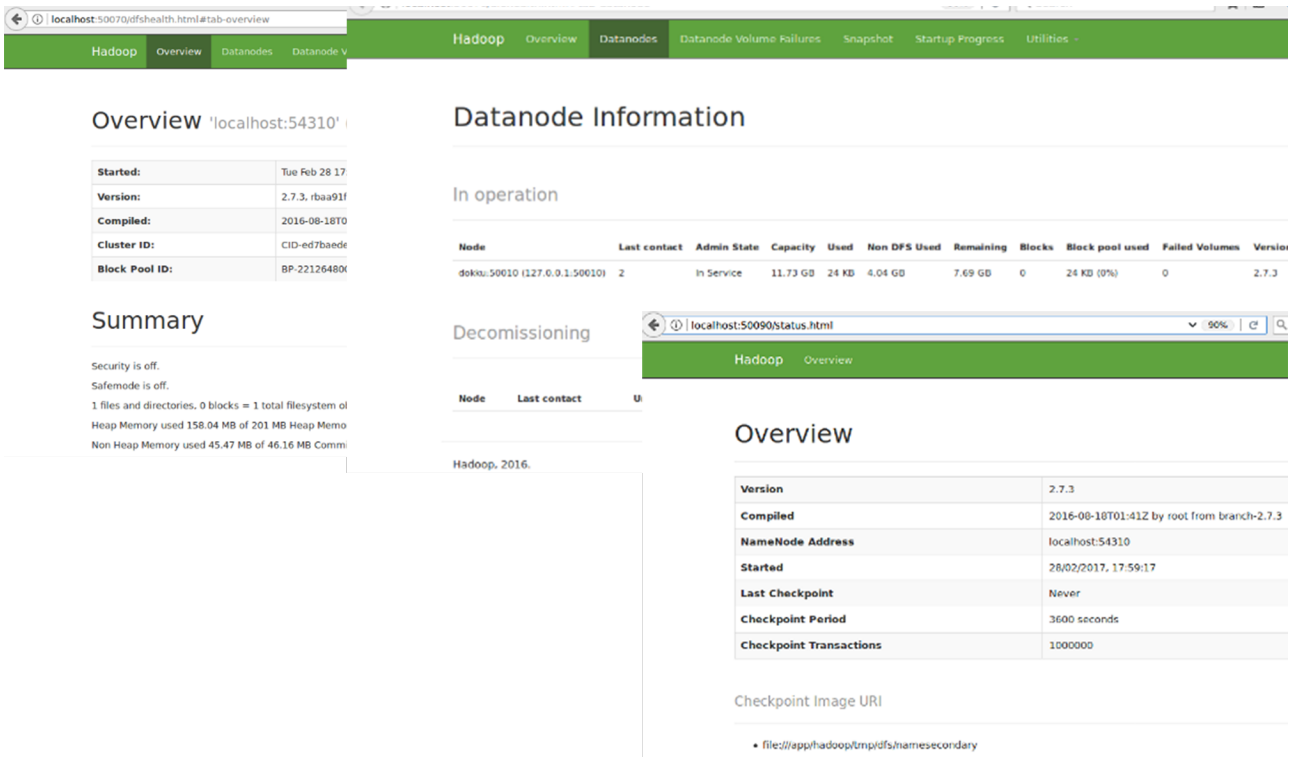
En aquesta primera prova s'instal·larà Hadoop amb HDFS en mode **pseudodistribuït** sobre una MV KVM (Ubuntu, 3 GB Ram, 15 GB de disc, però es pot executar amb màquines més petites que només tinguin SSH i LXDE+Firefox, si es vol accedir a les pàgines de gestió/monitoratge). Hadoop pot funcionar en tres modes: local (standalone), pseudo-distributed i fully-distributed, on en mode local total s'inclou en un procés Java i és útil per a depuració; en el mode pseudodistribuït cada *daemon* és un procés Java i s'utilitza per tenir la funcionalitat de Hadoop, però sense disposar de la infraestructura corresponent, ja que tot s'executarà en un node, i finalment la totalment distribuïda per a màquines en producció [Hui][Hsn].

Per a la instal·lació, cal seguir les instruccions (per a qualsevol de les modalitats) des de https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/singlecluster.html.

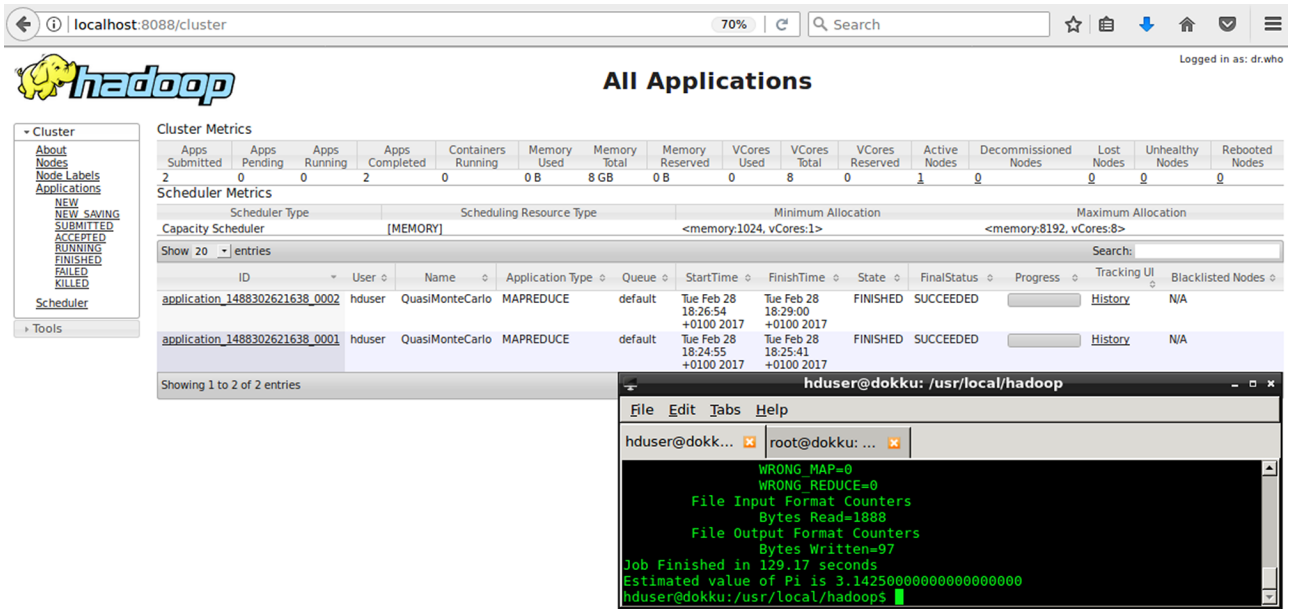
- Per iniciar l'entorn s'utilitza l'*script* **start-dfs.sh** i **start-yarn.sh**, i es podrà consultar utilitzant un navegador amb la URL http://localhost:50070/.

- Per detenir l'execució, **stop-dfs.sh** i **stop-yarn.sh**.

- Després d'haver iniciat YARN, a la URL http://localhost:50070/ es podran veure les característiques de l'entorn, l'estat del *SecondaryNode* i els *logs*. Com es mostra a continuació.



Per provar l'entorn s'executarà una aplicació de proves (càlcul de Pi utilitzant un mètode de Monte Carlo amb MapReduce) i aquest es visualitzarà a l'entorn del *Resource Manager* a través de la URL http://localhost:8088/, en la qual es podrà veure l'estat de les tasques durant la seva execució, com es mostra a la figura següent.



Per completar aquesta prova de concepte, s'utilitzarà una aplicació que compta paraules d'un text, es generarà l'entrada a HDFS i es compilarà i executarà l'aplicació [Mrt].

Primer, es crearà el fitxer sobre el qual s'aplicarà MR; per a això, s'ha generat un text del LoreIpsum i s'ha desat al fitxer *li.txt*, i després s'ha creat el directori sobre l'HDFS i s'ha copiat el fitxer (encara que es podria haver utilitzat l'ordre *put* també) i verificat.

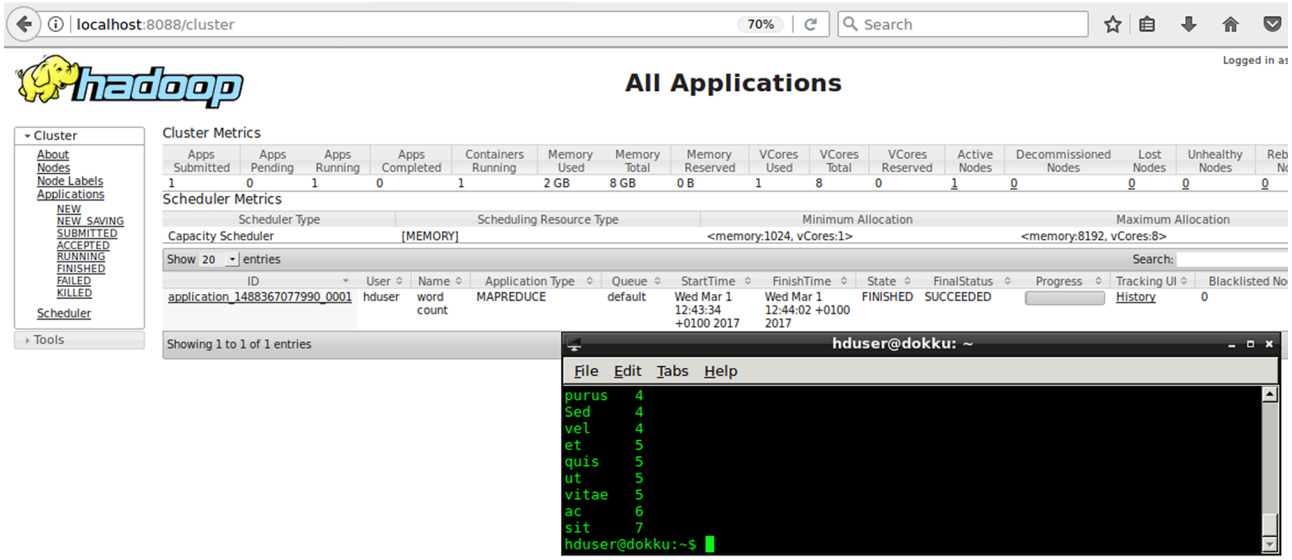
```
hdfs dfs -mkdir -p /user/hduser/in
hdfs dfs -copyFromLocal ./li.txt /user/hduser/li.txt
hdfs dfs -ls
hdfs dfs -cat /user/hduser/in/li.txt
```

A continuació, s'ha creat el fitxer WourdCount.java amb l'exemple de [Mrt], s'han inicialitzat les variables, compilat, verificat que els fitxers d'entrada existeixen (utilitzant *hadoop* en lloc de *hdfs*) i executat.

```
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
hadoop com.sun.tools.javac.Main WordCount.java
jar cf wc.jar WordCount*.class
hadoop fs -ls /user/hduser/in/
hadoop fs -cat /user/hduser/in/li.txt
hadoop jar wc.jar WordCount /user/hduser/in /user/hduser/output
```

La imatge a continuació mostra l'execució i la sortida ordenada dels comptadors de paraules després de l'execució al fitxer de l'HDFS i visualitzat amb:

```
haddop fs -cat /user/hduser/output/part-r-0000 | sort -k2
```



Si es vol experimentar, es podria complementar l'experiment agregant una altra MV i creant un cluster Hadoop sobre aquestes.

Una altra opció per tenir un entorn (basat en *docker*) en el qual es pugui experimentar amb els diferents components de l'arquitectura Hadoop és Apache BigTop, que permetrà instal·lar o crear els paquets sense preocupar-se per dependències o versions.

3. Spark i casos d'ús

Spark és un motor de processament optimitzat de dades en memòria que pot executar operacions amb dades de Hadoop (a HDFS) i amb un rendiment millor que MapReduce.

També pot processar dades en altres entorns (com clústers), i en minimitzar les lectures i escriptures de disc, ofereix un rendiment molt alt a col·leccions de dades dinàmiques i complexes i brinda una funcionalitat interactiva com a suport de consultes específiques (*ad hoc*) d'una manera escalable. Spark estén el paradigma de processament estàtic de dades emmagatzemades i per seqüències (lots o *batch*), que és el que utilitza MapReduce cap a aplicacions dinàmiques i en temps real o sobre fluxos de dades.

Spark suporta programes en diferents llenguatges de programació, que inclouen Java, Scala, Python i R, i permet interactuar amb dades incloses en bases de dades SQL.

Aquest nou enfocament al processament de grans volums de dades que proposa Spark està compost per una infraestructura de programari juntament amb tècniques de desenvolupament d'aplicacions simples però potents, per desenvolupar i desplegar aplicacions en *big data*. Spark es considera com una evolució de les dades massives que permet treballar amb diferents tipus de fonts de dades utilitzant diferents llenguatges de programació.

Si bé MapReduce ha sigut durant anys la metodologia de desenvolupament de programari per al processament distribuït per lots (*batch*) de dades massives, aquest tipus de processament no és la resposta per a totes les situacions computacionals. Amb l'interès de buscar millors prestacions al MapReduce, els investigadors van començar a elaborar noves propostes i dues de les més significatives han sigut **Apache Drill** i **Spark**.

Drill és un motor de llenguatge de consulta estructurat (SQL) d'alt rendiment destinat a proporcionar funcions d'exploració de dades, mentre que Spark va ser dissenyat per ser un motor de processament de propòsit general per a tasques interactives, per lots i de flux, que eren capaços d'aprofitar els mateixos tipus de recursos de processament distribuïts que fins ara havien impulsat les iniciatives MapReduce.

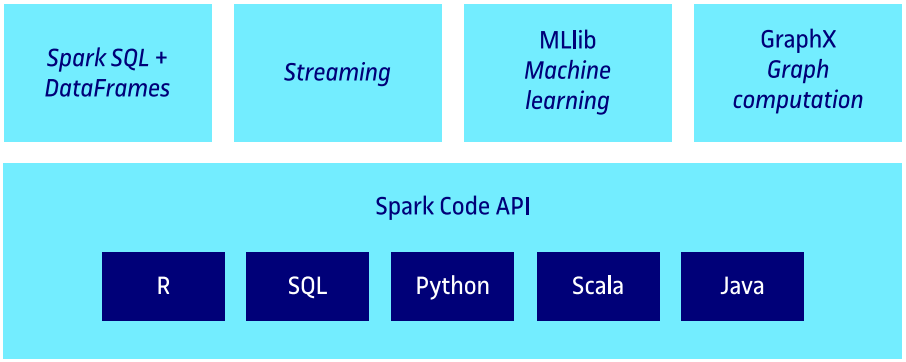
No han estat els únics intents de millorar les prestacions sobre les dades massives, ja que també es poden esmentar: **Apache Pig** (un llenguatge MapReduce especialitzat d'alt nivell), desenvolupat per Yahoo! per facilitar l'accés a les dades emmagatzemades a Hadoop; **Apache Hive**, que va aportar el poder d'SQL al MapReduce com a resultat de la recerca de Facebook per a interaccions de

dades més interactives i basades en SQL; **Apache Impala**, com a entorn de consultes de l'estil analítica de negocis per a Hadoop, o **Prest**, com un motor de processament distribuït d'alta velocitat per executar consultes SQL en bases de dades massives.

Partint des del disseny, Spark estava orientat a explotar el potencial, en particular la velocitat i l'escalabilitat, ofert per processament *in-memory* de dades, ja que l'algoritme de MapReduce tendeix a escriure contínuament dades intermèdies en el disc al llarg d'un cicle de processament. En aquest sentit, Spark pot oferir resultats amb la mateixa configuració de dades fins a cent vegades més ràpid que MapReduce.

A més, les seves premisses de disseny s'han concentrat a oferir **rendiment**, **simplicitat**, **facilitat d'administració** i **desenvolupament d'aplicacions més ràpid**, la qual cosa ha permès que una gran quantitat d'organitzacions i empreses l'adoptin ràpidament.

Com s'ha esmentat anteriorment, Spark utilitza un model simple de programació que pot ser a Python, R, Scala o Java, i disposa d'*application programming interfaces* (API) ben definides per a diferents casos d'ús, com Streaming, SQL, ML, DataFrames i Graph Processing. La figura següent mostra l'ecosistema de Spark:



3.1. Estructura de les dades Spark i casos d'ús

Un aspecte important en les dades massives és vincular i gestionar dades de diverses fonts. Aquesta tasca és àrdua i requereix molta dedicació i un còmput intensiu; aquest procés es coneix com extreure, transformar i carregar (*extract, transform, and load* –ETL–). Spark inclou mecanismes per realitzar la integració de proveïdors i plataformes de tecnologia d'emmagatzematge de dades i permet realitzar aquest procés de forma ràpida i senzilla pel seu processament en memòria i sense emmagatzemar les dades en el disc dur, la qual cosa aconsegueix eficiències des de ×10 fins a ×100, en relació amb les aplicacions clàssiques de MapReduce basades en disc.

Per aconseguir aquests resultats, Spark treballa amb diferents estructures de dades que han anat evolucionant en el temps: ***resilient distributed datasets*** (RDD), ***dataframes*** i ***datasets***. Inicialment, el 2011 Spark va ser desenvolupat amb RDD, però en veure l'auge de Spark en altres àmbits, els desenvolupadors van implementar *dataframes* el 2013 i *datasets* el 2015.

Els **conjunts amb resiliència de dades distribuïdes (*resilient distributed datasets* –RDD–)** és l'estructura de dades fonamental (inicial) de Spark. Permet emmagatzemar les dades distribuïdes en els múltiples nodes del clúster i fer el processament en paral·lel. És tolerant als errors, ja que, si es realitzen múltiples transformacions en l'RDD i després, per qualsevol motiu, hi ha un error en algun node, l'RDD és capaç de recuperar-se automàticament.

Més detalladament, són estructures en memòria destinades a contenir la informació que es vol carregar i després processar per Spark. Se les etiqueta «**amb resiliència**» perquè mantenen un registre del seu historial (per exemple, modificacions i eliminacions) perquè es puguin reconstruir, si hi ha errors durant el seu processament, i a més són **distribuïdes** perquè el *data set* es particiona i es reparteixen entre els diferents nodes de l'arquitectura subjacent del clúster per augmentar l'eficiència mitjançant el processament paral·lel.

Una aplicació de Spark pot crear una estructura RDD i després carregar dades que poden provenir de gairebé qualsevol font de dades, com Hadoop, Apache Cassandra, Amazon S3, Apache Hbase i bases de dades relacionals, entre altres.

Un RDD es pot crear de tres formes: **a)** paral·lelitzant una col·lecció de dades existent; **b)** llegint un fitxer de dades extern, o **c)** creant RDD a partir d'un RDD ja existent, per exemple:

```
# a) paral·lelització de la recopilació de dades
my_list = [1, 2, 3, 4, 5, 6]
my_list_rdd = sc.parallelize (la_meva_llista)

# b) Referència a un fitxer de dades extern
file_rdd = sc.textFile ("ruta_del_fitxer")
```

Una pregunta que es pot fer és quan utilitzar un RDD. És recomanable utilitzar l'RDD en les situacions següents:

1) Quan es volen fer transformacions de baix nivell en el conjunt de dades (més endavant es veuran transformacions sobre els RDD).

2) Quan les dades carregades no tenen un esquema i cal especificar manualment l'esquema de cada conjunt de dades en crear l'RDD.

Els ***dataframes*** es van integrar a Spark a partir de la versió 1.3 per superar les limitacions de l'RDD. Spark *dataframes* són la col·lecció distribuïda de dades, però organitzades a les columnes amb nom (admeten capçaleres) i permeten als desenvolupadors depurar el codi durant el temps d'execució, la qual cosa no és possible amb els RDD.

Els *dataframes* poden llegir i escriure les dades en diversos formats, com CSV, JSON, AVRO, HDFS i taules HIVE, i estan optimitzats per processar grans conjunts de dades per a la majoria de les tasques de preprocessament, per la qual cosa no cal escriure funcions complexes per part del desenvolupador. Per crear un *dataframe*, per exemple, a PySpark (Python per a Spark), fem el següent:

```
spark.createDataFrame (
   [
        (1, 'Pere'),
        (2, 'Anna'),
        .
        .
        .
        .
        (100, 'Alícia')
    ],
    ['id', 'Nom'] # etiqueta de les columnes
)
```

Els ***datasets*** són una extensió dels *dataframes* que inclouen els seus beneficis i els dels RDD, per la qual cosa permeten processar de manera eficient dades estructurades i no estructurades. És una estructura que permet un processament ràpid i proporciona una interfície segura, ja que el compilador de Spark validarà els tipus de dades de totes les columnes del conjunt de dades només durant la compilació i llançarà un error si hi ha alguna discrepància en els tipus de dades. Qui utilitzi RDD trobarà que el codi és molt similar, però notarà un increment notable en la velocitat de processament. Com que s'ha de verificar el codi durant la compilació dels *datasets*, només estan disponibles per a Scala i Java (no per a Python).

