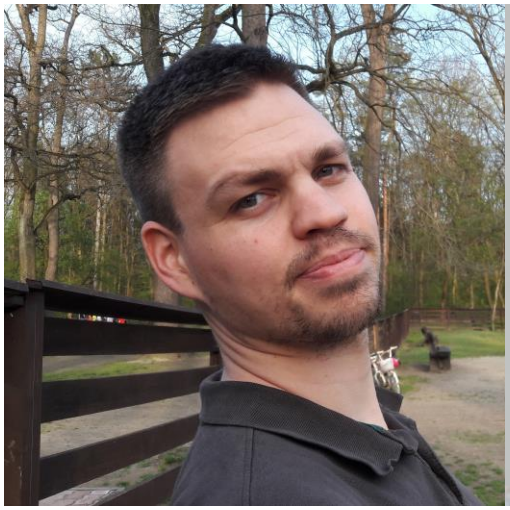



Dr. Hegedűs Péter, Dr. Ferenc Rudolf

Nagyméretű adatbázisok

Jelen tananyag a Szegedi Tudományegyetemen készült az Európai Unió támogatásával.

Projekt azonosító: EFOP-3.4.3-16-2016-00014

Szegedi Tudományegyetem
Cím: 6720 Szeged, Dugonics tér 13.
www.u-szeged.hu
www.szechenyi2020.hu

Apache Hadoop és HDFS

Összefoglalás

Az olvasó ebből az olvasóleckéből megtanulhatja az Apache Hadoop keretrendszer felépítésének és működésének alapjait. Áttekintjük a keretrendszer központi komponenseit és azok feladatát, majd kicsit részletesebben tárgyaljuk a Hadoop környezet erőforrás kezelését, tárolási és számítási modelljét. Végezetül áttekintjük a Hadoop ökoszisztéma főbb elemeit, olyan eszközöket, amelyek a Hadoop klaszterre épülnek, vagy képesek azzal együttműködni.
A lecke fejezetei:

- 1. fejezet: **Az Apache Hadoop keretrendszer áttekintése (olvasó)**
- 2. fejezet: **Az Apache Hadoop keretrendszer alap komponenseinek magas szintű ismertetése (olvasó)**
- 3. fejezet: **Az Apache Hadoop ökoszisztéma bemutatása (olvasó)**

Téma típusa: **elméleti**
Olvasási idő: **45 perc**

📖 1. fejezet
Az Apache Hadoop keretrendszer

Az Apache Hadoop [1] egy olyan keretrendszer, amely hatalmas méretű adatok elosztott feldolgozását teszi lehetővé számítógépek klaszterén egyszerű programozási modellek segítségével. Skálázhatóra tervezték, ami azt jelenti, hogy egyetlen szervertől egészen a több ezer gép klaszterbe történő szervezéséig is működik, ahol minden egyes számítógép lokális tárhelyet és számítási kapacitást biztosít. A BigData alkalmazások egyik de-facto szabvány keretrendszere, legtöbb alkalmazásban megtalálható valamilyen Hadoop komponens.

A keretrendszer magja két alapvető komponenssel rendelkezik, az egyik az adatok tárolását megvalósító Hadoop Distributed File System (HDFS), a másik pedig az adatok feldolgozásáért felelős MapReduce programozási modell. A Hadoop alapvető koncepciója, hogy a fájlokat nagy blokkokra osztja, amiket aztán szétoszt a klaszter egyes csomópontjaira. Az adatfeldolgozáshoz aztán nem az adatokat mozgatja újra egy helyre, hanem a feldolgozó programkódot másolja le minden egyes adat csomópontra, ahol aztán az adatfeldolgozás lokálisan történik, párhuzamosan egyszerre az összes csomóponton. Ez a módszer a lokális adatfeldolgozás előnyeit használja ki, ami így hatékonyabb feldolgozást eredményez, mint hagyományos architektúrák esetén. A Hadoop minden olyan területen használható, ahol batch alapú adat feldolgozásra van szükség (a Hadoop önmagában nem alkalmas valós idejű alkalmazásokhoz), és a hatalmas adatmennyiség kezelését a párhuzamos futtatás segíti.

Maga az alap keretrendszer szoftver modulok szintjén négy alapvető egységből áll:

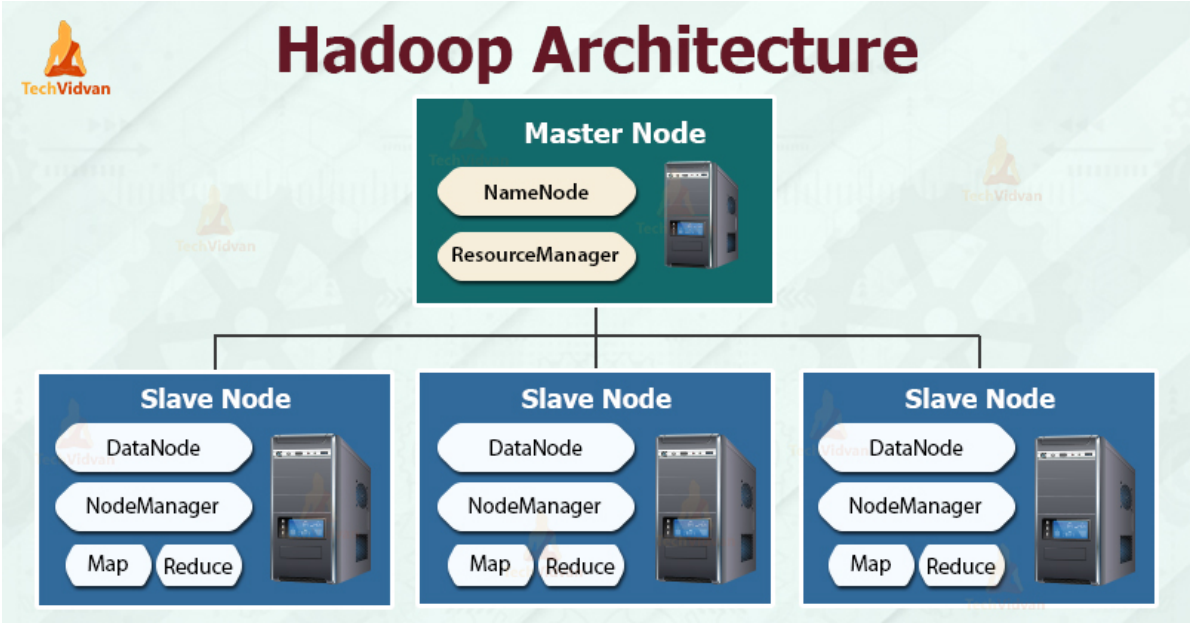
- *Hadoop Common* - közös könyvtárak, amelyeket a többi modul használ
- *Hadoop Distributed File System (HDFS)* - elosztott fájlrendszer, ami az adatokat hagyományos számítógépeken elosztva tárolja, ezáltal hatalmas adatátviteli sebességet nyújt
- *Hadoop YARN* - erőforrás menedzsment komponens a számítási kapacitás kezeléséhez és a feladatok ütemezéséhez
- *Hadoop MapReduce* - a MapReduce programozási modell implementációja nagy méretű adatok feldolgozásához

A Hadoop elnevezés sokszor nem csak ezekre az alap komponensekre vonatkozik, hanem egy egész szoftver ökoszisztémára, amely magában foglalja az alap komponenseket, és azokat az eszközöket/alkalmazásokat, amelyek ezekre épülve vagy ezek mellett üzemelhetnek. Az ökoszisztéma elemeit a 3. fejezetben részletezzük.

A Hadoop keretrendszer MapReduce és HDFS komponenseit két Google cikk inspirálta [3, 4], és maga a keretrendszer többnyire Java nyelven íródott, valamint néhány eszköz C-ben és shell szkript segítségével.