Els *dataframes* permeten un conjunt d'**operacions** per a la manipulació de dades estructurades (a continuació, se'n mostren algunes; consulteu la documentació de Spark). Abans de començar a treballar amb un *dataset*, cal crear una SparkSession (aquests paràmetres es creen automàticament, si s'hi accedeix per PySpark):

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('Spark Training').getOrCreate()
sc = spark.sparkContext
```

Per crear un *dataframe* des d'un fitxer CSV fem:

```
df = spark.read.format('csv').options(delimiter=',', header=True).load('/carpetas/archivo.csv')
```

Per convertir RDD a *dataframe*:

```
rdd = sc.parallelize([(1,2),(3,4),(5,6)])
rdf = rdd.toDF()
```

Per crear-lo utilitzant el mètode spark.createDataFrame:

```
tdf = spark.createDataFrame([('Pere',24),('Anna',43)],['nom','edat'])
tdf.printSchema()
root
|-- name: string (nullable = true)
|-- age: long (nullable = true)
```

Per mostrar les columnes del *dataframe*:

```
tdf.columns
['nom', 'edat']
```

Per mostrar les dades d'un *dataframe*:

```
tdf.show(5, truncate=False)
+------------+------+
|nom         |edat  |
+------------+------+
|Pere        |24    |
|Anna        |43    |
+------------+------+
```

Per als RDD, Spark permet dos tipus principals d'operacions: **transformacions i accions**.

Les transformacions inclouen filtratge, mapatge o manipulació de les dades, la qual cosa genera un nou RDD, ja que els RDD són immutables, és a dir, l'RDD original roman sense canvis. Aquesta inmutabilitat permet que Spark realitzi un seguiment de les modificacions que es van dur a terme durant la transformació i, per tant, desfer els canvis posteriors que s'han realitzat. Un altre punt interessant de la feina amb RDD és que les transformacions són «mandroses» (*lazy*), és a dir, no s'executen fins al moment en el qual es necessiten, per exemple, mitjançant una acció. Aquest comportament ajuda a millorar el rendiment en retardar els càlculs fins al moment en el qual es requereixen, en lloc de realitzar càlculs que podrien no ser necessaris segons el flux del programa.

Les accions són funcions per obtenir informació de les dades, però no les modifiquen, com a recompte, recuperació d'un element específic o agregació (unió d'informació no inserció). Quan una aplicació sol·licita una acció, Spark avalua l'RDD inicial i després crea instàncies d'RDD, en funció del que se li demana que faci, i observant quin lloc del clúster serà el més adequat, des del punt de vista de l'eficiència, per col·locar el nou RDD.

Com s'ha esmentat, les transformacions generen un nou RDD, i entre aquestes les més utilitzades que es poden trobar tenim les següents:

- **map(func):** retorna un nou RDD passant cada element de la font a través de la funció func.
- **filter(func):** retorna un nou RDD seleccionant aquells elements de la font que el seu func retorna true.
- **flatMap(func):** similar a map, però cada entrada pot ser mapada a 0 o més sortides, la qual cosa podria retornar una seqüència en lloc d'un sol ítem.
- **union(otherDataset):** retorna un nou *dataset* amb la unió de tots dos.
- **distinct([numTasks])):** retorna un nou RDD que conté aquells elements diferents de l'RDD original.
- **groupByKey([numTasks]):** si l'RDD està organitzat per claus i valor (K, V), retorna un nou *dataset* agrupat per K (K, Seq[V]).
- **reduceByKey(func, [numTasks]):** genera un nou RDD on els valors per a cada clau estan agregats per la func.
- **sortByKey([ascending], [numTasks]):** retorna un nou RDD ordenat.

Entre les accions més importants hi ha les següents:

- **reduce(func):** agrega els elements del *dataset* utilitzant func.
- **collect():** retorna tots els elements del *dataset*.
- **count()**: retorna el nombre d'elements del *dataset*.
- **take(n):** retorna un *array* amb els primers *n* elements del *dataset*.
- **saveAsTextFile(path):** escriu els elements del *dataset* com un fitxer de text en el sistema de fitxers local, en HDFS o en qualsevol altre sistema de fitxers suportat per Hadoop.
- **countByKey():** només disponible en RDD del tipus (K, V), retorna un mapa de (K, Int) tuples per a cada *key K*.
- **foreach(func):** executa una funció func sobre cada element del *dataset*.

3.2. Instal·lació de Spark, PySpark i Spider

PySpark és una llibreria de Spark escrita en Python per executar una aplicació Python usant les capacitats d'Apache Spark (PySpark és una API de Python per a Spark).

Spark bàsicament està escrit en Scala, però donada la ràpida adaptació a entorns de ciència de dades, es va desenvolupar l'API PySpark per a Python usant Py4J (llibreria de Java integrada dins de PySpark), i permet que Python interactuï en forma dinàmica amb objectes Java, per la qual cosa cal tenir instal·lat Java (es recomana la versió 8 o superior) per executar PySpark.

Per facilitar el desenvolupament de programes en aquesta prova de concepte, s'ha instal·lat sobre una màquina virtual, a més de Spark i PySpark, un IDE (*integrated development environment*) anomenat Spyder, que permet tenir un entorn d'edició, desenvolupament i execució sobre PySark (també és molt utilitzat Jupyter notebook). Per veure la configuració i integrar Sypder amb PySpark, podeu consultar la referència següent.

PySpark és una opció molt utilitzada en ciència de dades i aprenentatge automàtic, ja que hi ha moltes biblioteques de ciència de dades escrites en Python que inclouen NumPy o TensorFlow, entre altres. Apache Spark funciona en una arquitectura mestre-treballador (*master-worker*) en la qual el mestre es diu *Driver* i els treballadors, *Workers*. Quan executa una aplicació Spark, el *Driver* crea un **context**, que és un punt d'entrada a la seva aplicació, i totes les operacions (transformacions i accions) s'executen en els nodes treballadors. El *cluster manager* administra els recursos.

Entre els *cluster managers* més utilitzats per Spark es poden enumerar els següents:

- Standalone: és un *cluster manager* simple inclòs amb Spark per a una configuració fàcil d'un clúster.

- Apache Mesos: utilitza com *cluster manager*Mesos, que permet executar aplicacions Hadoop MapReduce i PySpark.

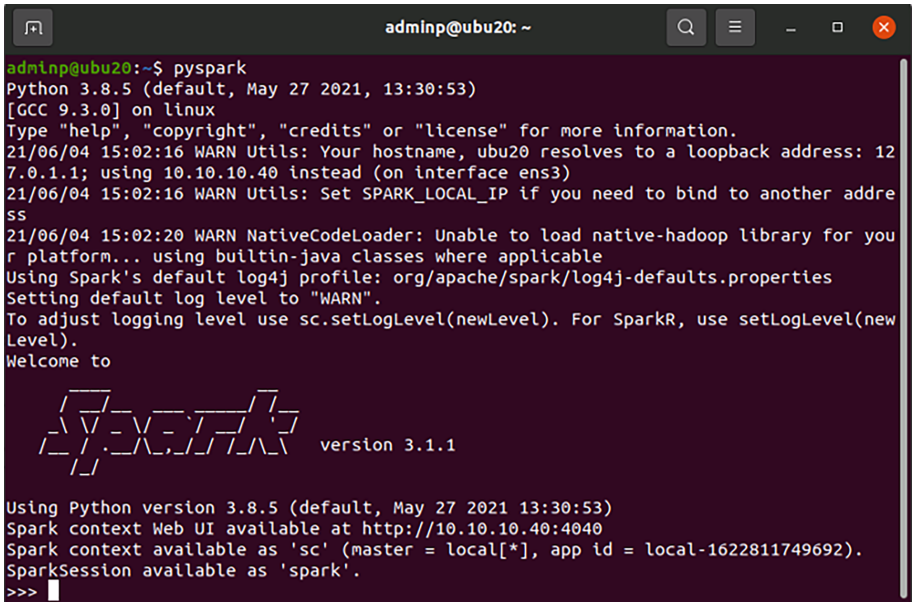
- Hadoop YARN: és el gestor de recursos de Hadoop 2/3 i normalment és el *cluster manager* més utilitzat.

- Kubernetes: és un entorn *open-source* per automatitzar, desplegar, gestionar i escalar aplicacions en *containers* (*docker*).

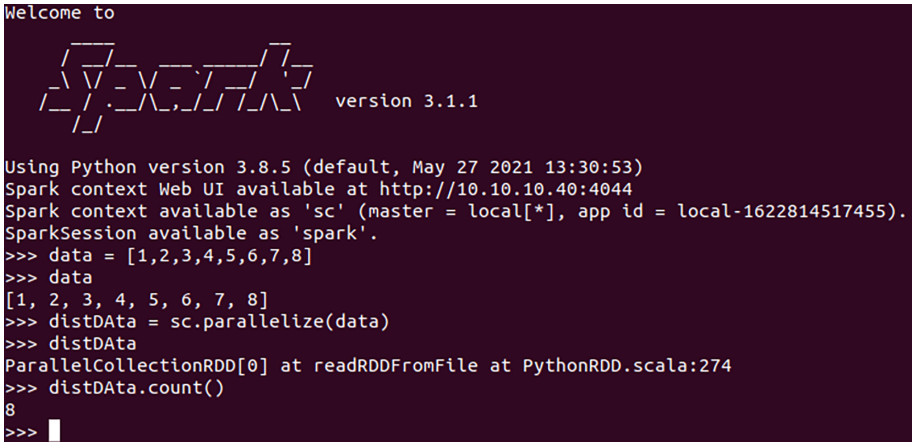
- local: no és realment un *cluster manager*, però és útil quan s'executa Spark sobre una màquina virtual o un ordinador, i és el que s'utilitzarà en aquestes proves de concepte. De manera local, se li pot indicar un paràmetre [k] per llançar *k workers* (idealment, igual al nombre de *cores* de la màquina virtual o de l'ordinador).

Com a informació addicional, s'ha de tenir en compte que hi ha una gran quantitat de llibreries i paquets que permeten estendre la funcionalitat de Spark i que es troben a https://spark-packages.org/.

Després de ser instal·lat i configurat, Spark inclou, a més de la consola, a la qual es pot accedir posant en un terminal **pyspark** una interfície web que permet analitzar el desenvolupament de les tasques en execució, posant com a URL en un navegador http://localhost:4040. Aquesta interfície mostrarà, entre altra informació: *Jobs*, *Stages*, *Tasks*, *Storage*, *Environment*, *Executors* i *SQL* per monitorar l'estat de l'aplicació en execució, com mostren les figures següents, en les quals disposa de l'*IDE Spyder* per executar les aplicacions en PySpark; però també es pot accedir a la consola interactiva de Spark simplement amb **pyspark**).



Com es pot interactuar amb Spark de manera fàcil? Simplement, executant **pyspark** i, quan es tingui el *prompt* > executar, per exemple (s'ha de veure que en iniciar pyspark es genera el context com **sc** i es crea una sessió com **spark**, per la qual cosa ja es pot entrar codi Python per processar les dades de forma interactiva):



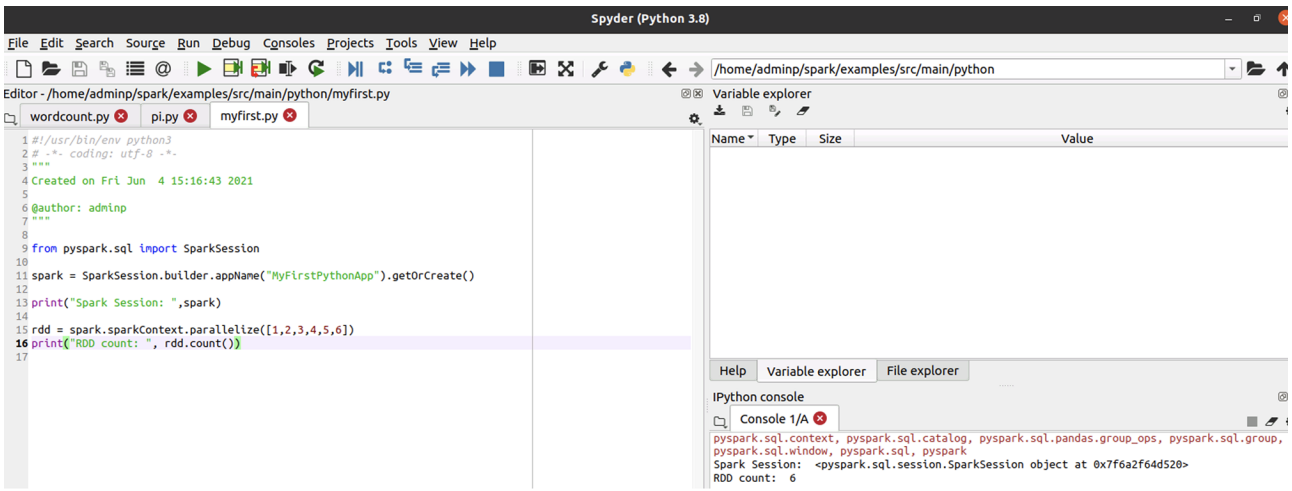
En aquest exemple s'ha creat un vector de dades (*data*), després s'ha **creat l'RDD** (*disDAta*) i paral·lelitzat les dades, sobre les quals després s'ha executat una **acció** per comptar les dades.

3.3. Programa en PySpark utilitzant Spyder (IDE)

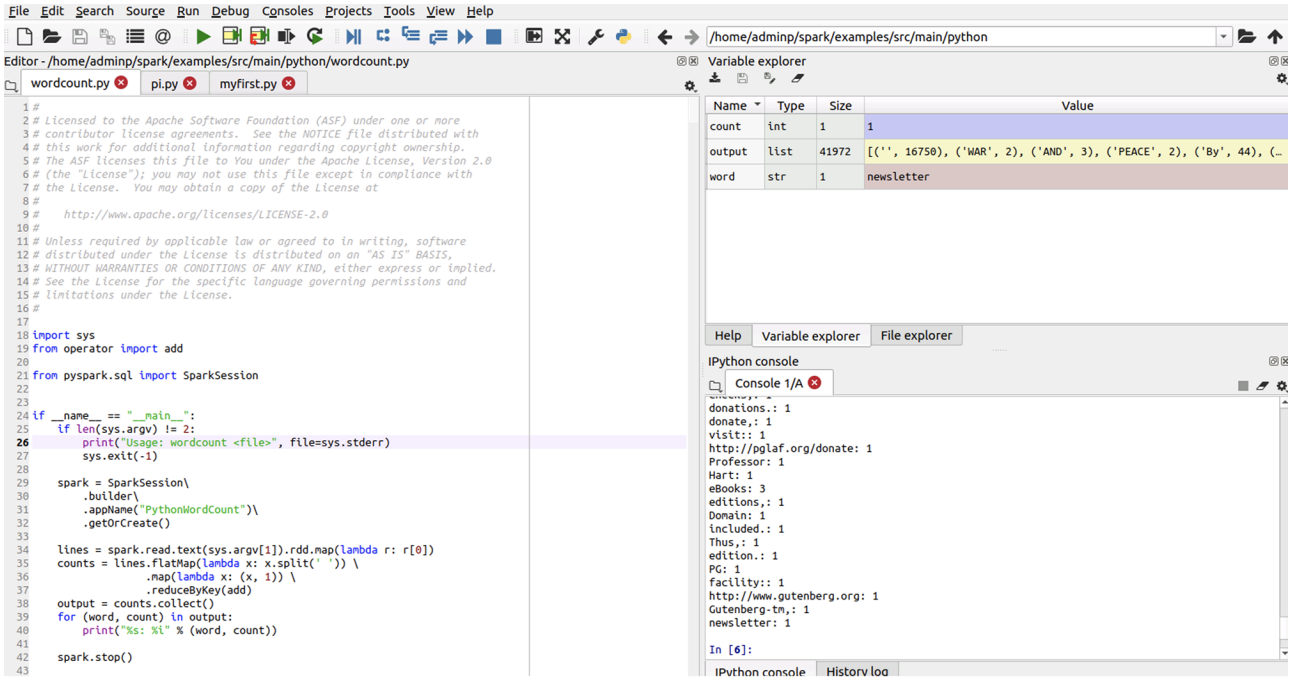
A la figura següent es pot veure un programa simple (*myfirst.py*). A continuació, es descriuen cadascuna de les línies:

```
9: importació de les llibreries
11: creació d'una sessió
13: impressió de valor de la variable Spark
15: creació d'un RDD distribuït amb un array de dades des d'1 a 6
16: impressió d'una acció (count) sobre l'RDD per comptar el nombre d'elements de l'RDD
```

Per executar el programa en Spyder, simplement s'obre el fitxer i després es fa un clic a la fletxa d'execució (▶). A la finestra dreta inferior es pot veure el resultat de l'execució.



En aquest exemple, es mostra el codi del programa *wordcount.py*, que és similar als que ja s'han tractat anteriorment.



En aquest, el codi de cada línia és el següent:

```
18,19,21: importació de les llibreries
25,26,27: funció d'error si no es proveeix un fitxer per processar
29: creació de la sessió de Spark
34: lectura del fitxer i creació d'un RDD amb cada línia del fitxer
35: creació d'un segon RDD amb les paraules separades per espais,
generació de la tupla (w, 1) i reducció per la clau (key),
en aquest cas la paraula
38: emmagatzematge de l'RDD en un array
39: impressió de les dades com a paraula: comptador
42: final de la sessió Spark
```

Per executar s'ha de fer clic al *play* (▶), però abans s'haurà d'indicar el fitxer d'entrada (en aquest cas, *war-peace.txt*); això es realitza en el menú *Run* > *Configuration per file* > marcar *Command line options*, i en el quadre de text s'ha d'incloure el directori on hi ha el fitxer d'entrada (en aquest cas, el directori txt/war-peace.txt). A la sortida es poden observar a la dreta, a la finestra superior, les variables utilitzades i el seu valor; i a la inferior, el resultat del processament.

4. Distribucions específiques

4.1. Cloudera (CDH)

CDH és una distribució *open-source* molt completa, provada i popular d'Apache Hadoop que, juntament amb altres projectes relacionats, ofereix la versatilitat de Hadoop en l'emmagatzematge escalable i el còmput distribuït juntament amb una interfície d'usuari basada en web, a més de processament per lots unificat, cerca/SQL interactiu i controls d'accés basats en rols [Cdo].

Nota de l'autor (actualització 2021)

Els casos d'ús en aquestes plataforma es mantenen com a documentació, ja que excepte a MapR que es pot actualitzar la versió 6.1 des de https://mapr.com/download/ el sanbox, o la 6.2 (docker) des d'HPE; a les de Cloudera i Hortonworks no s'hi pot accedir per canvis en la política de llicències l'octubre de 2021.

La plataforma CDH ofereix:

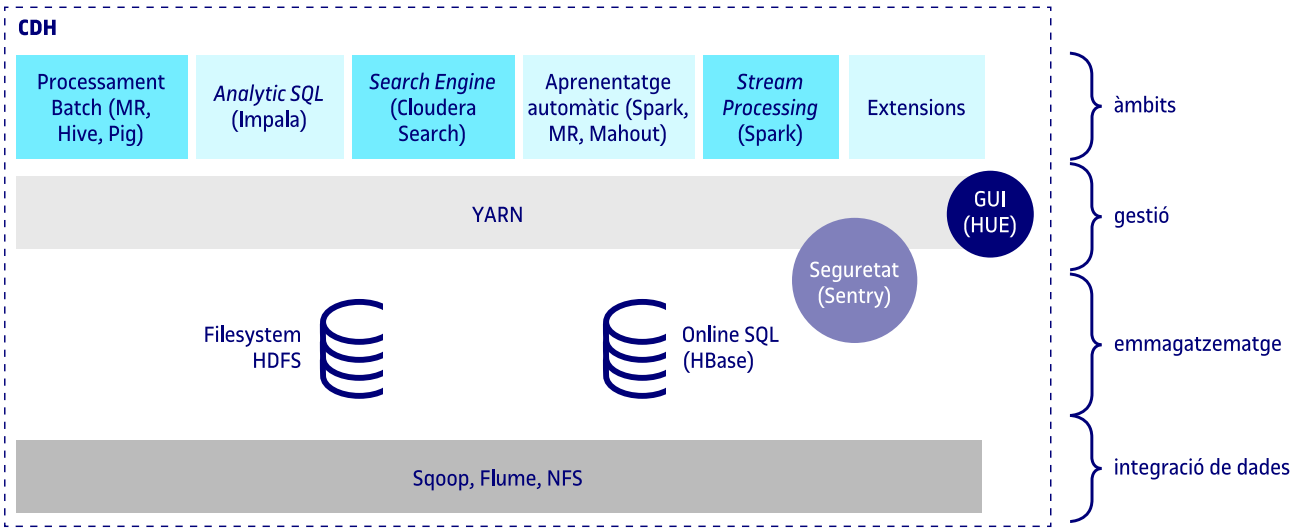
- flexibilitat (permet emmagatzemar qualsevol tipus de dades dins de diversos entorns de computació com processament per lots, SQL interactiu, cerca de text lliure, aprenentatge automàtic i càlcul estadístic),
- integració (desplegament ràpid de la plataforma Hadoop sobre una base de «llista per usar»),
- seguretat (s'han cuidat tots els aspectes d'integració i configuració per mantenir la confidencialitat de les dades processades),
- escalabilitat (permet integrar nous mòduls/projectes/aplicacions sobre la mateixa base),
- alta disponibilitat (integrada i fàcil de configurar) i
- compatibilitat (amb els sistemes actuals de la infraestructura de TU existent).

Els components de CDH (v.5.8.x) són:

- Apache Hadoop: entorn de processament/emmagatzematge distribuït de *big data*.
- Apache HBase: BD (registre/taules) que permet l'emmagatzematge escalable/distribuït per a *big data*.
- Apache Impala: entorn per al processament de consultes concurrents SQL amb baixa latència sobre HDFS, S3, o HBase.
- Apache Spark: processament en memòria per accelerar l'execució de les tasques en *big data*.
- Apache Avro: serialització de dades (estructures complexes, format binari, RPC).
- Apache Flume: recollida/agregació d'esdeveniments/*streams* en temps real sobre HDFS.
- Apache Hive: entorn per a lectura i escriptura de *big data* en emmagatzematge distribuït utilitzant SQL.
- Apache Kafka: servei de missatgeria distribuït i basat en el paradigma *publish-subscribe*.

- HUE: GUI basada en web per a la infraestructura Hadoop.
- Apache Oozie: planificació del *workflow* per a la gestió eficient de tasques Hadoop.
- Cloudera Search: cerca de text lliure (estil Google Search) sobre Hadoop.
- Apache Sentry: control d'accés (granular) basat en rols per a Hadoop.
- Apache Sqoop: transferència de dades (en ràfegues) eficient i escalable sobre HDFS.
- Apache ZooKeeper: servei de coordinació fiable i distribuït utilitzat en HBase.
- Apache Crunch: llibreria Java per escriure/depurar/executar *pipelines* MR.
- Apache DataFu: llibreria per a l'anàlisi estadística de *big data*.
- Kite SDK: API, exemples i documentació per construir aplicacions sobre Hadoop.
- Apache Parquet: representació de dades per columnes en forma comprimida/eficient per a Hadoop.
- Apache Mahout: llibreries per a ML (classificació, agrupació i filtratge col·laboratiu).
- Apache Pig: entorn de llenguatge d'alt nivell per a l'anàlisi de *big data*.

La figura següent mostra un esquema dels principals mòduls de CDH.



A més, Cloudera desenvolupa el Cloudera Manager (no és *open-source*, però es pot utilitzar sense llicència), que és un entorn sofisticat per al desplegament, l'administració i el monitoratge d'entorns CDH que, a través d'un *dashboard* (web), permet que el *sysadmin* pugui tenir tot el control sobre la infraestructura distribuïda de Hadoop.

Aquesta plataforma també disposa d'una API que permet obtenir informació de l'entorn o configurar el propi CM. Un altre paquet (orientat a Cloudera Enterprise, versió amb prestacions addicionals amb llicència de l'entorn Cloudera Hadoop) és Cloudera Director, que permet desplegar i gestionar la plataforma en el *cloud*.

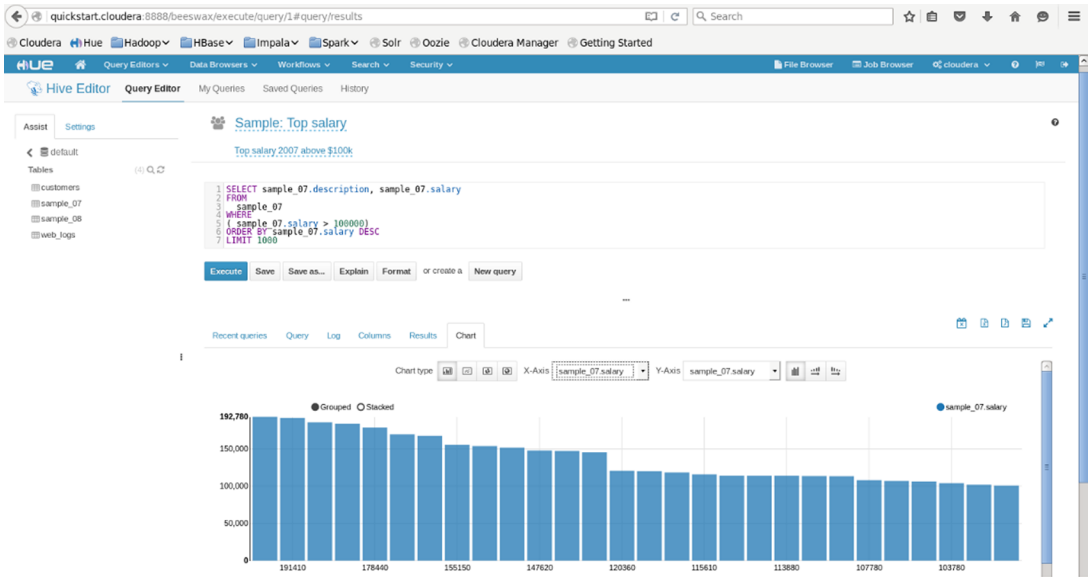
4.1.1. CDH sobre Virtualbox

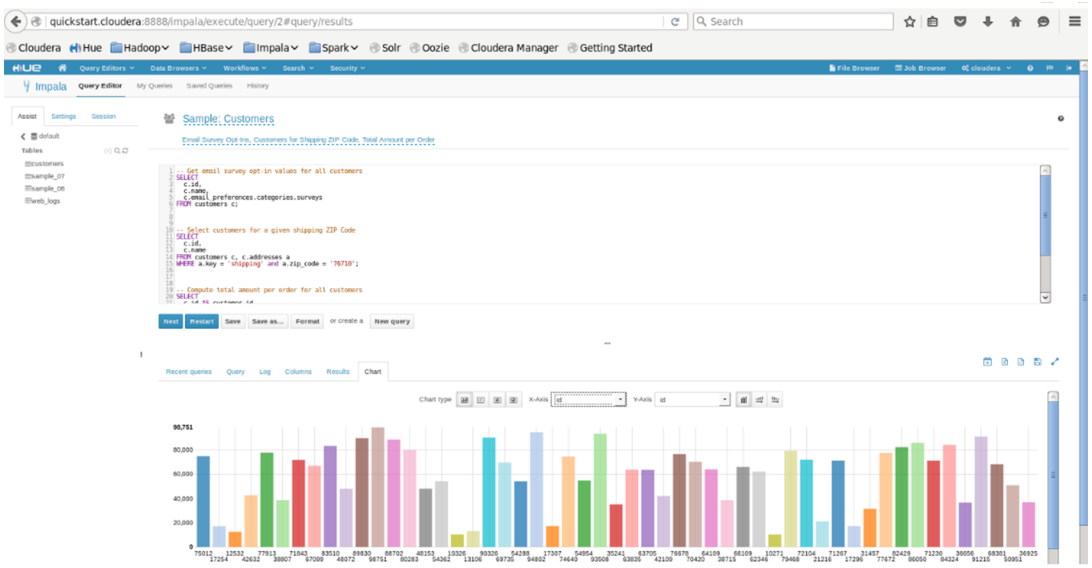
Per fer una prova de concepte amb la distribució Cloudera, es descarregarà l'MV (per a aquest cas s'ha utilitzat la de VirtualBox, però pot ser Docker, KVM, VMware), es descomprimirà el fitxer i importarà l'MV (fitxer OVF) i s'arrencarà l'MV. Per defecte, s'arrencarà CDH amb Hadoop (amb serveis Linux funciona sobre 4G de RAM o fins i tot menys) en *pseudo-distributed* amb un RManager i un node, i es podran veure sobre el navegador a l'inici de la sessió.

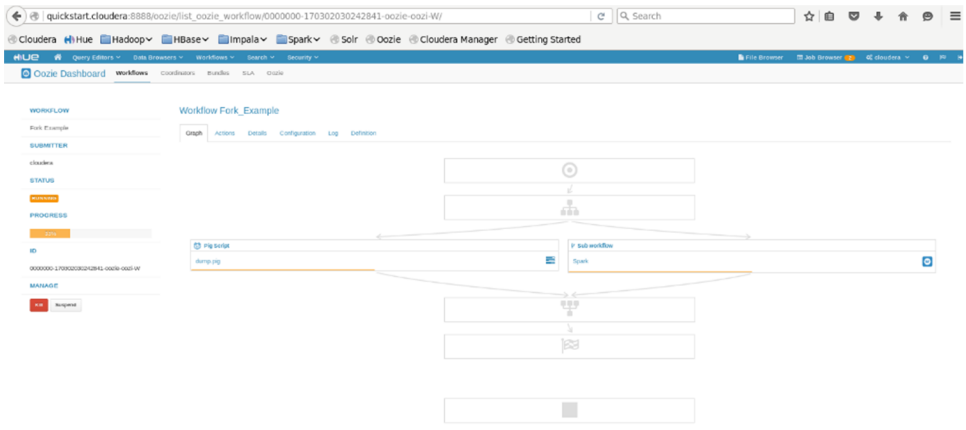
Sobre el navegador es podrà accedir a Hue (pel *bookmark* o http://quickstart.cloudera:8888/about/ amb usuari i contrasenya **cloudera/cloudera**), on primer verificarà i mostrarà la configuració i després es podran instal·lar els exemples que presenta l'entorn *Step 2* (Hue → *About*) i analitzar diferents casos d'usos (per a això, en el *home* de Hue (http://quickstart.cloudera:8888/home2) es veuran els diferents fitxers en els quals fent un doble clic es carregarà l'exemple i es podrà executar).

A les figures següents, es mostra el processament a través de Hive d'unes dades sobre professions/salaris, altres sobre adreces/correus electrònics de clients a través d'Impala, o s'executa un *workflow* amb Ozzie i Spark com a processament.

A la documentació es poden veure els detalls de cada aplicació i, a través de la interfície de Hue, veure les tasques i els *logs* o accedir a l'RM de Hadoop i veure l'estat de les tasques, com s'ha vist anteriorment.







Una segona manera de treballar és migrar a Cloudera Manager, que es pot posar en marxa des de l'escriptori executant l'enllaç a Cloudera Express; no obstant això, donarà un error si no es té la quantitat de memòria adequada, ja

que aquest necessita 8 GB i 2 VCPU. No obstant això, per a una prova es pot executar l'ordre des d'un terminal i forçar-lo perquè s'executi i la versió exprés (podria ser l'empresarial, però només per 60 dies de validesa):

```
sudo /home/cloudera/cloudera-manager --force --express
```

Després, accedint a l'adrça indicada en el terminal o la pestanya del navegador on indica Cloudera Manager, s'accedirà a la pantalla de gestió on es podran prendre decisions sobre cada servei, veure l'estat, arrencar/aturar CDH/CM, agregar nodes, i tot un conjunt d'opcions que permetran ràpidament gestionar tota la infraestructura Hadoop. No obstant això, cal tenir en compte que depèn molt de la RAM disponible i, per sota de 6 GB, molts serveis generen advertiments per poca disponibilitat de memòria; a més, el sistema es degrada de manera notable.

Per a una prova de funcionalitat s'ha seguit l'exemple donat a la plataforma (*tutorial 1*), on primer s'ha fet la ingestió de les dades i després s'han processat a través d'una consulta.

En primer lloc, s'han de traspassar les dades estructurades d'RDBMS a HDFS utilitzant Apache Sqoop (que permet carregar dades d'MySQL en HDFS preservant la seva estructura), per la qual cosa des d'un terminal de l'entorn executar:

```
sqoop import-all-tables \
    -m {{cluster_data.worker_node_hostname.length}} --connect \
    jdbc:mysql://{{cluster_data.manager_node_hostname}}:3306/retail_db \
    --username=retail_dba \
    --password=cloudera \
    --compression-codec=snappy \
    --as-parquetfile \
    --warehouse-dir=/user/hive/warehouse \
    --hive-import
```

Aquesta pròpia càrrega trigarà, ja que llança tasques MapReduce per obtenir les dades d'MySQL i desar-les en HDFS en format Apache Parquet; amb això, es creen les taules en Hive amb el mateix esquema d'MySQL.

Parquet és un format adequat per a l'anàlisi en Hadoop que, en lloc d'agrupar les dades en files, les agrupa en columnes, la qual cosa és més adequat per obtenir relacions entre els registres.

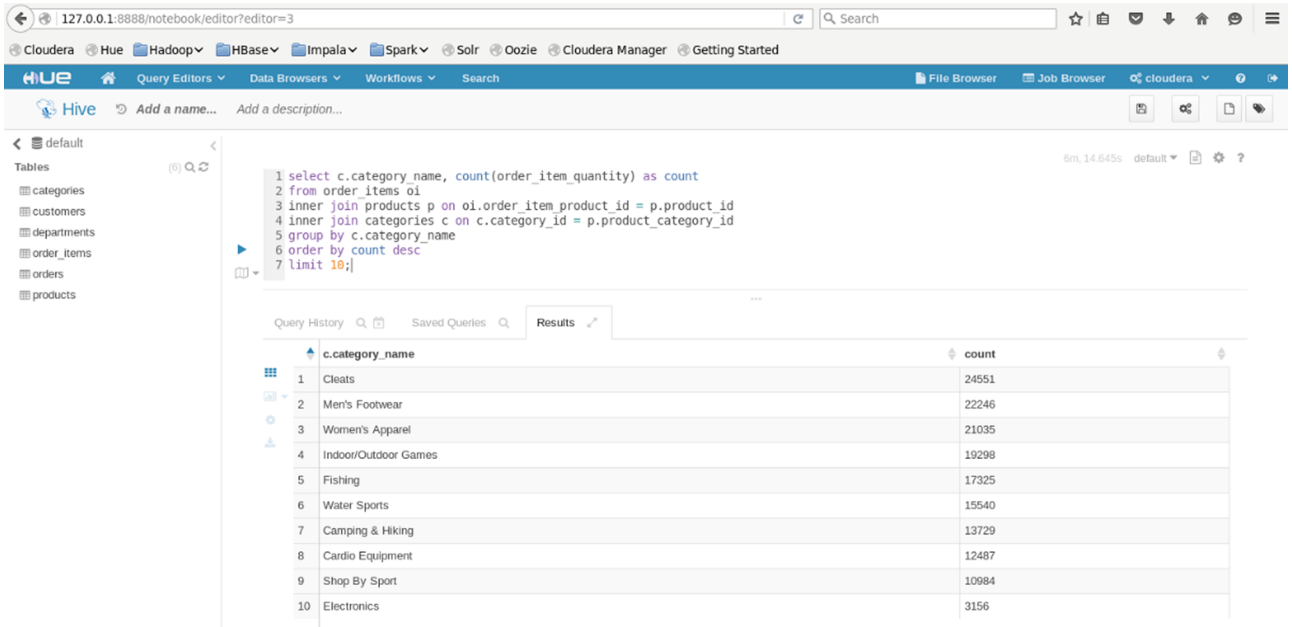
La comprovació que s'han creat els fitxers es podrà fer amb:

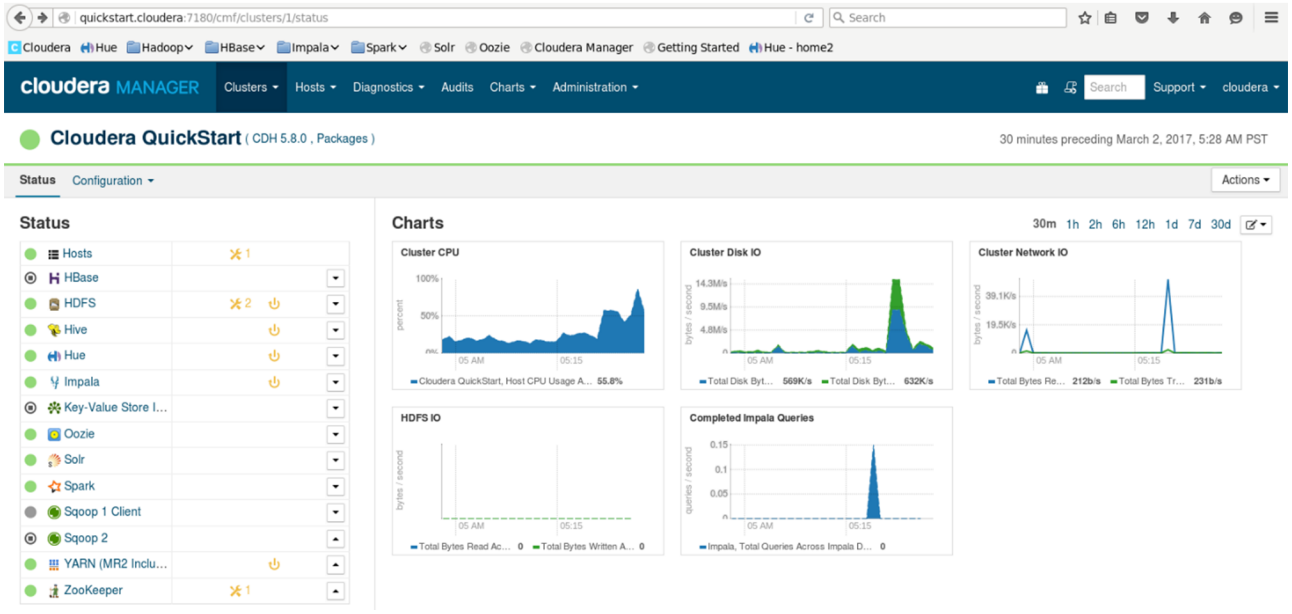
```
hadoop fs -ls /user/hive/warehouse/
hadoop fs -ls /user/hive/warehouse/categories/
```

Dins de Hue es pot anar a *Query Editors* → *Hive* (a la documentació es fa a través d'Impala, però en aquest cas, i per reduir la memòria utilitzada, s'ha fet a través de Hive), refrescar les bases de dades/taules disponibles i escriure i executar la consulta següent per determinar les deu categories més populars dels productes:

```
select c.category_name, count(order_item_quantity) as count
from order_items oi
inner join products p on oi.order_item_product_id = p.product_id
inner join categories c on c.category_id = p.product_category_id
group by c.category_name
order by count desc
limit 10;
```

La imatge, a continuació, mostra l'execució de la consulta a través de fer clic a ▶ (cal tenir en compte que la interfície en aquesta última versió ha canviat i és diferent a la interfície donada a la documentació) i la interfície de CM on es mostra l'estat de la infraestructura.





A la documentació, hi ha diferents consultes o altres exemples per analitzar la potencialitat de CDH + CM [Cma].

4.1.2. CDH sobre Docker

Una altra manera de provar CDH és a través de la màquina Docker. És important tenir en compte que, per a aquesta prova, s'ha de disposar d'un espai de memòria RAM lliure superior als 8 GB, ja que, sinó, el clúster CDH no arrencarà (vegeu les recomanacions al final de paràgraf).

Per a això, en primer lloc, s'ha de tenir instal·lat Docker (en una versió superior a l'1.11) i correctament configurat. En aquest cas, s'ha instal·lat Docker CE (*community edition*) sobre Ubuntu 16.04 seguint les indicacions del desenvolupador.

Després, s'ha accedit al projecte Clusterdock sobre l'Hub de Docker, que permet desplegar un clúster multinode sobre la mateixa Docker *host* (com requereix CDH per a proves de concepte).

A diferència d'altres eines com Docker Compose (dissenyat per manejar arquitectura amb microserveis), Clusterdock desplega múltiples contenidors Docker que actuaran com màquines tradicionals; per exemple, en una arquitectura Apache Hadoop de quatre nodes, aquest utilitzarà quatre contenidors. Com es va esmentar anteriorment, aquesta prova requereix una gran quantitat de memòria, i els desenvolupadors recomanen com a mínim 16 GB de RAM lliure per a dos nodes o 24 GB per a quatre nodes.

Clusterdock (en si mateix) està empaquetat en els contenidors juntament amb CDH per simplificar-ne l'execució i el desplegament; s'hi accedirà a través d'un *script* `clusterdock.sh` executant alguna de les funcions internes com

clusterdock_run o *clusterdock_ssh*. Per a això, s'utilitza l'ordre `source` per carregar l'*script* i després es podran executar les ordres esmentades; en aquest exemple, hi ha un clúster CDH de dos nodes:

```
source /dev/stdin <<< "$(curl -sL http://tiny.cloudera.com/clusterdock.sh)"
clusterdock_run ./bin/start_cluster cdh
```

Aquesta ordre descarregarà dos contenidors del Docker Hub que tenen la distribució completa de Cloudera Manager/CDH; es posarà en marxa el clúster CDH a través d'una xarxa *bridged*, i amb això es modificarà també el */etc/host* per adequar-lo als contenidors.

Una vegada validats els serveis CDH, es podrà accedir al clúster a través de Cloudera Manager (la URL i el port seran donats al final del desplegament, i l'usuari i la contrasenya per accedir a aquesta versió és **admin/admin**), o també s'hi pot accedir per SSH directament als nodes utilitzant la funció *clusterdock_ssh* amb el nom FDQ del node (per exemple, *node-1.cluster*).

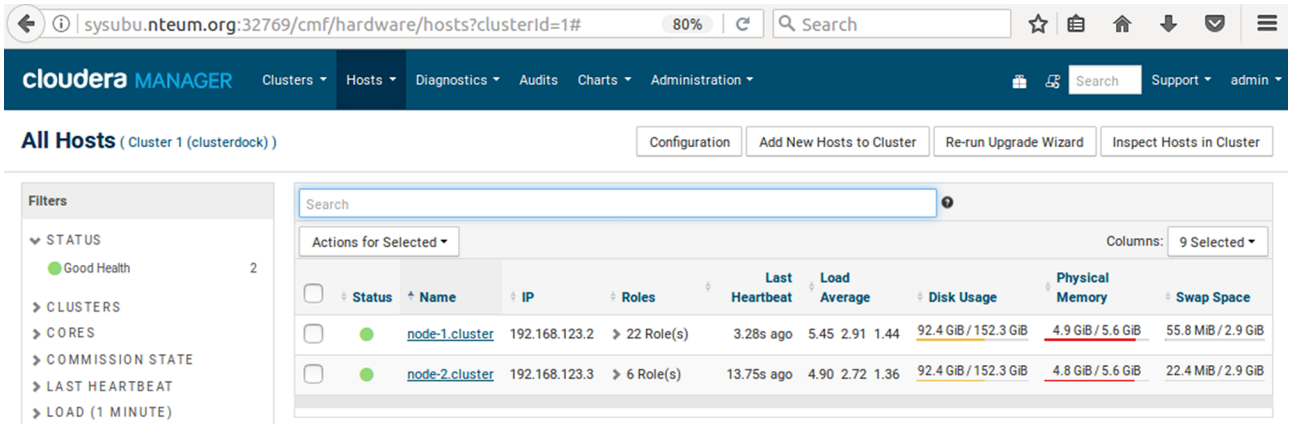
És molt important tenir en compte les restriccions de memòria per provar aquest apartat, ja que, si bé s'instal·larà tot, hi haurà dificultats quan inicií el clúster CDH i no serà possible arrencar-lo per aquest motiu (mostrarà un error com la figura següent).

```
INFO:clusterdock.topologies.cdh.actions:Detected Cloudera Manager server after 123.17 seconds.
INFO:clusterdock.topologies.cdh.actions:CM server is now accessible at http://sysubu.nteum.org:32769
INFO:clusterdock.topologies.cdh.cm:Detected CM API v13.
INFO:clusterdock.topologies.cdh.cm_utils:Updating database configurations...
INFO:clusterdock.topologies.cdh.cm:Updating NameNode references in Hive metastore...
INFO:clusterdock.topologies.cdh.actions:Once its service starts, Hue server will be accessible at http://sysubu.nteum.org:3
2768
INFO:clusterdock.topologies.cdh.actions:Deploying client configuration...
INFO:clusterdock.topologies.cdh.actions:Starting cluster...
Traceback (most recent call last):
  File "./bin/start_cluster", line 70, in <module>
    main()
  File "./bin/start_cluster", line 63, in main
    actions.start(args)
  File "/root/clusterdock/clusterdock/topologies/cdh/actions.py", line 154, in start
    raise Exception('Failed to start cluster.')
Exception: Failed to start cluster.
```

Si igualment es volen fer les proves amb una quantitat de memòria inferior a la suggerida, es pot posar en marxa la infraestructura sense arrencar el clúster i després fer-ho de manera controlada dins del CM.

```
clusterdock_run ./bin/start_cluster cdh -dont-start-cluster
```

La imatge següent mostra un desplegament realitzat amb Docker (sense iniciar els serveis de Hadoop) i dos nodes on es poden veure els requeriments de memòria a través de CM.



4.2. HortonWorks

La plataforma *open-source* de dades Hortonworks (HDP) està basada en Apache Hadoop, és escalable i permet emmagatzemar, processar i analitzar grans volums de dades de diferents fonts i formats d'una forma ràpida, fàcil i rendible. Aquesta integra diferents projectes d'Apache Software Foundation (ASF) que s'enfoquen en l'emmagatzematge i processament de dades massives com Hadoop, HDFS, YARN, Ambari, Falcon, Flume, HBase, Hive, Kafka, Knox, Oozie, Ranger, Slider, Spark, Sqoop, Storm, Tez i ZooKeeper.

L'empresa contribueix al seu desenvolupament de forma activa en algun d'aquests projectes i, a diferència d'altres plataformes que tenen com a base Apache Hadoop, Hortonworks aporta tot el codi produït a l'ASF, per la qual cosa HDP té llicencia Apache i és completament oberta. El model de negoci es basa en serveis, desplegament per experts i capacitació on la tecnologia és lliure i de codi obert [Hdp].

Per provar aquesta distribució, el desenvolupador proveeix un Sandbox, que és un entorn d'aprenentatge senzill i preconfigurat que conté els últims desenvolupaments de les eines relacionades amb Apache Big Data, que es pot executar al núvol (Azure) o en una màquina personal (VirtualBox, VMware, Docker), la qual cosa permet experimentar i explorar l'entorn HDP.

S'ha de tenir en compte que aquesta MV necessita 8 GB de RAM lliure; amb menys la màquina es degrada de manera notable.

Per fer una prova funcional, s'ha utilitzat una màquina amb 8 GB de RAM i s'ha deixat per a l'MV en Virtualbox 6 GB, i després d'una sèrie d'advertiments s'ha pogut arrencar.

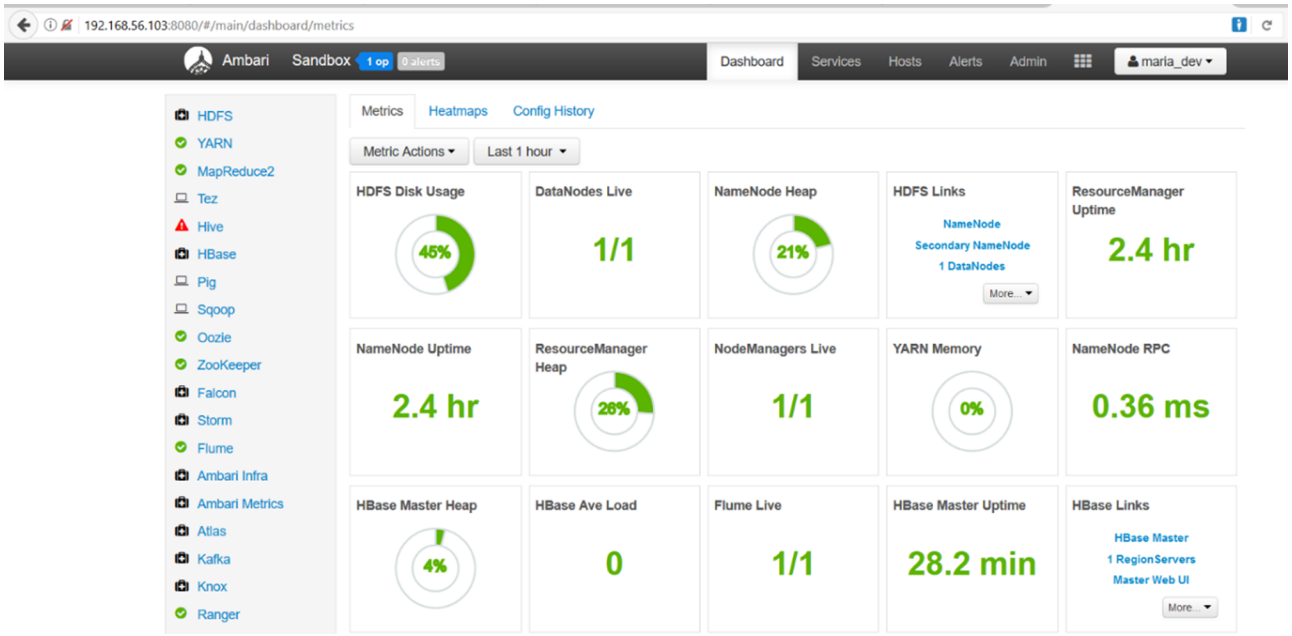
L'accés als serveis es pot fer a través de la *IP 127.0.0.1:port* sobre un navegador en el *host*, ja que l'MV té habilitat el *port-forwarding* per a la interfície NAT d'un gran conjunt de ports (vegeu a *MV → Network → Adaptador1 → Advanced → Port Forwarding*). També, si es prefereix, es pot agregar un segon adaptador configurat com *HostOnly* sobre Virtualbox (IP 192.168.56.100), ja que l'MV té

un dispositiu de xarxa configurat en 192.168.56.103 (s'ha de fer un `ifup` quan s'arrenqui la màquina, perquè aquest no s'aixeca durant el *boot*). Qualsevol dels dos mètodes pot ser utilitzat per accedir a la interfície gràfica a través d'un navegador.

Després d'arrencada l'MV, es pot accedir a través de la consola on l'usuari/contrasenya és **root/hadoop**, o s'hi pot accedir de moltes formes, tal com indica la guia d'Hortonworks.

Dins del sistema s'haurà d'executar `ip address` per veure els dispositius de xarxa (un d'aquests serà a NAT; per tant, tindrà IP = 10.0.2.15) i aixecar l'altre disponible amb `ifup<dispositivo>`. Després, des del *host* es podrà accedir a la URL http://192.168.56.103:8888 i, a través dels *quick links* o la URL http://192.168.56.103:8080, al *dashboard* (usuari/contrasenya **maria_dev/maria_dev** o consultar-ne d'altres a la guia abans esmentada). Si s'hi accedeix pel *port-forwarding*, reemplaçar la IP 192.168.56.103 per 127.0.0.1 a la URL.

La figura següent mostra el *dashboard* del Sandbox (amb molts dels serveis funcionant –i altres que necessiten atenció– i amb alguna alerta –per exemple, Hive– per la –reduïda– quantitat de memòria).



Per fer una prova de concepte sobre la infraestructura Hortonworks, s'ha utilitzat el Sandbox sobre una màquina de 16 GB de RAM, carregant un conjunt de dades dels desenvolupadors i analitzant la potencialitat de la distribució. Sobre l'MV s'han seguit els mateixos passos (amb una interfície *HostOnly*) abans

descrits per accedir a la interfície d'HDP amb un navegador sobre el *host*, però també s'hi pot accedir utilitzant la *IP 127.0.0.1:port* sobre el *host* utilitzant el *port-forwarding* que té habilitat l'MV sobre la interfície NAT.

Per a l'anàlisi s'han descarregat les dades *Geolocalization.zip* de la guia *Getting Started with HDP*, on s'utilitzen dades d'una empresa de transports que inclouen vehicles, dispositius i persones que es mouen a través de la geografia d'un país i on l'empresa vol vincular la informació d'ubicació amb les seves dades analítiques i analitzar el risc que pot representar per als treballadors diverses rutes/hores de conducció i vincular-ho a altres paràmetres.

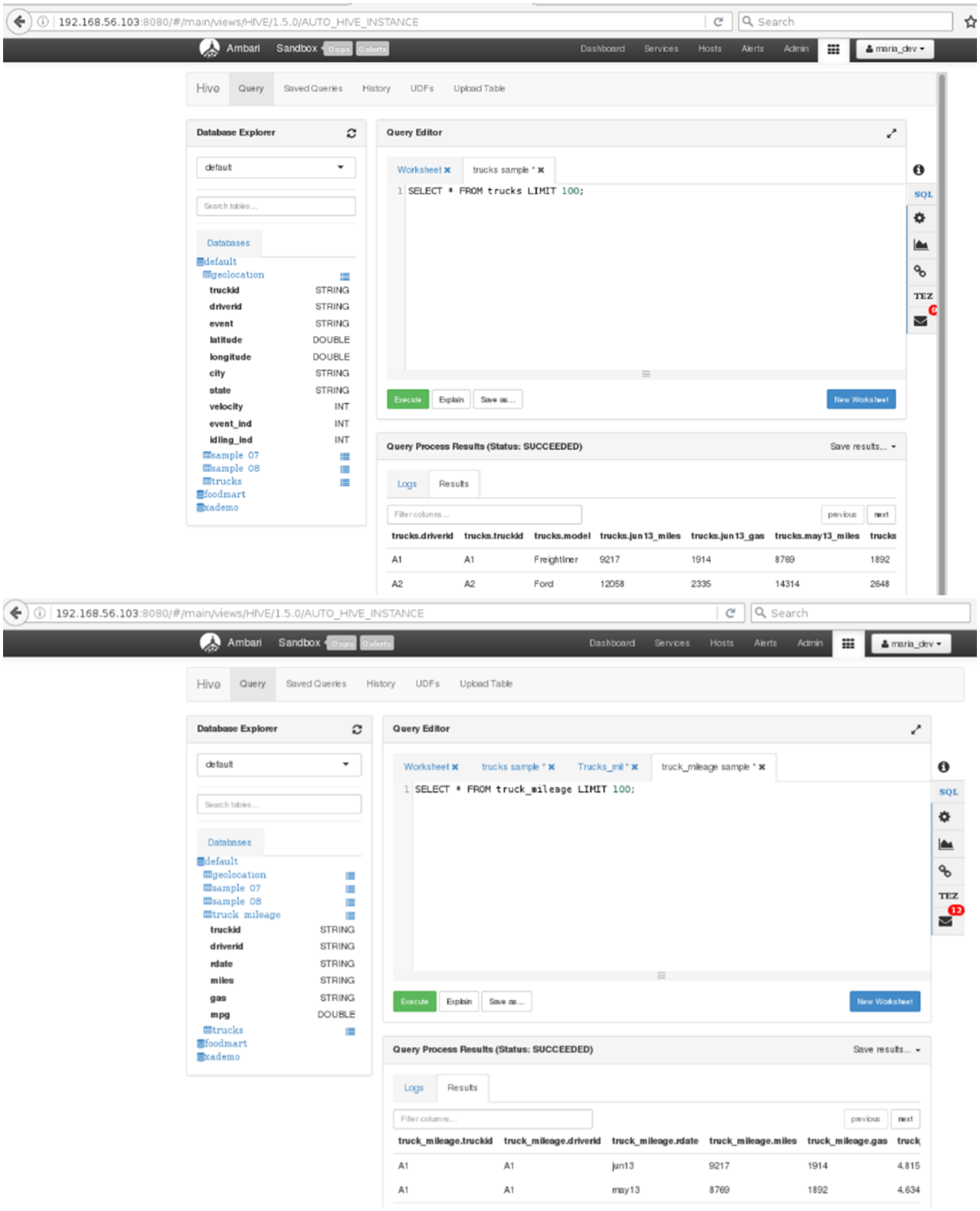
En aquest exemple només s'han carregat les dades en HDFS i, posteriorment, s'ha realitzat una anàlisi amb Hive, però en el tutorial esmentat s'explica com calcular el factor de risc dels conductors utilitzant Pig, una anàlisi amb Spark i visualització amb Zeppelin.

Per a aquest mínim exemple, s'han seguit els passos indicats a *Loading Sensor Data into HDFS* i posteriorment *Data manipulation with Hive*.

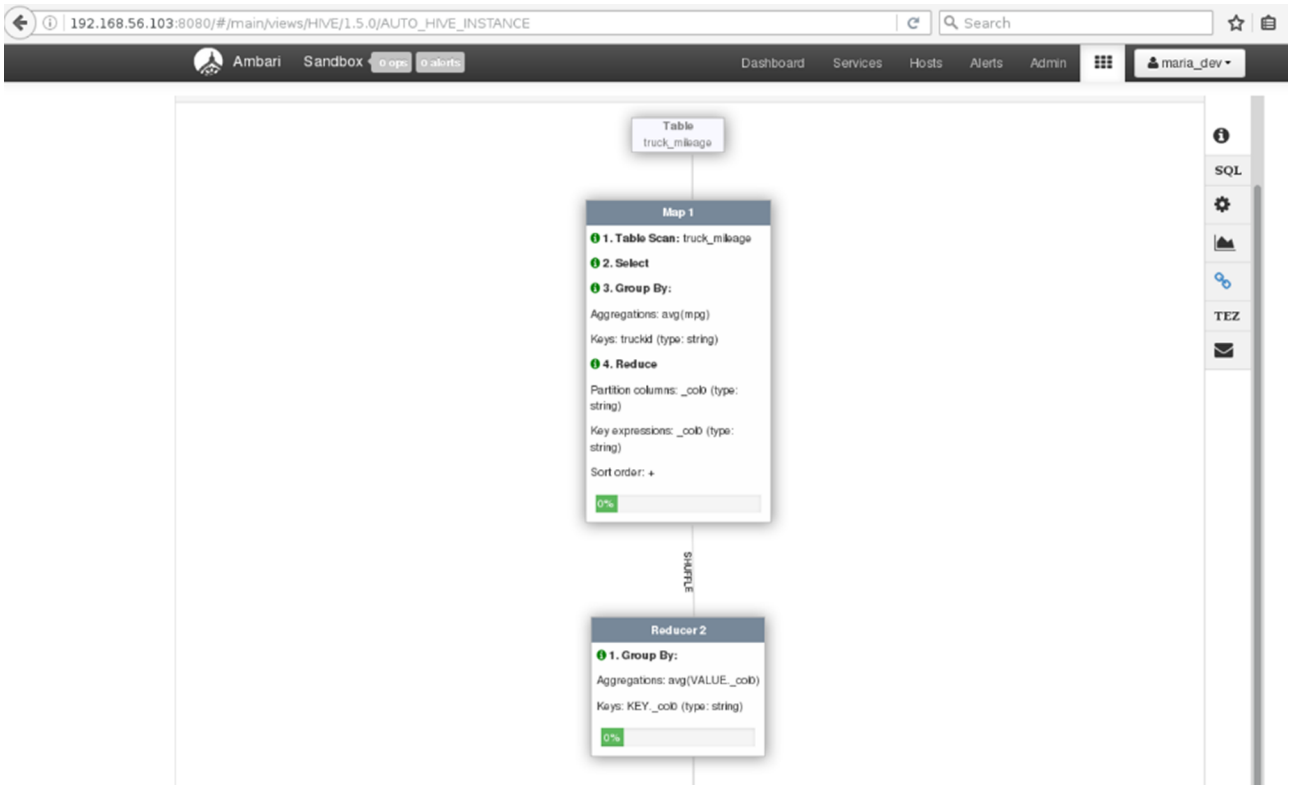
Com mostra la gràfica, a continuació s'ha executat una consulta sobre la taula *trucks*; amb això, s'ha mostrat el resultat dels 100 primers registres i després s'ha creat una nova taula:

```
CREATE TABLE truck_mileage STORED AS ORC AS SELECT truckid, driverid, rdate, miles, gas, miles /
  gas mpg FROM trucks LATERAL VIEW stack(....) dummyalias AS rdate, miles, gas;
```

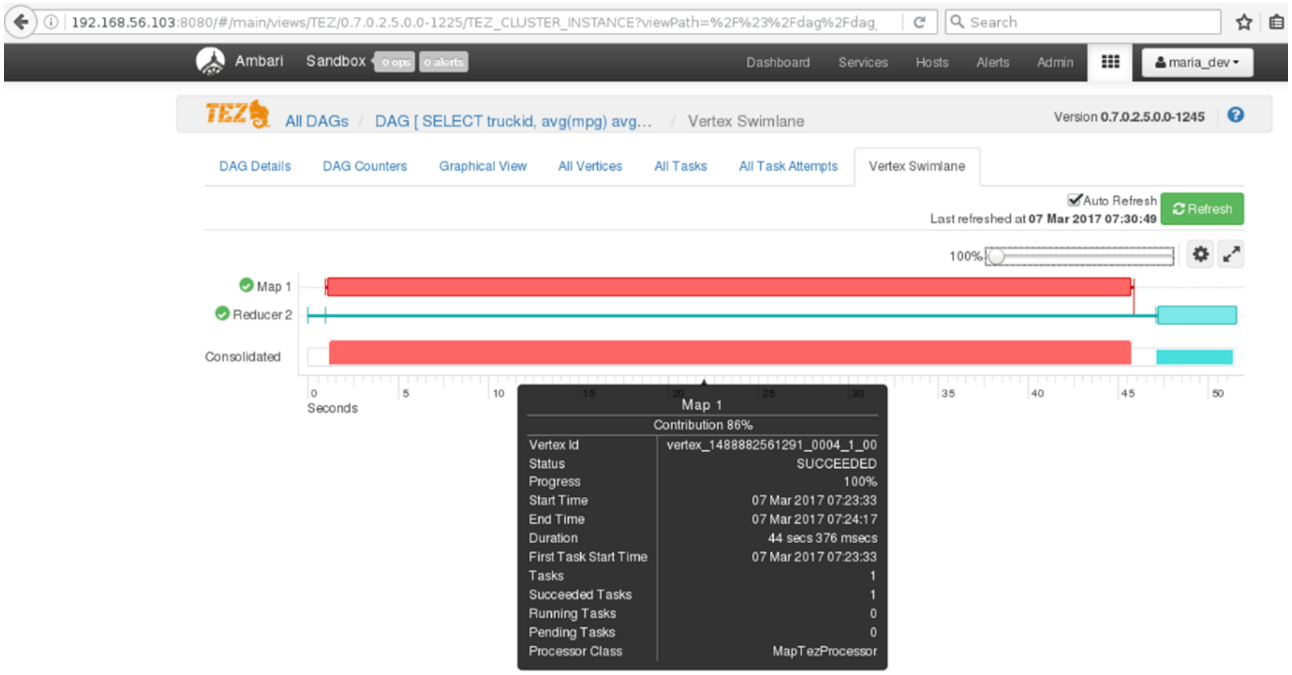
per a després generar una consulta per mostrar les milles/galó de combustible de cada camió, com es mostra a la segona imatge.



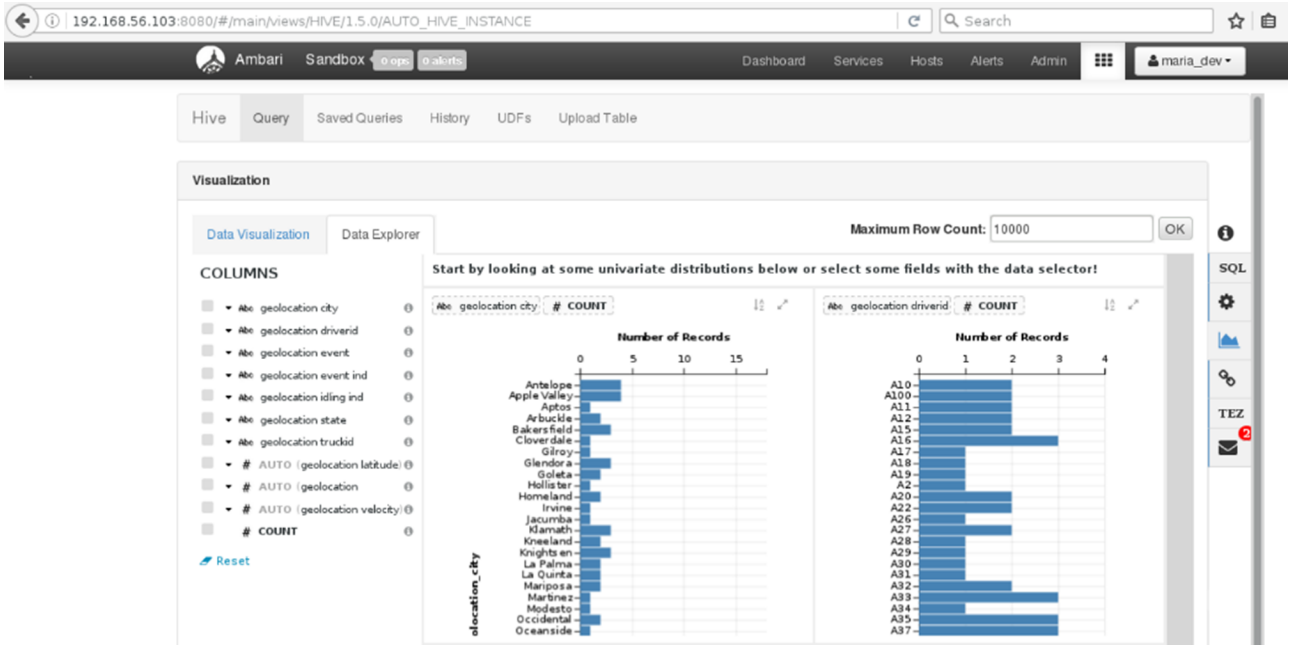
Com a part de l'experiència s'ha accedit a l'«*Explain*», que permet conèixer com es realitza la consulta i quines fases hi intervenen, visualitzant aquesta de manera gràfica, com mostra la figura següent.



A continuació, a través de Tez, s'han visualitzat els detalls de l'execució de la consulta anterior, i s'ha obtingut informació sobre cada fase de MapReduce amb informació detallada de temps i detalls sobre l'execució.



Finalment, s'ha accedit a la visualització de les dades mitjançant Hive on, a través del Data Explorer, s'ha pogut observar la distribució de les dades i accedir fàcilment a les dades processades per la *Query*, com es pot apreciar a la imatge següent.



Com es pot observar (i només s'ha vist un mínim exemple), la potencialitat d'HDP és molt gran, igual que les possibilitats que representa per al tractament de grans conjunts de dades. És per això que es recomana fer proves i experimentar-hi (sobre una màquina amb la memòria adequada), ja que és una plataforma molt completa i amb una funcionalitat avançada, a més de tenir una alta integració amb tot l'entorn Hadoop.

4.3. MapR (NA 2021: Where is MapR today)

MapR Converged Data Platform és una plataforma d'àmbit empresarial que permet aplicar els coneixements analítics a processos operatius en temps real per crear coneixement sobre grans conjunts de dades.

Segons la visió del desenvolupador, és una plataforma que permet analitzar la convergència de segments/productes històricament separats per generar nous valors de manera simple i fàcil amb una infraestructura de dades segura i oberta, que redueix dràsticament el TCO, i analitzar aplicacions basades en dades en temps real.

En el disseny, MapR ha incorporat els criteris més importants que s'han d'aplicar en un centre de dades de mida gran que utilitzen Hadoop/Spark, que han de tenir un funcionament 24/7 i amb diferents escenaris, com són l'anàlisi per lots o en temps real i la consulta interactiva. Això ha permès una evolució de l'arquitectura de la plataforma, que manté la compatibilitat binària amb el

sistema de fitxers distribuït Apache Hadoop (HDFS) per garantir la compatibilitat, però que, a més, inclou MapR-FS, que és un sistema de fitxers distribuït, amb notables prestacions, donada la seva capacitat d'accedir directament al maquinari d'emmagatzematge.

Des del punt de vista estructural, MapR es diferencia d'altres distribucions de Hadoop, que requereixen clústers separats per a múltiples aplicacions, ja que permet el processament de fitxers distribuïts, taules de base de dades i fluxos d'esdeveniments en una única capa unificada. Això facilita tenir aplicacions operatives (per exemple, HBase) i analítiques (per exemple, Apache Drill, Hive o Impala) en el mateix clúster, la qual cosa es tradueix en una reducció de costos significativa.

Des del punt de vista de la integració amb l'entorn Hadoop i els projectes de l'AFS, MapR contribueix a l'ecosistema (sobretot en el desenvolupament d'Apache Mahout i Apache Drill) i totes les aplicacions passen rigoroses proves d'integració/execució abans de l'adopció; les aportacions pròpies i llibreries també són publicades en obert a través de repositoris públics (GitHub i Maven) [Wmr].

MapR Converged Data Platform està disponible en dues edicions *Converged Community Edition*, per a la utilització lliure (amb restriccions especificades a l'*End User License Agreement* –EULA–), i la versió *Converged Enterprise Edition*, per a desenvolupaments de serveis crítics que requereixen continuïtat de negoci (vegeu les diferències).

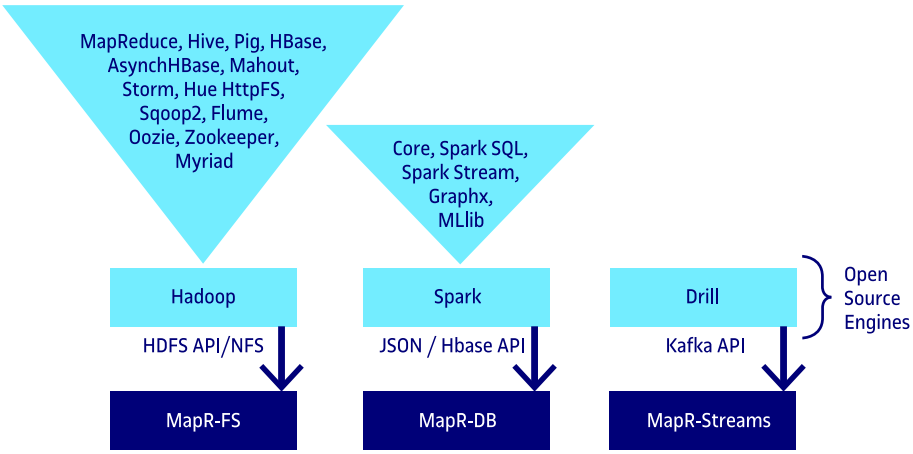
La instal·lació (**NA 2021:** també es pot fer des dels **repositoris d'HPE**, però només fins a la versió 6.1; la nova versió HPE DataFabric es pot provar per a entorns de desenvolupament, versió 6.2, però necessita el client de DataFabric instal·lat prèviament) no presenta dificultats i està automatitzada a través d'un *script* i posterior instal·lació a través d'una pàgina web (si bé cal disposar d'almenys dos nodes amb uns requeriments molt específics) [Mrd].

Per experimentar amb MapR, els desenvolupadors proveeixen un Sandbox per a VMware o Virtualbox, si bé cal disposar de 20 GB de disc disponibles, 4 *cores* de 64 bits –mínim a 1.3+ GHz amb suport VT-x– i 8 GB de RAM lliures. Els analistes en BI o desenvolupadors interessats en MR i Apache Drill poden utilitzar un altre Sandbox específicament dissenyat per a aquesta finalitat. MapR 5.2 Sandbox proveeix tutorials i aplicacions de demostració amb interfícies basades en web que permeten, ràpidament, conèixer i experimentar amb la plataforma, ja que ofereix una instància de MapR totalment funcional que s'executa sobre una MV. Hi ha una gran quantitat de cursos gratuïts del desenvolupador en MapR Academy [Mra], que permeten conèixer detalladament, a més de qüestions generals sobre les dades massives i el Hadoop, la plataforma, la seva administració, cadascun dels mòduls i com experimentar-hi.

La plataforma incorpora tres serveis –MapR-FS, MapR-DB i MapR Streams– que formen un nucli unificat en l'arquitectura MapR-CDP i són els que proveeixen d'alta disponibilitat, accés en temps real, seguretat unificada, *multi-tenancy*, espai de noms global, gestió i monitoratge.

Sobre aquests es poden instal·lar *open-source engines* (com Hadoop, Spark, Drill) o altres privades (com Vertica, Sap, MySQL).

MapR 5.2 Sandbox inclou Apache Hadoop (MapReduce, Hive, Pig, HBase, AsynchHBase, Mahout, Storm, HttpFS, Sqoop/Sqoop2, Flume, Oozie, ZooKeeper, Hue i Myriad), Apache Spark (Core, Spark SQL, GraphX, Spark Streaming, MLlib), MapR-DB (amb suport per a taules JSON i HBase), MapR Streams (amb suport per a l'API Apache Kafka) i MapR-FS. La figura següent mostra la imatge integrada de l'MV [Mrd].



MapR Sandbox es distribueix com *appliance* OVA, per la qual cosa s'ha d'importar des de Virtualbox; en aquest cas, el desenvolupador aconsella utilitzar el *port-forwarding* que s'ha fet sobre la interfície NAT cap al *host*, per la qual cosa, per accedir als serveis/interfície sobre el *host*, s'ha de fer http://127.0.0.1:8443 (o utilitzar *localhost* en lloc de 127.0.0.1).

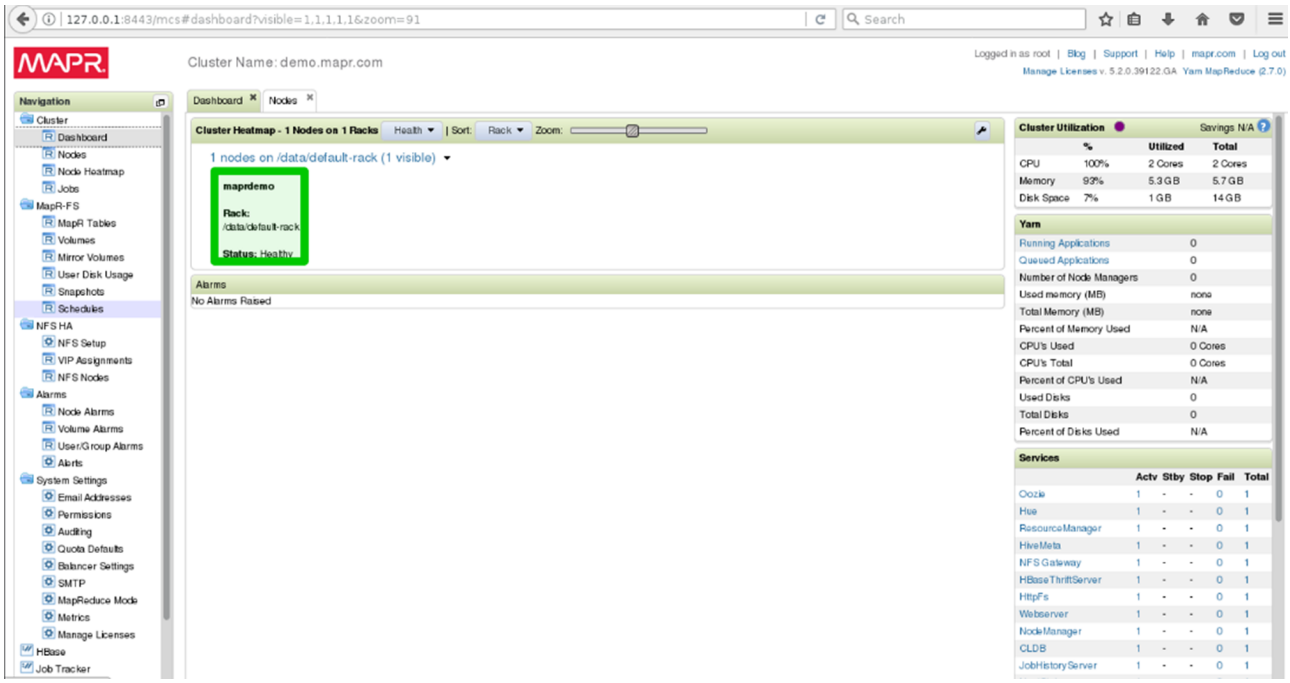
També s'hi pot accedir des d'un client de SSH executant (el servei SHH s'ha mapat sobre el port 2222 del *host* per no interferir amb el propi servei SSH d'aquest) amb usuari/contrasenya **mapr/mapr** (la contrasenya per a l'usuari *root* també és **mapr**).

```
ssh mapr@localhost -p 2222
```

Quan s'accedeix a la URL http://127.0.0.1:8443 del *host* (cal recordar que la interfície NAT de l'MV té un *port-forwarding* cap al *localhost* del *host*), es podrà accedir als dos rols definits:

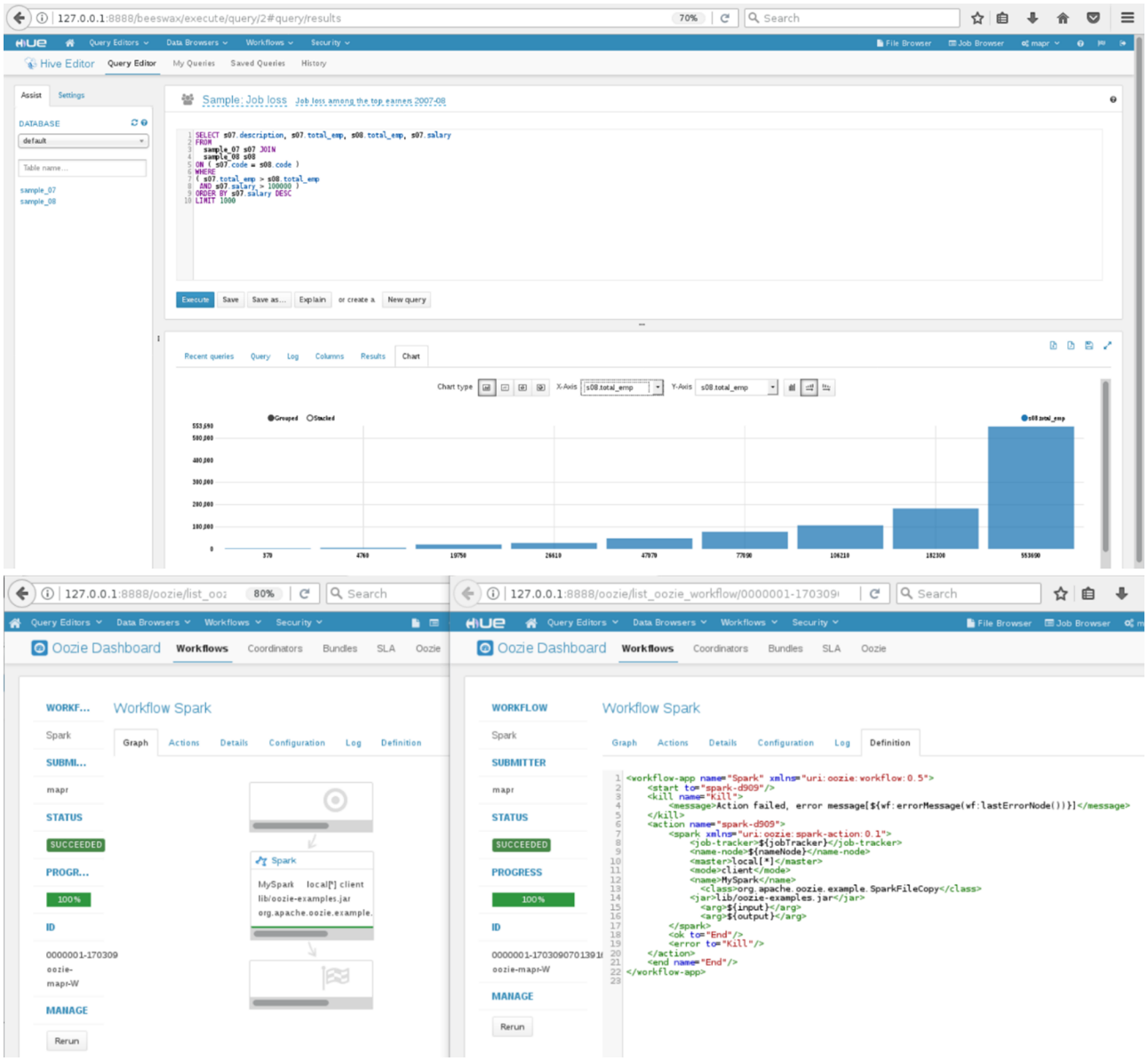
1) Desenvolupadors i analistes: orientat als desenvolupadors de Hadoop o analistes de dades que volen experimentar o treballar amb Hadoop i MapR utilitzant els exemples inclosos o carregant noves dades i processant-les. L'accés serà a Hue amb usuari/contrasenya **mapr**.

2) Administradors: perfil d'administrador d'infraestructura que permet accedir a MapR (MCS), la qual cosa permetrà configurar, monitorar i gestionar el clúster a través d'una interfície web (usuari/contrasenya **root/mapr**), com la mostrada a la figura següent.



És interessant, per guanyar coneixement a la plataforma i les eines, seguir els tutorials que ha desenvolupat MapR per conèixer l'entorn i les eines.

Per fer una prova funcional, i amb les dades ja integrades a MapR, s'ha executat una consulta SQL sobre Hive per obtenir la quantitat de tasques perdudes entre els que més guanyen i la seva representació gràfica, i l'execució d'un *workflow* de còpia d'un fitxer mitjançant Spark, com es pot apreciar a les figures següents.



En aquesta última, es pot veure la representació del *workflow* i la seva definició Oozie.

Com a prova funcional es compilarà i executarà el programa WordCount, però utilitzant Spark i seguint les línies de la guia *Getting Started with Spark on MapR Sandbox* i el codi de Github. Per a això, primer a l'MV MapR s'ha instal·lat, necessari per compilar el projecte, Apache Maven, que és una eina de gestió de projectes de programari i de comprensió que gestiona la compilació, elaboració d'informes i la documentació d'un projecte a partir d'un fitxer d'especificacions anomenat Project Object Model (POM) i que existeix en el directori de l'aplicació com a *pom.xml*.

Aquest fitxer és el nucli de la configuració del projecte a Maven, que conté la majoria de la informació requerida per construir el projecte, com en aquest cas. Per instal·lar Maven sobre CentOS (que és el SO de l'MV MapR), s'ha d'executar com a *root*:

```
cd /opt
wget http://www-eu.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz
tar xzf apache-maven-3.3.9-bin.tar.gz
ln -s apache-maven-3.3.9 maven
vi /etc/profile.d/maven.sh
   export M2_HOME=/opt/maven
   export PATH=${M2_HOME}/bin:${PATH}
source /etc/profile.d/maven.sh
mvn -version
   Apache Maven 3.3.9
   Maven home: /opt/maven
   Java version: 1.7.0_79, vendor: Oracle Corporation
   ...
```

Ara s'haurà de descarregar l'aplicació des de GitHub i compilar-la (això trigarà uns instants, ja que s'han de descarregar tots els paquets necessaris per compilar):

```
cd /user/user01
yum install git
git clone https://github.com/caroljmcdonald/sparkwordcountapp wordcount
cd wordcount
mvn package
cp /user/user01/wordcount/target/sparkwordcount-1.0.jar /user/user01
```

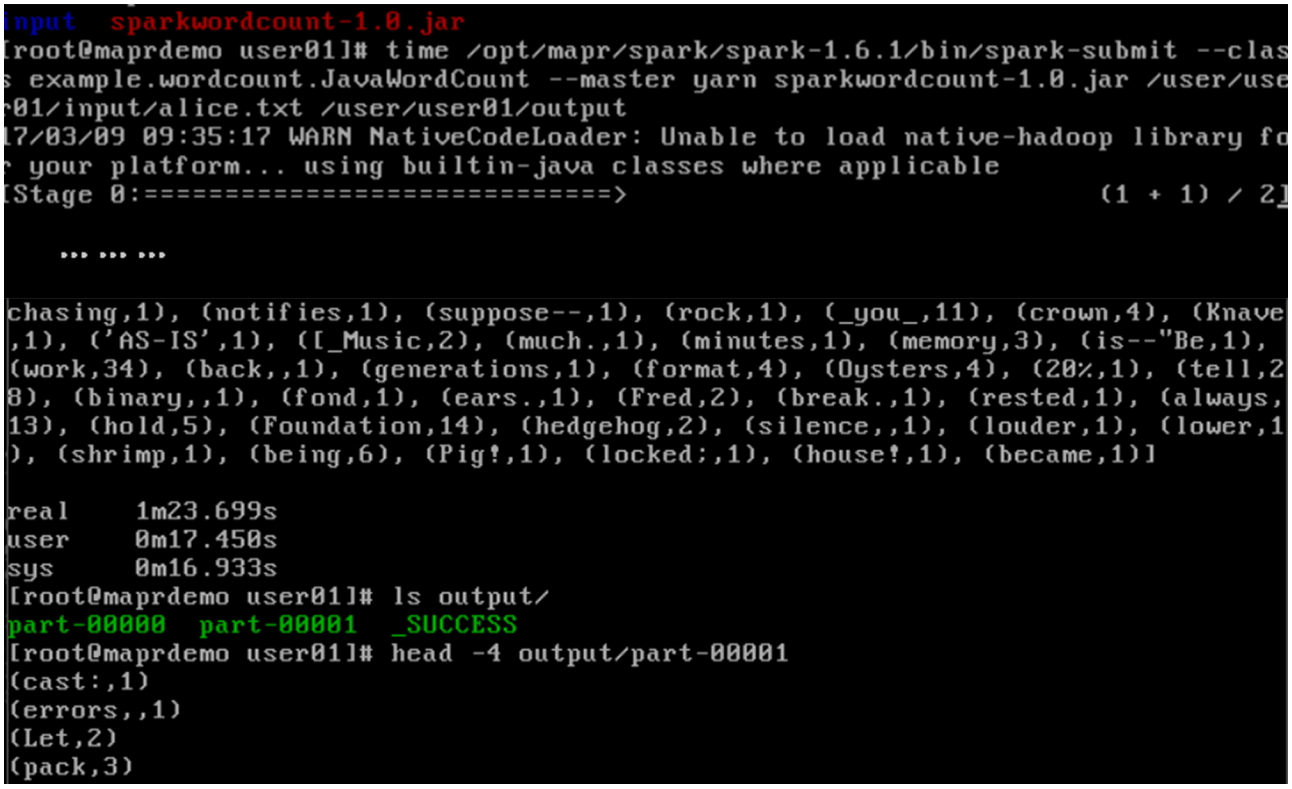
Per executar-lo, primer hem de tenir un text; s'utilitzarà *Alice in Wonderland*, del projecte Gutenberg:

```
mkdir /user/user01/input
cd /user/user01/input
wget -O alice.txt http://www.gutenberg.org/cache/epub/35688/pg35688.txt
```

Finalment, es podrà executar sobre Spark (verificar que la crida a `spark-submit` estigui tota en una línia):

```
cd /user/user01/
/opt/mapr/spark/spark-1.6.1/bin/spark-submit --class example.wordcount.JavaWordCount \
--master yarn sparkwordcount-1.0.jar /user/user01/input/alice.txt /user/user01/output
```

Les imatges a continuació mostren l'execució de l'ordre, la seva finalització i visualització dels resultats (s'ha anteposat l'ordre *time* per generar el temps gastat en la seva execució i tenir constància de la reducció significativa que hi ha quan es processa per Spark).



Tal com es va esmentar per a les plataformes anteriors, Mapr és una plataforma amb una gran funcionalitat i amb una visió una mica diferent a les altres plataformes, ja que incorpora MapR-FS com a reemplaçament a HDFS i, per tant, pot incloure Pig, Hive i Sqoop sense dependències de Java; proveeix accés a NFS des de qualsevol node, per la qual cosa es pot muntar MapR *file system* sobre NFS permetent amb això a les aplicacions accedir a dades de Hadoop de la forma tradicional; a més, està considerada per usuaris de renom una de les plataforma amb millors prestacions per al processament de les dades massives.

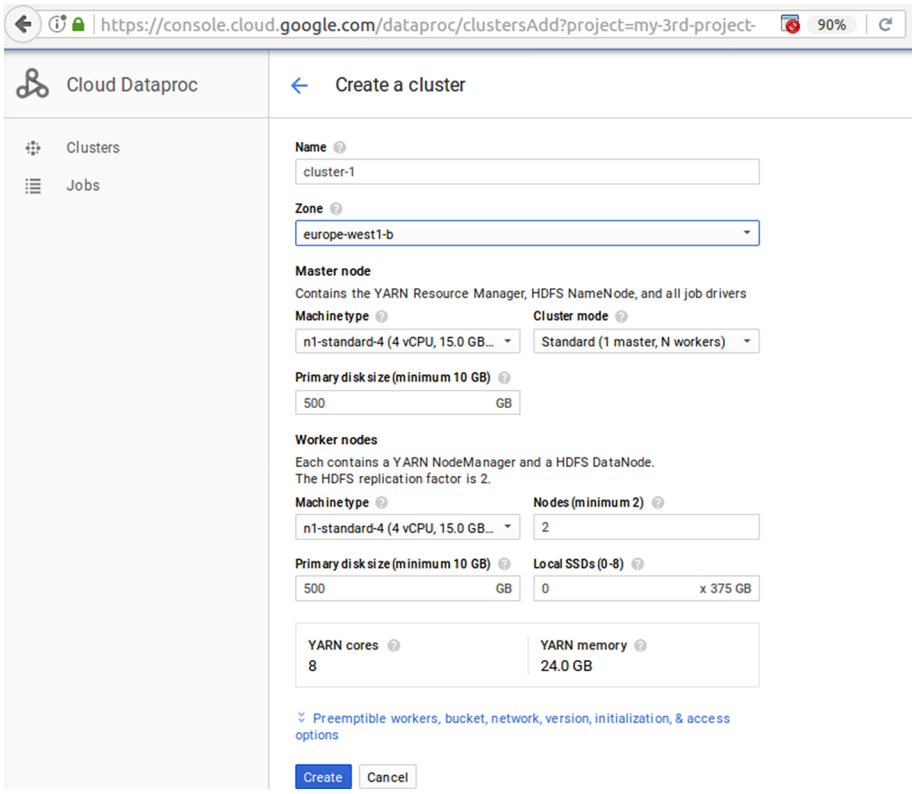
És interessant l'article [Gmr] de *Forbes* arran de la inversió de capital a MapR (2014) per part de Google; s'hi analitzen els objectius i mercats/negocis de cadascuna de les tres grans companyies tractades en aquest apartat que tenen com a base Hadoop.

5. Cas d'ús de *cloud* públic i Hadoop/Spark: Google Cloud DataProc

L'auge de les dades massives també ha arribat als *clouds* públics (per exemple, Amazon ERM (*Elastic MapReduce*), Azure HDInsight, Google Cloud Dataproc entre altres) i és una opció com les tractades en capítols anteriors, ja que proporciona un entorn pseudo-SaaS orientat a les dades massives on l'usuari només ha de desplegar les seves tasques, executar-les i obtenir-ne els resultats. Fent servir l'entorn de Google, es realitzarà una prova de concepte similar a les que es poden fer en altres proveïdors (òbviament més enllà de les interfícies i dels passos a seguir). Els passos indicats són els que s'han realitzat sobre la plataforma real, però no necessàriament els que es trobarà l'alumne, ja que aquestes plataforma estan en constant evolució i pot haver-hi diferències en la forma, les opcions, els menús, etc. (consulteu la documentació indicada a les referències i les seves actualitzacions).

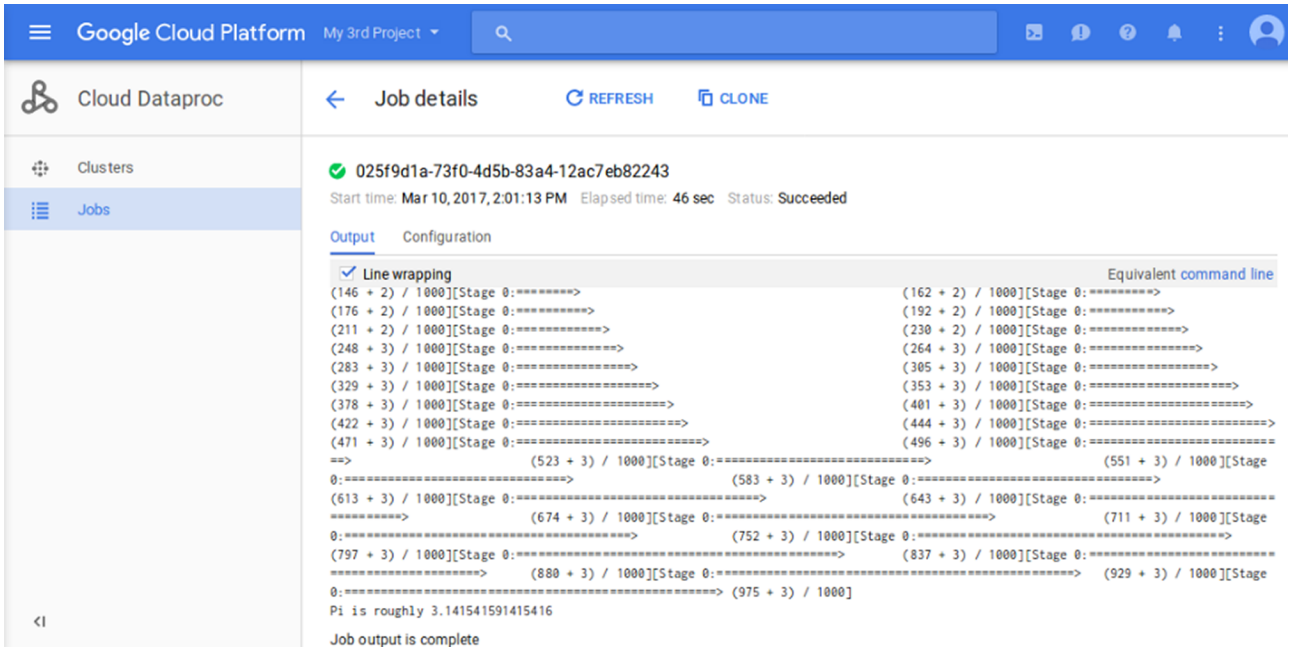
Cloud Dataproc és un servei d'Apache Spark i Apache Hadoop que permet aprofitar les eines de dades de codi obert per al processament per lots/consulta interactiva/*streaming*/*machine learning* a través de configuracions automàtiques on es pot ràpidament crear un clúster i enviar les tasques a executar només pagant pel seu ús.

La creació d'un clúster es pot fer des de la consola en el menú lateral→ *Big Data* → *DataProc* → *Cluster* → *Create a Cluster* d'acord amb les indicacions donades, o també a través de l'SDK sobre la màquina local i utilitzant l'API, o la CLI (`gcloud`). S'han de seleccionar les màquines –en tipus i disc– tant per al màster com per als *workers*, regió i altres paràmetres (com mostra la figura següent) i a continuació *Create*.

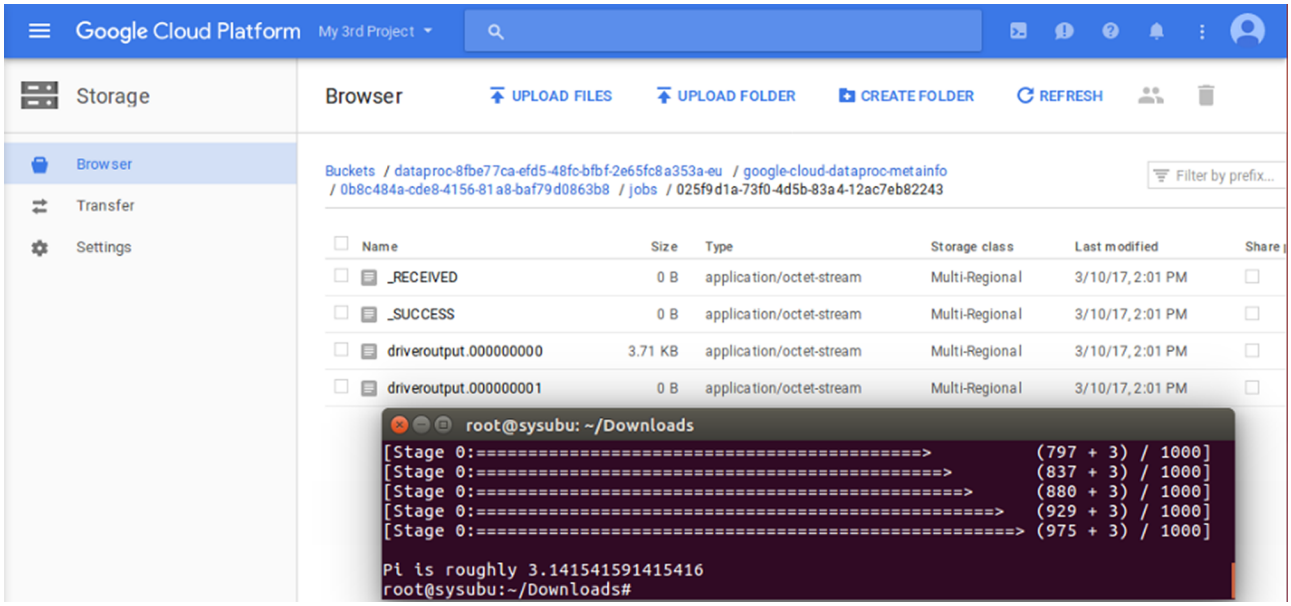


Per executar una tasca, per exemple Spark (en aquest cas serà un programa que calcula el valor de PI pel mètode de Monte Carlo), es pot fer en *Jobs* → *Submitjob* → indicar el nom del clúster (*clúster-1*) → seleccionar el tipus de tasca (*Spark*) → indicar el fitxer a executar en el camp de Jar (*file:///usr/lib/spark/examples/jars/spark-examples.jar*) → la classe en el camp *MainClass* (*org.apatxe.spark.examples.SparkPi*) → i l'argument (1000).

Després, es pot llançar l'execució de la tasca i, fent-hi clic a sobre, es veurà el resultat de l'execució (seleccionar *Line Wrapping* per evitar l'*scrolling*), com mostra la imatge a continuació.



No obstant això, es poden consultar els fitxers de sortida de l'execució de la tasca a *Storage* → seleccionant la tasca i el directori de sortida, com mostra la figura següent.



Considerant les diferents solucions que proposa Google com casos d'estudi, se n'ha seleccionat un de *Financial Services* → Monte Carlo *Methods* using Google Cloud Dataproc and Apache Spark. Google Cloud Dataproc i Apache Spark són solucions òptimes per a simulacions de Monte Carlo que s'utilitzen per respondre a qüestions en finances, enginyeria, ciència i matemàtiques que, d'una altra manera, no tindrien respostes. Un dels problemes que té aquest mètode és que cal treballar amb una gran quantitat de simulacions, per la qual cosa si el còmput es realitza sobre una arquitectura convencional, consumirà molt

temps i recursos, mentre que en Cloud Dataproc permet escalar i executar-lo d'acord amb les necessitats pagant per ús (per minut), amb els consegüents avantatges de reducció de temps, cost de la infraestructura i de l'administració.

Seguint la guia abans indicada, verificar que el clúster està creat i actiu; després, s'haurà d'accedir al menú lateral → *VM Instances* → *Node Màsterdel clúster* → botó SSH a la dreta (per obrir un terminal i connectar-nos per SSH a aquest node, que és des d'on es llançarà la tasca).

Sobre el terminal modificar el fitxer */etc/spark/conf/log4j.properties* i canviar *log4j.rootCategory* a *log4j.rootCategory=ERROR*, *console* per visualitzar els possibles errors en el terminal. Per a aquest exemple, s'utilitzarà Python (a través d'un intèrpret interactiu anomenat `pyspark`), que intenta predir com es podria fer una inversió. És a dir, a través de mostres aleatòries de resultats en un rang de condicions probables del mercat, la simulació de Monte Carlo pot respondre a preguntes sobre com una cartera podria evolucionar de mitjana. Per a això, executar (respectar la sagnia de la funció *grow*):

```
pyspark
>>> import random
>>> import time
>>> from operator import add

>>> def grow(seed):
        random.seed(seed)
        portfolio_value = INVESTMENT_INIT
        for i in range(TERM):
            growth = random.normalvariate(MK_AVG_RETURN, MK_STD_DEV)
            portfolio_value += portfolio_value * growth + INVESTMENT_ANN
        return portfolio_value
```

Després d'acabar d'introduir la funció, pressionar *<enter>* fins que surti novament el *prompt* de `pyspark` (>>>).

Aquest codi defineix una funció que modela el que podria succeir quan un inversor té un pla de pensions que inverteix en el mercat de valors i que, a més, aporta capital addicional cada any. La funció genera un rendiment aleatori de la inversió, com un percentatge durant cada any i durant un període específic.

És important tenir en compte que la funció pren un valor de llavor com a paràmetre, que utilitza per reiniciar el generador de nombres aleatoris, la qual cosa garanteix que la funció no rep la mateixa llista de nombres aleatoris cada vegada que s'executa. A més, es genera un creixement distribuït aleatòriament a través d'una distribució normal (per a una mitjana i la desviació estàndard especificades) i s'augmenta el valor de la cartera pel creixement (pot ser positiu o negatiu) i se li afegeix un valor que representa una inversió addicional.

El pas següent és crear diferents llavors per introduir a la funció (10.000 en aquest cas):

```
>>> seeds = sc.parallelize([time.time() + i for i in xrange(10000)])
```

Això genera un conjunt de dades optimitzades (RDD) per al processament paral·lel, basat en l'hora del sistema. Quan es crea aquest RDD, Spark divideix les dades en funció del nombre de *workers/cores* disponibles (en aquest cas vuit conjunts) per a les 10.000 dades utilitzades.

El pas següent és posar-les a la disposició de la funció *grow*:

```
>>> results = seeds.map(grow)
```

El mètode *map* passa cada llavor en l'RDD a la funció *grow* i afegeix cada resultat a un nou RDD, que s'emmagatzema en els resultats realitzant una transformació i no genera resultats fins que calguin.

A continuació, s'especifiquen els paràmetres de la funció:

```
>>> INVESTMENT_INIT = 100000  # valor inicial de la inversión
>>> INVESTMENT_ANN = 10000  # inversión adicional c/año
>>> TERM = 30  # período en años
>>> MK_AVG_RETURN = 0.11 # porcentaje medio
>>> MK_STD_DEV = 0.18  # desviación estándar
```

Ara s'ha d'executar el *reduce* per agregar els valors:

```
>>> sum = results.reduce(add)
```

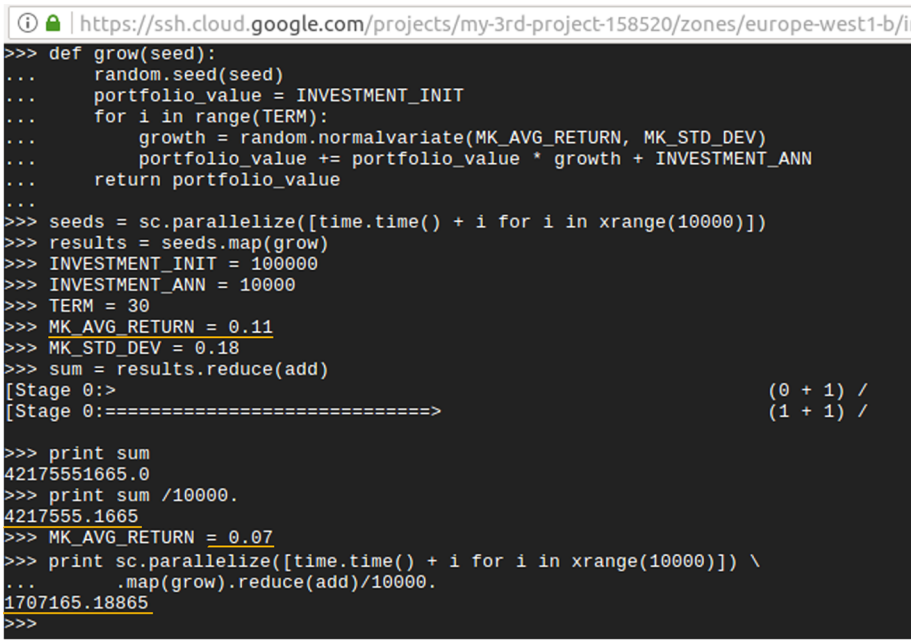
I finalment, imprimir el resultat (no us oblideu del caràcter '.' al final, ja que significa *float*):

```
>>> print sum / 10000.
```

Es podria canviar per exemple l'`MKT_AVG_RETURN` i tornar a executar:

```
>>> MKT_AVG_RETURN = 0.07
>>> print sc.parallelize([time.time() + i for i in xrange(10000)]) \
            .map(grow).reduce(add)/10000.
```

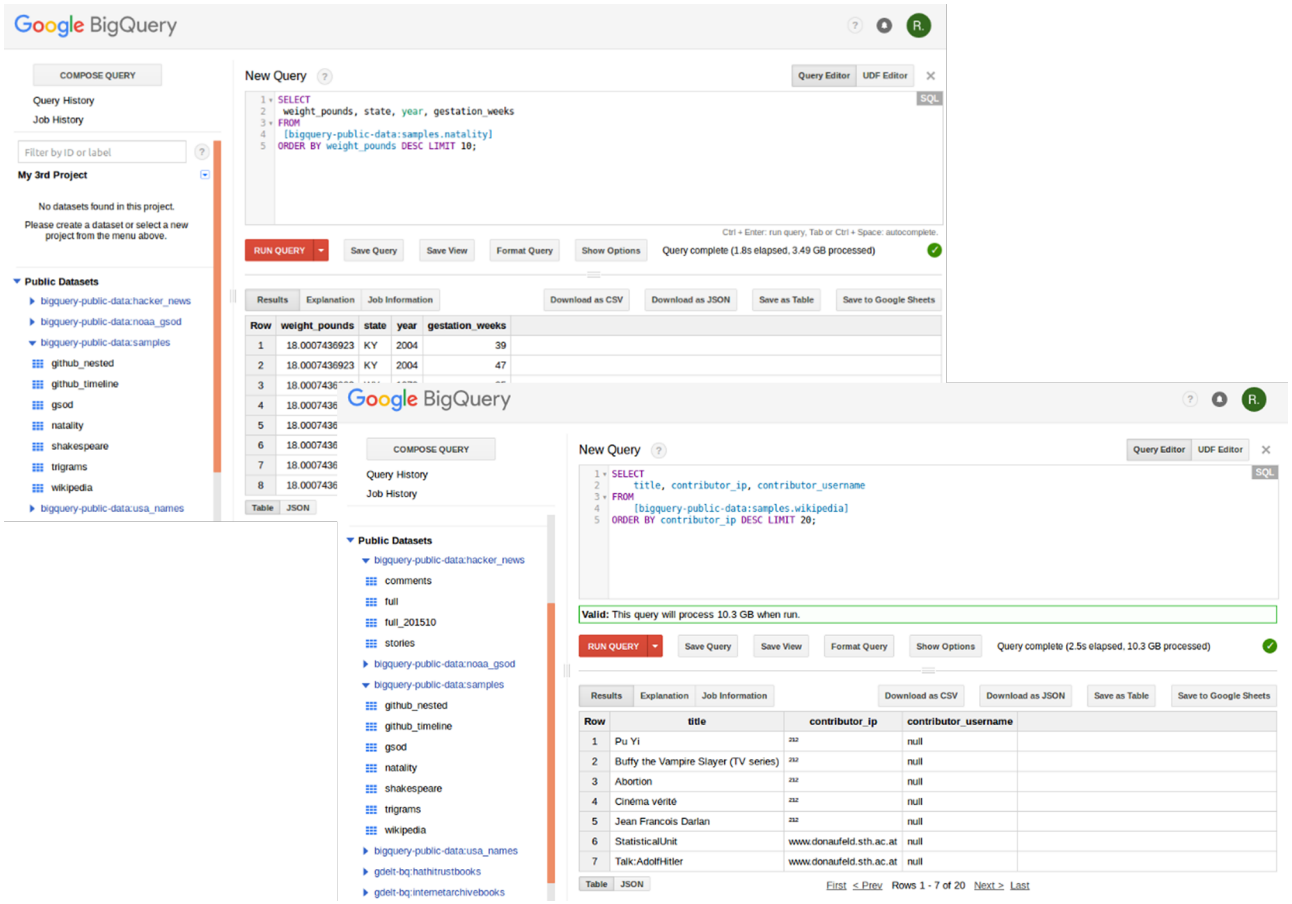
Feu *Ctrl + D* per sortir de l'intèrpret i recordeu que cal eliminar el clúster i els fitxers de l'*Storage* (a través de la consola o de la línia d'ordres) per no incórrer en despeses addicionals. La figura següent mostra l'execució en el terminal del node màster i els resultats obtinguts per als valors MK_AVG indicats.



Un altre exemple interessant és utilitzar dades públiques o pròpies per fer consultes a través de BigQuery. Aquesta aplicació permet emmagatzemar dades i fer consultes (quant a petabytes) totalment de forma administrada i de baix cost, a partir del paradigma *NoOps* (sense infraestructura per administrar i sense administrador de base de dades), la qual cosa permet a l'usuari centrar-se en analitzar dades per trobar valor agregat utilitzant SQL i amb un model de pagament per ús.

Per fer una prova s'ha accedit a través de la consola i el menú lateral → *BigQuery* → *Compose Query* i s'ha introduït la consulta SQL sobre dues bases de dades públiques (una de dades sobre la natalitat i una altra de *Hacker News*).

Les consultes i els resultats es mostren a continuació (és important verificar la titlla verda a la dreta abans d'executar la *query*, ja que si està en vermell indica algun error de la sintaxi).



Com es pot observar a la primera prova (natalitat), es va realitzar la consulta sobre 3,5 GB en 1,8 segons i a la segona (*Hacker News*) sobre 10,3 GB, i va trigar 2,5 segons, tot això sense preocupar-se de la infraestructura/base de dades/administració. És interessant seguir la documentació indicada importar/exportar les dades i qüestions respecte a la seguretat i privacitat d'aquestes.

Les eines de *big data* de Google (Cloud DataProc, BigQuery) mostren una plataforma amb una gran funcionalitat i uns connectors per vincular altres eines com, per exemple, BigTable –DB NoSQL–, i se suggereix consultar la documentació i els casos d'ús per guanyar coneixement i experimentar-hi.

A més de treballar amb més detall en les eines utilitzades, s'aconsella estendre l'anàlisi a altres eines no menys potents de les quals disposa Google Cloud per al tractament de les dades massives com, per exemple, Pub/Sub, Dataflow i ML Engine entre altres [Ghs].

6. Cas d'ús de *cloud* públic i *machine learning*: Azure ML

L'aprenentatge automàtic (ML) és una tècnica que s'aplica a la ciència de dades i que està molt lligat a la intel·ligència artificial (IA), que permet extreure informació i generalitzar comportaments a través d'aplicacions/dispositius capaces d'aprendre d'unes dades existents i, a través d'un procés d'inducció, obtenir uns resultats que permetin conèixer tendències/comportaments o, simplement, agrupar la informació per les seves característiques.

Aquestes tècniques, a vegades amb molta part d'estadística i molt lligades a la complexitat computacional d'un problema, permeten que aplicacions o dispositius puguin «aprendre» de les indicacions donades o de les dades introduïdes i fer suggeriments/classificar i/o ajudar a prendre decisions, atès que si el problema és de complexitat elevada, aquest el pot comparar amb les dades per a les quals ha estat entrenat i així oferir un camí possible que és probable que sigui millor, o almenys sigui una decisió més encertada que la que es podria prendre si no es disposa d'informació. Per a molts autors és l'evolució del mètode científic on algunes parts han estat automatitzades mitjançant procediments computacionals, estadístics i matemàtics, i amb una gran aportació de la IA.

Per exemple, un servei de correu electrònic que utilitza ML pot decidir si un correu és brossa o no en funció d'una gran quantitat de coneixement que té emmagatzemat sobre el que ha après de casos passats (entenent que qui envia el correu brossa ha canviat notablement les característiques de cada correu perquè no pugui ser detectat per les regles habituals, com origen, contingut, mida, etc.); un robot de neteja utilitza un algoritme d'ML per prendre la decisió quan ha acabat o no la neteja d'una habitació, o també s'utilitza ML en la detecció de seqüències i classificació d'ADN/RNA, en la interpretació de paraules/traducció d'idiomes o el reconeixement de l'escriptura i la seva traducció automàtica, entre molts altres exemples.

Amb dues grans divisions dels algoritmes d'ML és com es produeix l'entrenament/aprenentatge, en el qual es pot distingir el supervisat o el no supervisat (si bé hi ha altres classificacions com semisupervisat, per esforç, multitasca, etc.).

En el primer, s'entrena el sistema perquè pugui establir una relació entre les entrades i les sortides, és a dir, quin conjunt de dades formen part d'un cas o d'un altre, proporcionant-li informació sobre, per exemple, les categories d'una imatge (paisatge, edifici, persona) i el sistema podrà després classificar una imatge sobre la base del que ha après. En canvi, l'aprenentatge no supervisat no compta amb dades que defineixin quina informació satisfà els objectius

de l'aprenentatge o no, i ha de trobar patrons entre les seves característiques que permetin establir categories i poder fer conjunts de dades que tinguin els mateixos atributs.

Seguint l'exemple anterior de classificació d'imatges, no podria dir si és un edifici o no, però podria agrupar tots els edificis, les persones i els paisatges en grups. Aquesta informació després haurà de ser «interpretada» per indicar que un grup d'imatges de paisatges són paisatges.

Azure Machine Learning és un servei d'anàlisi predictiva en el *cloud* que permet crear i implementar, de manera simple i ràpida, models predictius i analitzar (a partir d'una llibreria d'algoritmes ja integrats) interactivament la seva execució i els seus resultats a través d'un servei totalment administrat que es pot usar per implementar els models predictius com a serveis web llestos per utilitzar.

El *workflow* que adopta Azure per a aquesta proposta és:

- recol·lecció de dades que poden ser d'Azure Storage (*blobs* i taules), HDinsight (Hadoop), Azure SQL (BD), Azure Data Lake (magatzem de dades massives),
- aplicació ML amb aprenentatge automàtic utilitzant models ja provats, adaptacions d'aquests o propis del dissenyador,
- servei web que pot rebre dades externes (per exemple, d'un BD),
- utilització de la informació generada per agregar «intel·ligència» o proporcionar informació.

Cal consultar la documentació per tenir un detall més ampli dels termes i conceptes implicats en Azure ML, així com del cicle de vida i les qüestions vinculades a la ciència de les dades [Aml].

Azure disposa d'un entorn anomenat Machine Learning Studio (**NA del 2021:** ML Studio només tindrà suport fins al 2024, i el desenvolupador suggereix migrar a Azure ML) on es poden crear de manera gràfica els models predictius per mitjà de la interconnexió de mòduls, ja siguin de la seva pròpia galeria (Cortana Intelligence) o amb modificacions sobre aquesta base o propis.

La utilització és molt senzilla, ja que presenta un entorn de treball (*canvas*) on s'arrossegaran els mòduls, els quals disposen de ports d'interconnexió d'entrada i sortida que permetran vincular les accions i indicar com les dades van fluint i són processades.

A la part esquerra, es trobaran una sèrie d'icones que indiquen projectes (col·leccions d'experiments, conjunts de dades, *notebooks* i altres recursos que representen un projecte individual), experiments (que s'han creat/executat i desat), serveis web (implementats a partir d'experiments), *notebooks*, conjunt de dades, models entrenats i configuració.

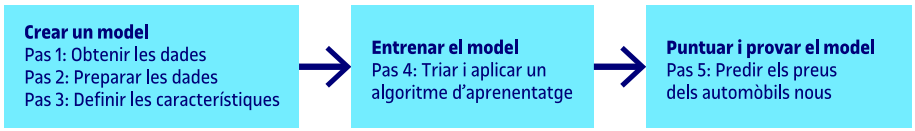
A continuació, es trobarà un menú de cerca de recursos per treballar, sobre la part dreta, un menú de propietats sobre cada recurs seleccionat en el *canvas* i, a la part inferior, els controls per crear i executar experiments/models.

La dinàmica és senzilla, es crea un experiment en *New* ✚ a la part inferior (se'n pot seleccionar un de la galeria o un en blanc), es busca el recurs a la caixa de cerca, s'arrossega al *canvas*, es modifiquen les propietats i s'interconnecta amb les seves fonts i destinacions (per a això, simplement seleccioneu el port d'origen i amb un clic del ratolí estireu la línia fins a la destinació).

Quan s'ha acabat es pot executar (*Run* ▶) per verificar la seva funcionalitat o visualitzar les dades a través del botó dret a cada mòdul (o altres tasques que permetin una font de dades, per exemple, descarregar-la en local).

Com a prova de concepte s'utilitzarà un cas d'ús d'ML basat en la guia *Estudi d'aprenentatge automàtic d'Azure* (**NA 2021**: per al nou entorn se suggereix seguir els exemples de la documentació), que permetrà aplicar els conceptes més importants i fer una prova funcional sobre la plataforma Azure.

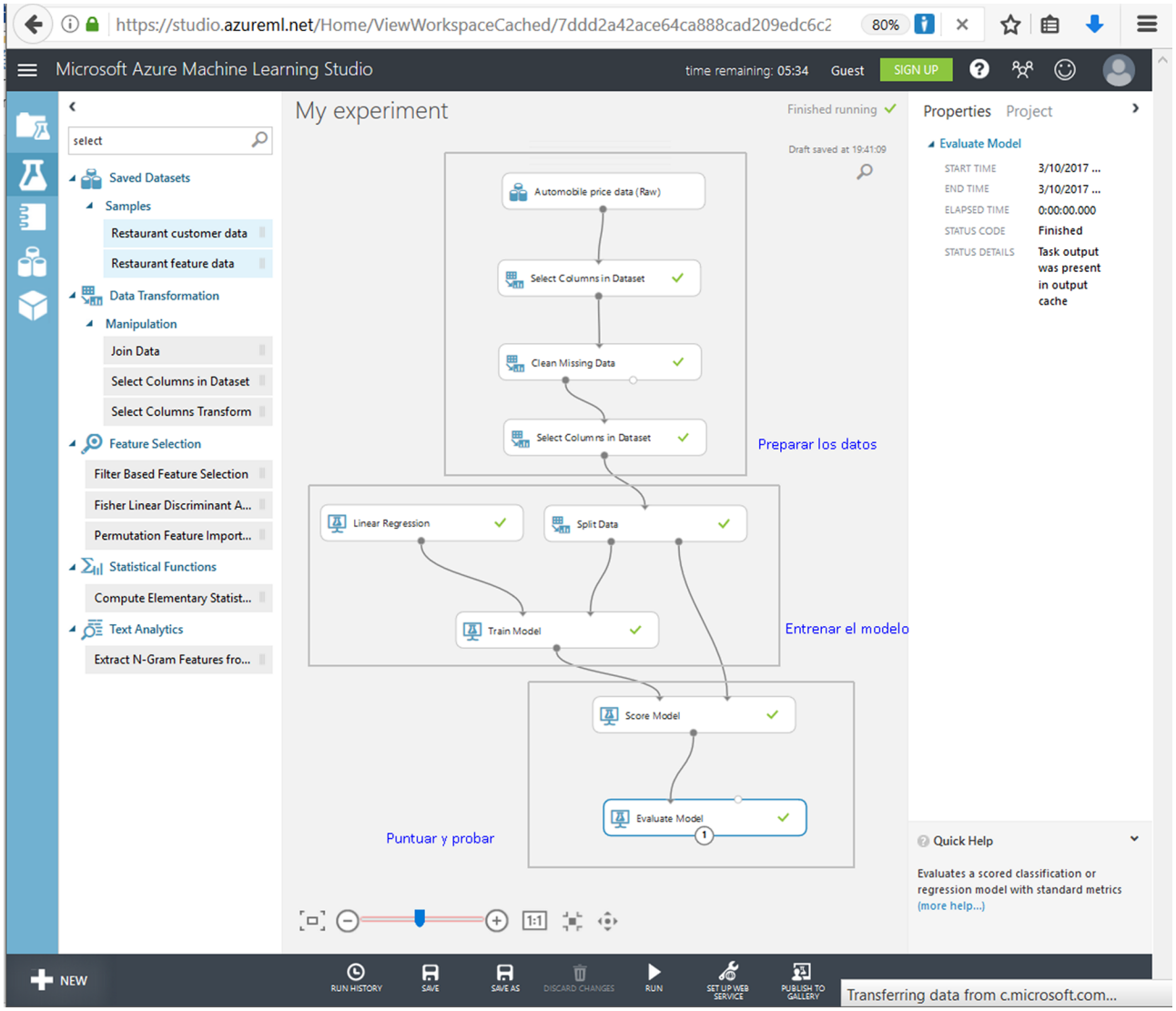
El *workflow* per a una aplicació d'aquest tipus és molt simple i queda representat per la figura següent:



Per a aquesta prova s'accedirà a la plataforma Azure ML Studio i se seleccionarà el tipus d'accés (convidat de 8 hores i sense registre o gratuït si es disposa d'un compte a Microsoft).

Sobre aquesta plataforma es treballarà amb un conjunt de dades que ja disposa, que inclou dades d'automòbils de diferents fabricadors, models, especificacions tècniques i preu.

El *workflow* acabat es mostra a la figura següent i s'aniran referint a cadascuna de les caixes indicades anteriorment.



Per iniciar la tasca:

1) A la part inferior feu *New* (✚) i seleccioneu la galeria *Blank experiment*, que mostrarà un esquelet per inserir els mòduls; canvieu, a la part superior, el nom per un altre de més significatiu (seleccionar i canviar).

2) A la caixa de cerca (pot estar plegada) s'introduirà *automobile* per seleccionar les dades i de la llista arrossegueu el conjunt de dades **Automobile price data (Raw)** al *canvas*. Amb el botó dret sobre el punt de sortida, podrem visualitzar aquest conjunt de dades, salvar una còpia en local o altres operacions. Com es pot veure, des de la visualització hi ha dades incompletes, per la qual cosa s'hauran de «netejar» aquestes files/columnes.

3) A la caixa de cerca s'introdueix *select* i de la llista *Manipulation → Select columndata set* s'arrossega a sota el mòdul anterior i s'interconnecten com a la figura anterior. Per seleccionar la «neteja» es fa clic al mòdul agregat i a *Properties → Launch column selector → With rules → All Columns → Exclude → Column Names → normalized-losses →* clic a la titlla inferior per acceptar.

4) Repetiu el procediment agregant el mòdul *Clean Missing Data → Properties → Clean Mode → Remove entire row*. Des del botó *Run* inferior (▶) executeu el *workflow* definit, que mostrarà una titlla verda al costat de cada mòdul quan s'hagi executat correctament.

En ML es denomina «característiques» a les propietats individuals i que es poden mesurar pel seu interès; en aquest conjunt de dades, cada fila representa un automòbil i cada columna, una característica d'aquest automòbil. Es necessita experiència i coneixement per decidir quines característiques analitzar o treure (per exemple, si dues d'aquestes estan fortament correlacionades, se'n pot treure una sense que afecti la predicció); no obstant això, en aquest cas, els desenvolupadors suggereixen *make*, *bodi-style*, *wheel-base*, *engine-size*, *horsepower*, *peak-rpm*, *highway-mpg*, *Price*, però podrien ser unes altres a voluntat (i coneixement) de l'usuari que fa l'anàlisi.

5) Per a això, agregueu un altre mòdul *Select Columns in Dataset → Properties → Launch column selector → With rules → No Columns → Include → Column Names →* i agregueu l'indicat anteriorment i accepteu. Després connecteu la sortida esquerra de *Clean* amb l'entrada d'aquest mòdul (vegeu figura anterior) i amb això finalitzarà la preparació de les dades (primer rectangle).

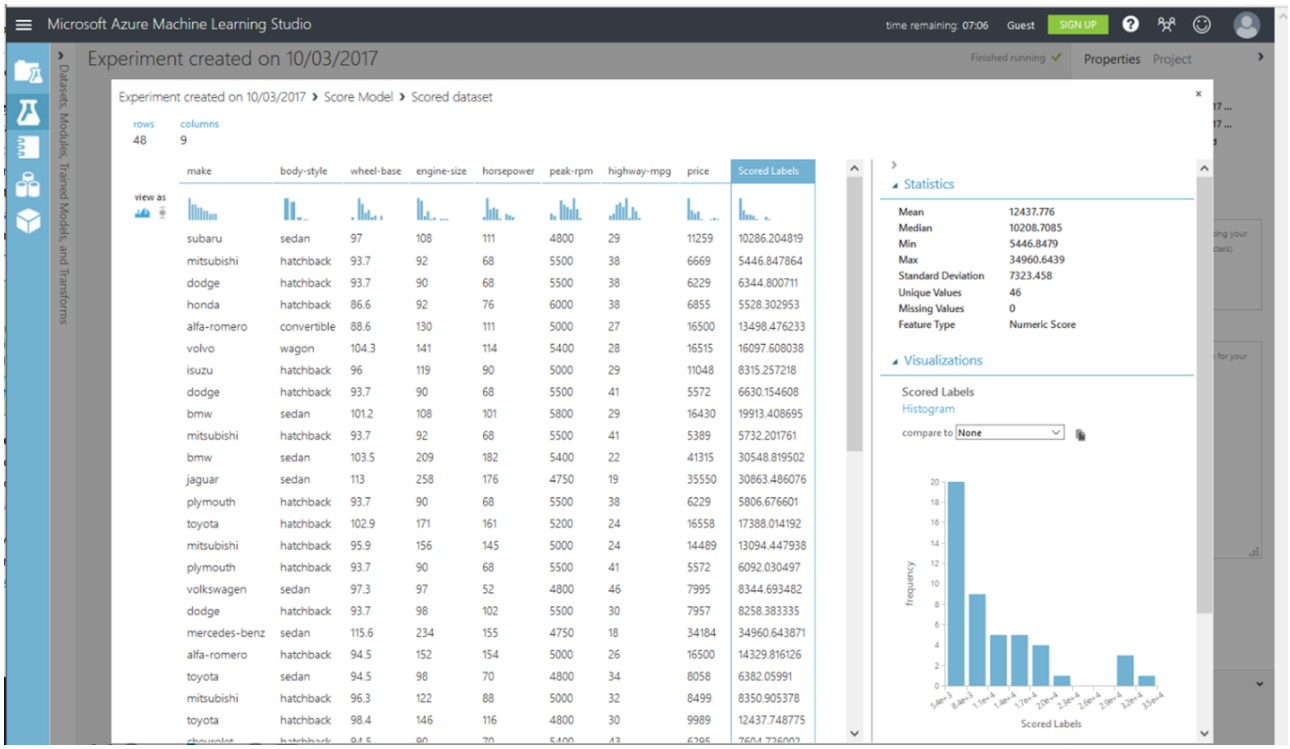
6) Ara s'han de seleccionar les dades per entrenar el model i les dades per predir-lo; per a això, s'agregarà un mòdul de *Split Data* i es connectarà a l'anterior indicant a *Properties → Fraction of rows in the first output dataset* un valor de 0,75, que indica que el 75 % de les dades seran utilitzades per a l'entrenament i el 25 % restant per a la predicció. Executeu novament el *workflow* per verificar que tot funciona correctament (titlla verda a cada mòdul).

En aquest exemple, utilitzarem l'aprenentatge supervisat per predir el preu a partir d'unes característiques; com algoritmes bàsics d'ML es poden utilitzar els de classificació o de regressió, on el primer permet predir una resposta a partir d'un conjunt definit de categories, com el color (vermell, blau o verd), mentre que la regressió s'usa per predir un nombre. Com que es vol predir el preu, s'utilitzarà un model de regressió lineal, que analitzarà les dades i cercarà, a la fase d'entrenament, correlacions entre les característiques d'un automòbil i el seu preu per després donar-li, a la fase de predicció, un conjunt de característiques d'automòbils i predir-ne el preu.

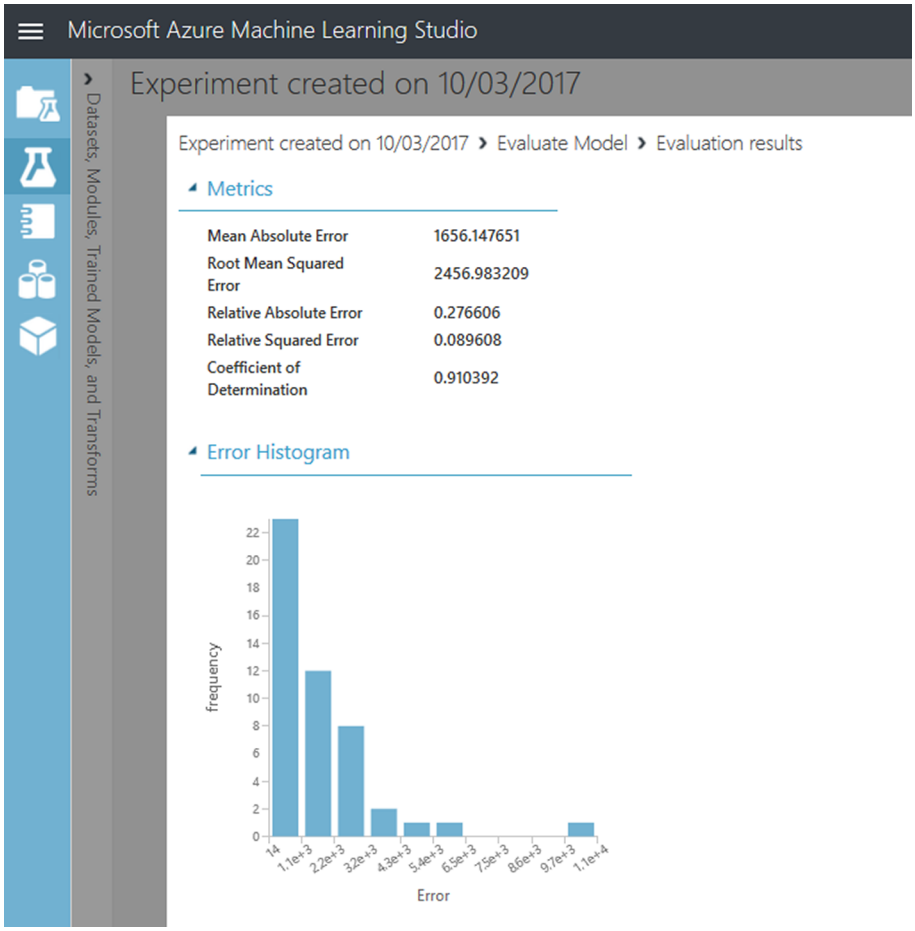
7) Cerqueu/agregueu el mòdul de *Linear Regresion*, i cerqueu/agregueu el mòdul de *Train Model* i interconnecteu-los, i també a *Split Data*, com es mostra a la figura anterior. A *Train Model → Properties → Launch column selector →*

seleccioneu la columna *Price* i moveu-la cap a la dreta. Finalment, executeu l'experiment i verifiqueu que tot és correcte. A partir d'aquest moment, es disposa d'un model de regressió entrenat que es pot utilitzar per realitzar prediccions de preus en funció de les seves característiques.

8) Ara s'ha de puntuar i verificar com funciona el model amb el 25 % de les dades restants; per a això, busqueu/agregueu el mòdul *Score Model* i interconnecteu-lo amb *Train Model* i *Split Data*, com s'indica a la figura anterior, executeu el model i observeu la sortida (visualitzar), que mostrarà els valors donats per la predicció, que es poden comparar amb els reals ja coneguts per analitzar com ha anat aquesta. Vegeu les últimes dues columnes de la figura següent (es poden seleccionar els gràfics corresponents per veure si l'histograma té una forma similar).



Finalment, per provar la qualitat dels resultats, busqueu/agregueu/interconnecteu el mòdul *Evaluate Model* i executeu/visualitzeu els resultats, on es mostraran una sèrie d'estadístiques que donaran informació sobre l'adequació del model: desviació mitjana de l'error o MAE (la mitjana d'errors absoluts, diferencia entre el valor de predicció i el valor real), arrel quadrada d'errors o RMSE, l'error absolut relatiu o RAE, l'error al quadrat relatiu o RSE, i el coeficient de determinació o CD (valor R quadrat és una mètrica estadística que indica com de bé s'ajusta un model a les dades; com més a prop és el valor d'un, millors són les prediccions).

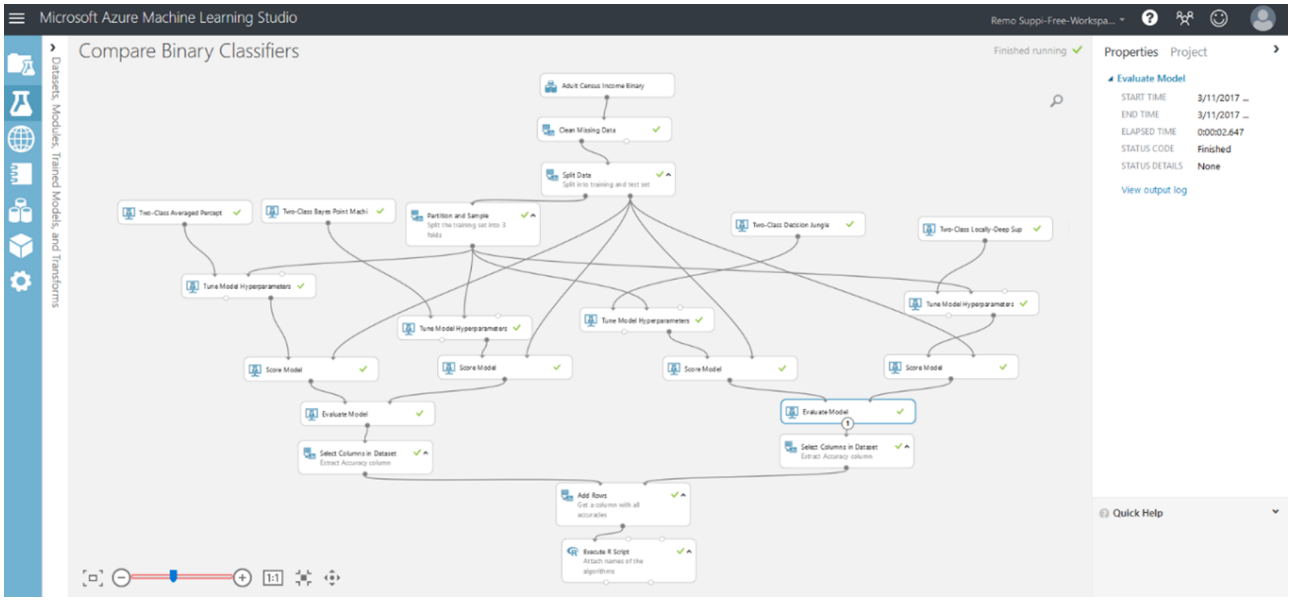


9) Com a passos següents es podrien modificar els paràmetres del model de regressió linear, canviar els paràmetres de *Split Data* o agregar un altre model addicional, i com que el mòdul *Evaluate Model* disposa de dues entrades, comparar els resultats de la predicció de dos models diferents o, finalment, quan es tingui ajustat, implementar-lo com un servei web predictiu.