Apache Hadoop architektúra

A lenti kép egy tipikus Hadoop klasztert mutat. Ezt egy ~~master~~ primary node és több ~~slave~~ worker node alkotja. A primary node tartalmazza a NameNode-ot, valamint a job és task tracker komponenseket. A worker node-ok DataNode-ként és task tracker-ként egyaránt funkcionálnak, azaz adatokat is tárolnak, illetve MapReduce job-okat is végre tudnak hajtani. Ez a jellemző konfiguráció, noha van lehetőség csak adat vagy csak feldolgozó node-ok használatára is.



> https://techvidvan.com/tutorials/wp-content/uploads/sites/2/2020/03/hadoop-architecture.jpg

Nagyon nagy klaszterek esetén a HDFS node-okat egy dedikált NameNode szerver szolgálja ki, ez végzi a fájlok indexelését, valamint e mellett még egy másodlagos NameNode is üzemel, ami a memória mentéséért felelős, ezzel megelőzve az esetleges adatvesztést. Hasonlóan, a feladatok ütemezéséhez egy önálló JobTracker szerver is futhat a klaszteren belül.

📖 2. fejezet

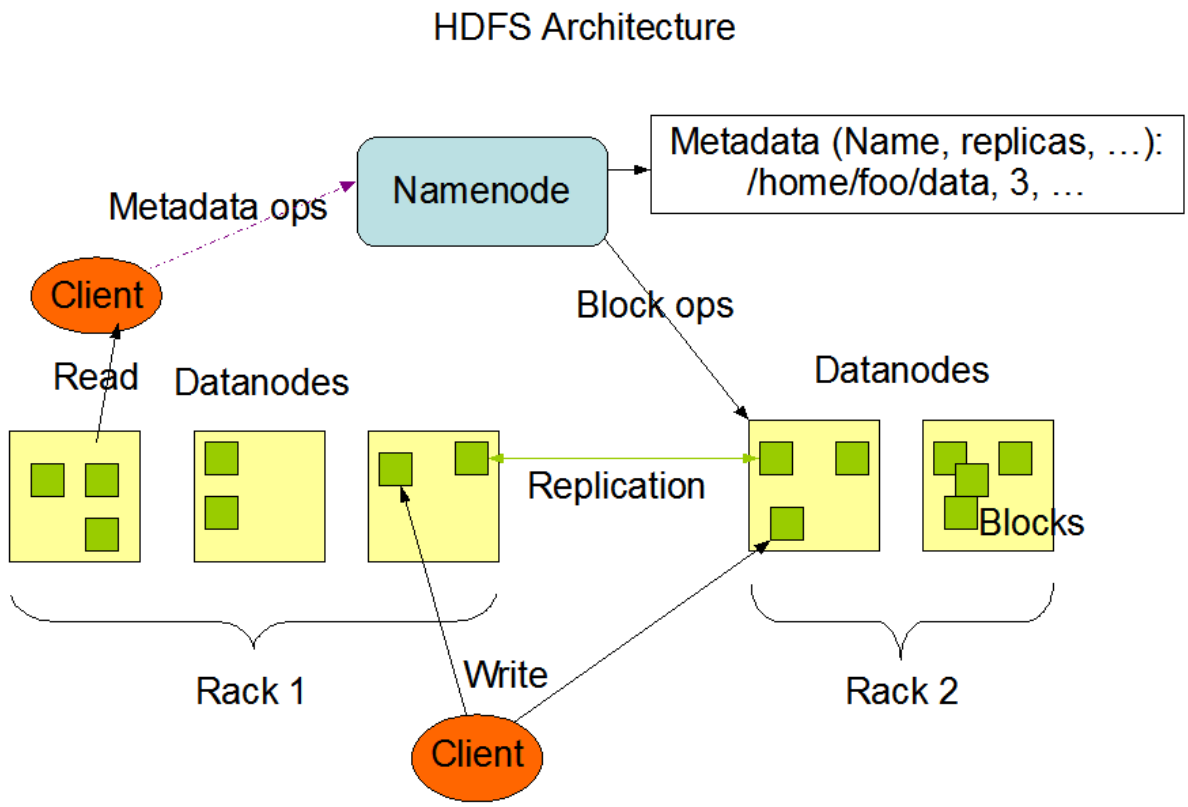
Apache Hadoop core komponensek

Ebben a fejezetben az Apache Hadoop keretrendszer core komponenseit tekintjük át. Az egyes modulok részletes bemutatását külön olvasóleckék tartalmazzák, itt a működésük magas szintű bemutatása a célunk.

Hadoop Distributed File System (HDFS)

Az Apache Hadoop keretrendszer skálázható és elosztott adattárolásáért felelős fájlrendszer komponens, amit hagyományos számítógépek klaszterén való működésre terveztek. Noha sokban hasonlít más elosztott fájlrendszerekhez, jelentős különbségek is vannak azokhoz képest. Magas hibatűrés jellemzi, és alacsony költségű hardware-en való futtatásra tervezték. Nagy sávszélességű adatelérést biztosít azon alkalmazások számára, akik hatalmas méretű adatokkal dolgoznak (a.k.a. BigData alkalmazások). Nem teljesen POSIX kompatibilis annak érdekében, hogy lehetővé tegye a rendszer adatokhoz való stream-elt hozzáférést. A HDFS fájlrendszert elsődlegesen nagy adatátviteli kapacitásra és batch alapú adatfeldolgozásra tervezték.
A rendszer ~~master/slave~~ primary/worker architektúrával rendelkezik a következő komponensekkel (lásd lenti ábra):

- *NameNode*: központi szerver, egyetlen primary csomópont, amely fogadja a kliens hozzáféréseket, és szabályozza a fájlokhoz való hozzáférést
- *DataNode*: tipikusan minden klaszter csomóponton fut egy DataNode, amely az alatta levő hardware segítségével tárolja az adatokat

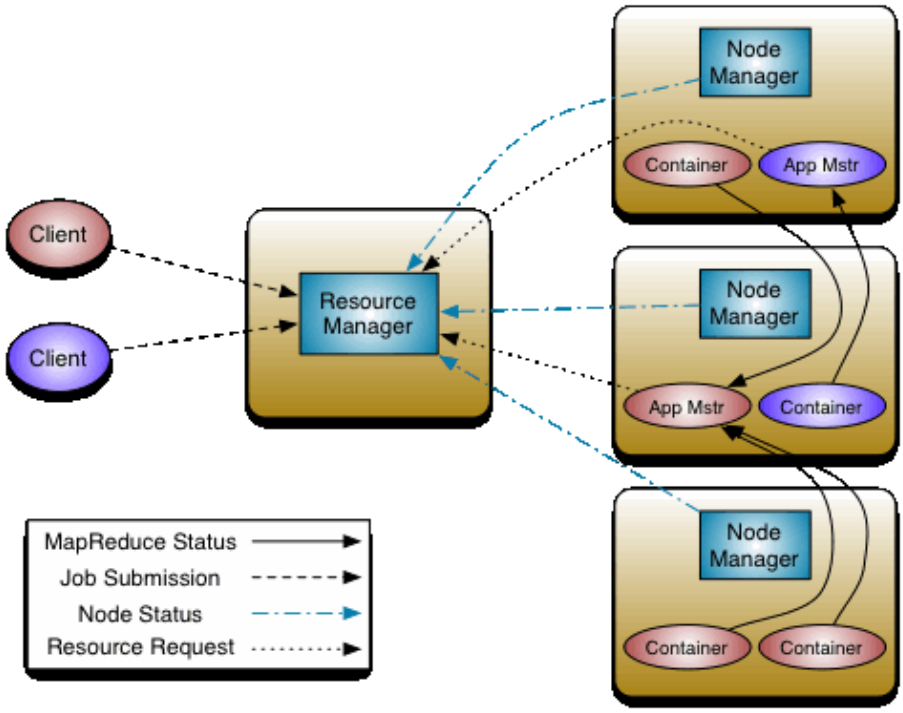


> https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/images/hdfsarchitecture.png

YARN

Az Apache Hadoop 1-es verziójában még nem létezett, a feladatok (job-ok) ütemezését és szétosztását a MapReduce engine végezte. A 2-es verziótól azonban megjelent a YARN (Yet Another Resource Negotiator), ami leváltotta a MapReduce engine-t. A YARN feladata az erőforrások hatékony elosztása az alkalmazások között. A YARN két daemon folyamatból áll:

- *Resource manager* (RM): job-ok követéséért és az erőforrások allokációjáért felelős
- *Application master* (AM): a job-ok futását monitorozza



> https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/yarn_architecture.gif

Az `RM` globális, egyetlen példány fut belőle az egész klaszteren belül, míg minden egyes külön alkalmazás rendelkezik saját `AM` -el. Ebben a kontextusban egy job-ot, vagy job-ok egy irányított körmentes gráfját nevezzük. Az `RM` az erőforrások elosztásáért felelős, míg a `NodeManager` a klaszter minden számítógépén futó keretrendszer ügynök (agent), amely az erőforrások felhasználásáért és monitorozásáért felel (CPU, memória, tárhely, hálózat), valamint az információt visszajuttatja az `RM`-hez. Az `AM` valójában egy keretrendszer specifikus könyvtár, aminek elsődleges feladata az erőforrások egyeztetése az `RM`-el, valamint együttműködés a `NodeManager`-el a feladatok elvégzésének és monitorozásának érdekében.

A `ResourceManager` két fő komponensből áll:

- *Scheduler*
- *ApplicationsManager*

A Scheduler (ütemező) felelős az erőforrások lefoglalásáért a különböző alkalmazások számára. Az ütemező a feladatok kiosztásán kívül nem csinál semmi mást, nem követi nyomon a feladatokat, stb. Arra sem ad garanciát, hogy az alkalmazás vagy hardware hibák miatt sikertelen feladatok újra lesznek indítva. Az ütemező az alkalmazások erőforrás igénye alapján végzi a feladatát, amelyek absztrakt módon, ún. erőforrás *Container*-ek (CPU, memória, tárhely, hálózat, stb.) segítségével kerülnek meghatározásra. Az ütemező által használt ütemezési algoritmus plug-in módon bővíthető/cserélhető. Jelenleg támogatott ütemező megvalósítások például a CapacityScheduler, vagy a FairScheduler.

Az ApplicationsManager feladata a job kérések fogadása, az első container meghatározása az alkalmazás függő ApplicationMaster végrehajtásához és az ApplicationMaster container újraindítása hiba esetén. Ezek mellett ő a felelős a futó feladatok nyomon követéséért és az előrehaladásuk monitorozásáért.

A YARN klaszterek több ezer node fölé történő skálázásának érdekében, a YARN támogatja az ún. **Federation** opciót a YARN Federation funkción keresztül. A Federation lehetővé teszi több YARN (al-)klaszter transzparens módon történő összekapcsolását, ami a külvilág felé egyetlen hatalmas klaszterként jelenik meg. Ez a funkció használható arra, hogy több önálló klasztert egyesítve

használjunk arra, hogy hatalmas méretű feladatokat hajtsunk végre.

MapReduce

A Hadoop MapReduce egy keretrendszer, amely lehetővé teszi olyan alkalmazások készítését, amelyek hatalmas mennyiségű adatot (több terra byte-os adathalmazok) párhuzamosan dolgoznak fel nagyméretű hagyományos számítógépekből álló klasztereken (több ezer csomópont) megbízható és hibatűrő módon.

Egy MapReduce *job* általában felosztja a bemeneti adatkészletet független részekre, amelyeket a leképező (*map*) feladatok teljesen párhuzamosan dolgoznak fel. A keretrendszer rendezi a leképezések kimeneteit, majd ezután a *reduce* feladatok ezt feldolgozzák. Általában a feladat bemenete és kimenete egy fájlrendszerben kerül tárolásra. A keretrendszer gondoskodik a feladatok ütemezéséről, megfigyeléséről és a sikertelen feladatok újbóli végrehajtásáról.

A számítási és a tárolási csomópontok általában azonosak, vagyis a MapReduce keretrendszer és a Hadoop elosztott fájlrendszere (lásd a HDFS Architecture Guide) ugyanazon csomóponton futnak. Ez a konfiguráció lehetővé teszi a keretrendszer számára, hogy azon a csomóponton hajtsa végre a feladatokat, ahol az adatok is vannak, így a feldolgozás nagyon hatékony lesz, és magas összesített sávszélességet eredményez a klaszteren belül.