10) Per veure un exemple de com comparar diversos models en un únic experiment, consulteu *Compari Regressors* a la galeria de Cortana Intelligence.

Cal no oblidar salvar l'experiment i consultar la documentació indicada per a més detall i/o aspectes addicionals indicats pel desenvolupador.

11) Més complex per comparar diferents algoritmes és el que es proposa a la galeria de Cortana Compare Binary Classifiers, que permet, a través de les dades d'una població (32 k + files de 15 característiques) del repositori UCI, i centrant-se en si el salari/any és > que $50 K, comparar diferents classificadors binaris (*Two-Class Averaged Perceptron*, *Two-Class Bayes Point Machine*, *Two-Class Decision Jungle and Two-Class Locally-Deep Support Vector Machine*). La figura següent mostra el model i la seva execució.



I la comparació a la sortida d'aquesta dels dos models de la dreta.

Com s'ha pogut observar, les possibilitats i funcionalitats que presenta la plataforma són molt extenses i cal temps i dedicació per experimentar-hi i utilitzar l'ML en el *cloud* aplicat a les dades d'interès.

És evident que les prestacions i possibilitats sobre les dades massives que presenten aquest tipus de plataformes és realment molt interessant; l'usuari pot experimentar i analitzar els seus models/dades sense haver de disposar/gestionar la infraestructura o la plataforma subjacent. Es recomana consultar la informació i els casos d'ús proposats a la documentació per a més detalls i exemples [Aml].

7. Conclusions

Tal com s'ha pogut apreciar, hi ha una **gran** quantitat de metodologies, eines i plataformes per tractar les dades massives disponibles perquè els usuaris, simplement, s'hagin de dedicar a la tasca d'anàlisi o a desenvolupar els models de predicció sobre les seves dades. Els models de feina poden ser locals o en el *cloud* o mixtos, per exemple, MapR en local per als desenvolupaments preliminars i després passar a AWS amb MapR amb l'escalabilitat i conjunt de dades adequat (**NA 2021:** encara es pot utilitzar MapR Data Platform Community Edition Versió 6.1-MEP6.1 sobre AWS, però s'ha de tenir en compte que és l'última versió generada per MapR abans que la comprés HPE. Consulteu més informació a https://aws.amazon.com/marketplace/pp/prodview-7jnqj26y3g7f4).

S'ha de tenir en compte que tota aquesta tecnologia és emergent (com s'ha demostrat a les compres d'empreses i el desenvolupament d'eines en relació amb el 2016) i que es pot comprovar que hi ha una gran «ebullició» sobre aquests temes ja que, donat l'interès que ha despertat i les facilitats de processament a través d'eines i el *cloud*, hi ha una gran quantitat d'empreses i institucions que volen analitzar les seves dades per millorar els seus processos/predir comportaments o simplement tenir informació de com evolucionen els seus productes/serveis.

La ciència de les dades és una realitat, i en bona part gràcies al *cloud*, ja que en transformar-ho tot en PaaS o SaaS ha permès acostar la tecnologia a les dades i aquestes als usuaris que no entenen d'infraestructura/administració, però sí de dades i negoci. Com a mostra d'aquesta ciència de les dades, darrerament ha sorgit formació específica a les universitats per generar professionals que sàpiguen de dades i el seu processament; se l'ha denominat *data scientist*. Des que el 2013 es va crear l'*IEEE Task Force on Data Science and Advanced Analytics*, tot ha estat una successió d'esdeveniments que mostren aquest interès sobre el tema; per exemple, l'any següent, la primera conferència internacional sobre Data Science and Advanced Analytics (en aquest moment hi ha la crida de presentació de treballs per a l'edició del 2022 que se celebrarà a la Xina) o publicacions com *Journal of Big Data* (Springer), *Big Data Research* (Elsevier) o *Transactions on Big Data* (IEEE), entre altres, són una mostra de la importància i la recerca realitzada sobre aquests temes.

Com a punt final és interessant (tenint en compte el que s'ha analitzat i experimentat en els apartats anteriors) rellegir l'informe *Big data: The next frontier for innovation, competition, and productivity* [Bdn], esmentat a l'inici del capítol,

per comparar l'evolució i les prediccions l'any 2011 i l'actualitat (per exemple, Top 11 Data Science Trends To Watch in 2021 o 7 Key Data Science Trends For 2021-2025).

Activitats

1. Instal·leu i desplegueu una aplicació de proves sobre Apache Hadoop. Analitzeu i valoreu les possibilitats de la plataforma i de les diferents eines i la seva potencialitat sobre diferents conjunts de *big data* obtinguts de repositoris públics.

2. Repetiu l'experiència per a Spark.

3. Considerant la versió 6.1 de MapR, desplegeu-la mitjançant el Sandox i executeu una de les aplicacions desenvolupades a l'exercici anterior. Valoreu i analitzeu les possibilitats de la plataforma creant una taula comparativa de les diferents opcions i equivalències, avantatges i desavantatges.

4. Repetiu l'experiència sobre Google DataProc.

5. Realitzeu una anàlisi comparativa de dades utilitzant diferents conjunts de dades/anys de repositoris públics amb l'eina Google BigQuery.

6. Feu una anàlisi i una avaluació detallada d'un cas de predicció utilitzant Azure ML, basats en dades públiques i segons algun dels exemples publicats a Github o quaderns de Jupyter a la pàgina de la documentació.

Glossari

Ambari, Avro, Cassandra, Chukwa, HBase, Hive, Mahout, Pig, Spark, Tez, ZooKeeper Principals eines de l'entorn *open-source* Apache Hadoop.

Azure Data Lake Eina d'Azure que implementa un magatzem de dades massives.

Azure Machine Learning Entorn d'Azure per a l'aprenentatge automàtic.

Azure SQL Entorn d'Azure per a la tasca treball amb BD.

Azure Storage (*blobs* i taules) Font de dades acceptades per Azure ML.

big data Dades massives.

BigQuery Eina de Google (SaaS) per al processament de grans conjunts de dades (petabytes) sense infraestructura.

business intelligence Tècniques per transformar les dades en informació i aquesta en coneixement per millorar el procés de presa de decisions en l'àmbit empresarial.

canvas Espai on es pot «dibuixar» un diagrama a l'entorn d'Azure ML.

CDH, Cloudera Manager, Cloudera Director Distribució de Hadoop, entorn de gestió i entorn de traspàs de dades al *cloud* respectivament de la companyia Cloudera.

chunk Seqüència/segment de dades.

cloud computing Proposta tecnològica que permet accedir a aplicacions o serveis remots en forma ubiqua a través d'un navegador; aquests poden ser aprovisionats/alliberats sota demanda i en un temps reduït.

Cloud Dataproc Servei d'Apache Spark i Apache Hadoop a Google Cloud.

Cloudera, Hortonworks, MapR Tres de les distribucions més importants que utilitzen Apache Hadoop com a base (el 2021, Hortonworks va ser adquirida per Cloudera i la seva plataforma integrada en els seus productes i accés mitjançant subscripció; MapR va ser comprada per HPE, i la plataforma avui es diu HPE Data Fabric, encara que segueix amb la numeració de versions de MapR versió 6.2 disponible el desembre de 2021).

clustering Algoritmes d'agrupació de dades en aprenentatge automàtic.

Cortana Intelligence Entorn d'Azure ML amb algoritmes i casos d'ús per a l'aprenentatge automàtic.

CRM (*customer relationship management*) Eina/model de gestió d'una organització que es basa en la satisfacció del client.

daemon Procés que s'executa independent de la resta en un SO i que generalment atén les peticions d'un servei.

data mining Tècniques que permeten trobar comportaments predictius combinant mètodes estadístics sobre les dades.

***deep learning* o aprenentatge profund** Tècniques d'ML que modelen abstraccions d'alt nivell sobre les dades utilitzant transformacions no lineals múltiples.

desviació mitjana de l'error (MAE), arrel quadrada d'errors (RMSE), error absolut relatiu (RAE), error al quadrat relatiu (RSE), coeficient de determinació (CD) Mesures estadístiques habituals per a la comparació de resultats (utilitzades en les prediccions obtingudes per ML).

exabytes Mesura en bytes equivalent a un milió de terabytes (1 exabyte = 10^{18} bytes) utilitzada per mesurar les dades massives.

fork-join Sentència de programació (també utilitzada en els processos) que implica una bifurcació amb processament concurrent i una unificació/sincronització posterior.

Gartner Consultora en TIC.

Hadoop YARN, MapReduce, HDFS, Hadoop Common Les quatre parts essencials de l'entorn Apache Hadoop.

Hadoop Plataforma *open-source* per al còmput distribuït, de confiança i escalable.

HDinsight (Hadoop) Entorn Hadoop d'Azure.

Hortonworks (HDP) Plataforma Hadoop de la companyia del mateix nom.

IBM Un dels grans actors en el tema de les dades massives.

IdC (internet de les coses) Interconnexió a través d'internet de dispositius de consum.

linear regresion Algoritme de predicció basat en la relació d'una variable escalar i una altra (o unes altres) a través d'una relació lineal.

local (standalone), pseudo-distributed, fully-distributed Tres dels modes en els quals es pot instal·lar i funcionar la plataforma Hadoop.

***machine learning* o aprenentatge automàtic** Tècniques d'aprenentatge automàtic.

Machine Learning Studio Entorn d'ML interactiu d'Azure.

MapR Converged Data Platform Plataforma Hadoop de la companyia MapR.

MapReduce (MR) Model de programació orientat al còmput paral·lel.

***massive paralell processing* (MPP)** Execució coordinada/sincronitzada d'un programa sobre múltiples processadors que treballen en diferents parts del mateix programa.

master-workers Paradigma de computació distribuïda on el *master* reparteix tasques que realitzen els *workers* concurrentment i retornen els resultats al *master*.

NoSQL (Not Only SQL) Base de dades per a dades no estructurades.

paradigma NoOps Tècnica que permet l'execució d'una aplicació en mode SaaS sense haver-se de preocupar per la infraestructura, ni per la base de dades si es necessita.

Pub/Sub, Dataflow, ML Engine Eines de dades massives de Google complementàries a DataProc i BigQuery.

semistructured data Dades parcialment estructurades a través d'un fitxer XML o Json.

shared nothing Arquitectura de còmput distribuïda on cada node és independent/autosuficient de la resta.

streams Flux de dades (dinàmic).

structured data Dades classificades per una estructura/tipus utilitzats en una BD relacional.

text analytics Tècnica d'anàlisi de dades continguda en un text en llenguatge natural.

unstructured data Dades no estructurades com, per exemple, les utilitzades en BD NoSQL.

***workflow* o cicle de treball** Seqüència de processos/fluxos d'informació que representa una seqüència ordenada de treball/processament.

zettabytes Mesura en bytes equivalent a 1.000 milions de terabytes (10^{21} bytes) utilitzada per quantificar les dades massives d'un futur proper.

Bibliografia

Tots els enllaços s'han visitat el desembre del 2021.

[Abd] Awesome-bigdata. Onur Akpolat. <https://github.com/onurakpolat/awesome-bigdata>

[Ahd] Apache Hadoop Project. <http://hadoop.apache.org/>

[Aml] Azure Machine Learning. <https://docs.microsoft.com/es-es/azure/machine-learning/machine-learning-what-is-machine-learning>

[Bad] Big Data Storage Architecture Design in Cloud Computing (2016). Chen, X.; Wang, S.; Don, Y; Wang, X. Big Data Technology and Applications. BDTA 2015, CCIS 590 (pág. 7-14).

[Bda] Big Data: A Revolution That Will Transform How We Live, Work, and Think (2013). Mayer-Schönberger, V.; Cukier, K. Houghton Mifflin Harcourt.

[Bdd] Big Data for Development. A Review of Promises and Challenges (2015). Hilbert, M. <http://www.martinhilbert.net/big-data-for-development/>

[Bdn] Big data: The next frontier for innovation, competition, and productivity (2011). Manyika, J.; Chui, M.; Brown, B.; Bughin, J.; Dobbs, R.; Roxburgh, C. i altres. McKinsey Global Institute. <http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation>

[Bds] Is Big Data Still a Thing? (The 2016 Big Data Landscape). Turck, Matt. FirstMark. <http://mattturck.com/2016/02/01/big-data-landscape/> (2021 Landscape Forbes: <https://www.forbes.com/sites/brunoaziza/2021/10/03/the-2021-data-landscape-is-out-now-what/?sh=7b33996d4280>)

[Ccb] Del *cloud computing* al *big data*. Torres i Viñals, Jordi. FUOC. <http://www.jorditorres.org/wp-content/uploads/2012/03/Del.Cloud_.Computing.al_.Big_.Data_.JordiTorres.ES_.pdf>

[Cdo] Cloudera Documentation (CDH) (2021). <https://www.cloudera.com/documentation/enterprise/5-6-x/topics/introduction.html>

[Cma] Cloudera Manager (2021). <https://www.cloudera.com/documentation/manager/5-1-x.html>

[Dbd] Design of big data processing system architecture based on Hadoop under the cloud computing. Duan, Chun Mei. Applied Mechanics and Materials (vol. 556-562, pág. 6.302-6.306). DOI:10.4028/www.scientific.net/AMM.556-562.6302.

[Fbd] Hadoop Distributions Reviews and Ratings (2021). <https://www.gartner.com/reviews/market/hadoop-distributions>

[Gmr] Why Google Capital Placed Its Hadoop Bet On MapR?*Forbes*. <https://www.forbes.com/sites/danwoods/2014/06/30/why-google-capital-placed-its-hadoop-bet-on-mapr/#3cff542678e7>

[Ghs] Spark and Hadoop on Google Cloud Platform Documentation (2021). <https://cloud.google.com/hadoop/>

[Hao] Hadoop Architecture Overview. Coppa, Emilio. <http://ercoppa.github.io/HadoopInternals/HadoopArchitectureOverview.html>

[Hdp] Hortonworks Data Platform (2021). <http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.5.3/bk_release-notes/content/index.html>

[Hui] Hadoop 2.6.5 Installing on Ubuntu 16.04 (Single-Node Cluster) (2017). Hong, K. <http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_16_04_single_node_cluster.php> <http://www.bogotobogo.com/Hadoop/BigData_hadoop_Running_MapReduce_Job.php>

[Hsn] Hadoop: Setting up a Single Node Cluster (2021). <https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html>

[Hfd] Hadoop For Dummies (2014). John Wiley & Sons. ISBN: 978-1-118-60755-8.

[LAS] Learning Apache Spark 2 (2017). Asif Abbasi, Muhammad. <https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781785885136>

[Mra] MapR Academy. Free Courses. A: 2021 HPE Ezmeral. <https://learn.ezmeral.software.hpe.com/>

[Mrd] MapR Documentation (2021). <https://docs.datafabric.hpe.com/62/MapROverview/c_overview_intro.html>

[Mrt] MapReduce Tutorial. <http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

[PyD] Pyspark DataFrame Operations - Basics | Pyspark DataFrames. <https://datanoon.com/blog/pyspark_dataframe_operations/>

[SDG] Spark: The Definitive Guide. Big Data Processing Made Simple (2018). Chambers, Bill; Zaharia, Matei. O'Reilly.

[Spa] Spark for Dummies. IBM Limited Edition (2019). Schneider, Robert D; Karmiol, Jeff. <https://www.ibm.com/downloads/cas/WEB4XBOR> (promocionado por IBM).

[SQL] Spark SQL, DataFrames and Datasets Guide. <https://spark.apache.org/docs/3.1.1/sql-programming-guide.html>

[Wbd] De Mauro, A.; Greco, M.; Grimaldi, M. (2015). What is big data? A consensual definition and a review of key research topics. AIP Conference Proceedings 1644 (pág. 97-104). <http://aip.scitation.org/doi/abs/10.1063/1.4907823>

[Wmr] DataOps Requires a Modern Data Fabric (enlace original de Why MapR?) <https://www.hpe.com/us/en/resources/software/dataops-data-fabric.html>

[Ydo] YARN Documentation. <http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html>

[Iar] Icones amb llicència d'ús lliure. <http://www.customicondesign.com> <http://icons8.com>