Egy minimális MapReduce alkalmazáshoz meg kell határoznunk a bemeneti/kimeneti adatok helyét és meg kell valósítanunk a *map* és *reduce* függvényeket a megfelelő interfészek és / vagy absztrakt osztályok implementálásával. Ezek és más job paraméterek képezik az ún. *job konfigurációt* .

A Hadoop *job kliens* ezután elküldi a feladatot (jar / végrehajtható stb.) és a konfigurációt a `ResourceManager`-nek, amely kiosztja a szoftvert / konfigurációt a worker node-oknak, elvégzi a feladatok ütemezését és megfigyelését, valamint gondoskodik az állapot- és diagnosztikai információknak a klienshez történő visszacsatolásáról.

Bár a Hadoop keretrendszert Java nyelven írták, a MapReduce alkalmazásokat nem kell feltétlenül Java-ban írni.

- A Hadoop streaming egy olyan segédprogram, amely lehetővé teszi a felhasználók számára, hogy feladatokat hozzanak létre és futtassanak bármilyen futtatható programmal (pl. Shell segédprogramokkal), mint mapper és / vagy reducer.
- A Hadoop Pipes egy SWIG kompatibilis C ++ API a MapReduce alkalmazások (nem JNI alapú) megvalósításához.

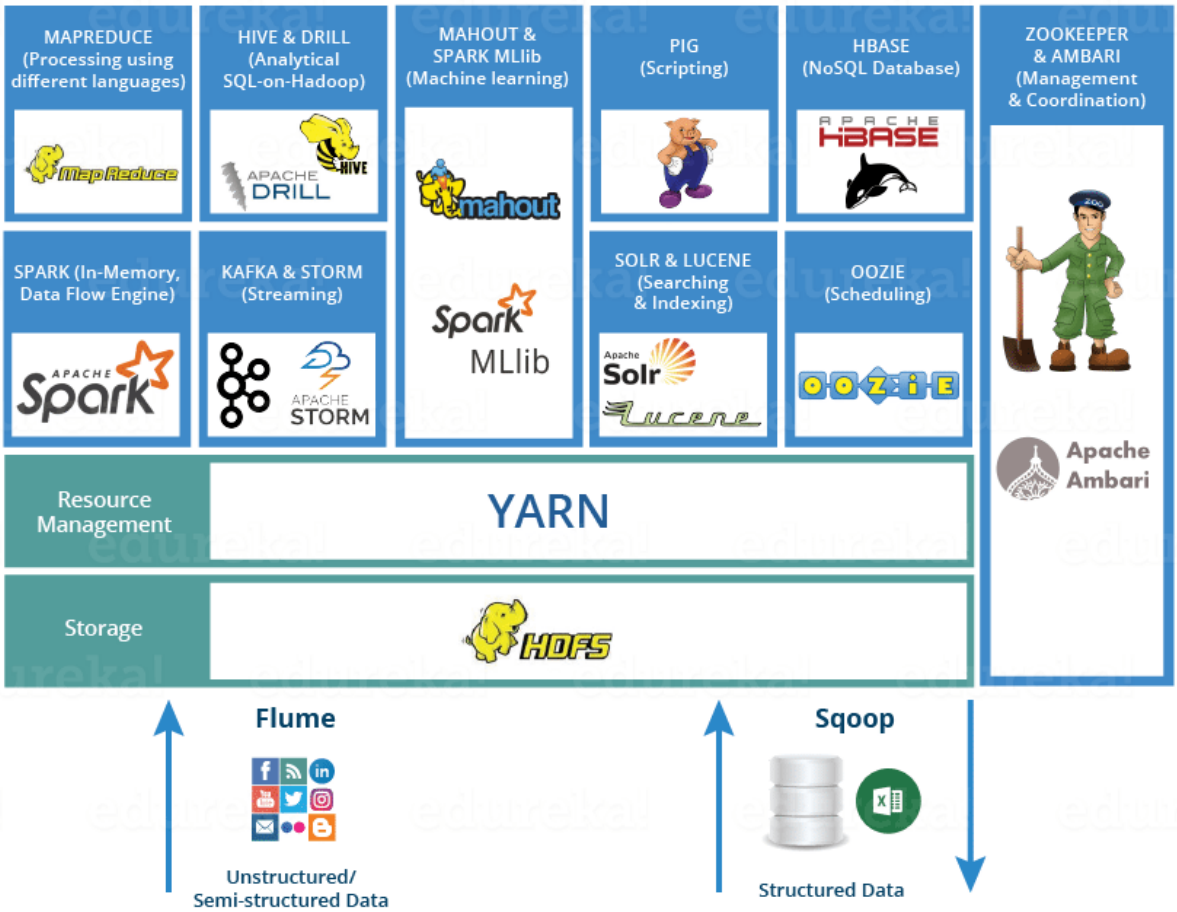
📖 3. fejezet

Az Apache Hadoop ökosztisztéma

Ahogy már említettük, a Hadoop nem csak egy keretrendszer, hanem egy teljes szoftver ökoszisztéma, azaz maga az alap modulok, és az arra ráépülő vagy mellette párhuzamosan futtatható eszközök összessége. A Hadoop ökoszisztéma fontosabb eszközei a következők:

- Spark - egyszerű és gyors programozási modell Hadoop adatok feldolgozásához (in-memory adatfeldolgozás)
- Pig - egy magasz szintű adatfolyam nyelv és végrehajtó motor párhuzamos számításokhoz
- Hive - adattárház infrastruktúra komplex adat lekérdezési lehetőségekkel (SQL szintaxist használva)
- HBase - egy skálázható, elosztott NoSQL adatbázis hatalmas méretű adattáblák tárolásához (hasonló, mint a Google BigTable [5] rendszere)
- Mahout - egy skálázható gépi tanulási (machine learning) és adatbányász rendszer

- Spark MLlib - az Apache Spark skálázható gépi tanuló keretrendszere, amely támogatja a Python és R könyvtárakat
- Apache Drill - SQL szerű lekérdezés Hadoop fölött, tetszőleges NoSQL adatbázishoz (pl. HBase, MongoDB, HDFS, stb.)
- ZooKeeper - elosztott alkalmazások koordinálását végző szoftver
- Oozie - munkafolyamat ütemező rendszer Apache Hadoop job-okhoz
- Flume - nagy méretű, nem strukturált adatok betöltése HDFS-be
- Sqoop- adatok importálása és exportálása HDFS és tetszőleges relációs adatbázis között
- Solr és Lucene - keresés és adat indexelés
- Ambari - egy webes eszköz Hadoop klaszterek menedzseléséhez és monitorozásához
- Tez - YARN fölötti adat folyam programozási modell, amely leváltja a MapReduce megközelítést
- Submarine - egységesített MI keretrendszer elosztott adat klaszter fölé
- Cassandra - skálázható, NoSQL adatbázis
- Avro - adat szerializáló rendszer
- Storm - valós idejű stream feldolgozó keretrendszer ZooKeper fölé
- Impala - elosztott SQL lekérdező engine Hadoop fölé
- Kafka - nyílt forráskódú, elosztott esemény-streaming-platform



> https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2016/10/HADOOP-ECOSYSTEM-Edureka.png

A fenti lista természetesen nem teljes, számos egyéb projekt és eszköz létezik, amely együttműködik az Apache Hadoop keretrendszerrel.

Ellenőrző kérdések

1. Mire szolgál az Apache Hadoop keretrendszer?
2. Mik a Hadoop fő komponensei (core modulok) és mire szolgálnak?
3. Hány DataNode lehet egy Hadoop klaszteren belül?
4. Mik a YARN fő komponensei és milyen feladatot látnak el?
5. Mire szolgál a MapReduce keretrendszer?
6. Mi a HDFS alap architektúrája és miben különbözik más elosztott fájlrendszerektől?
7. Sorolj fel néhány olyan eszközt a Hadoop ökoszisztémából, melyek támogatják a HDFS klaszteren történő gépi tanulást!
8. Milyen eszközök léteznek HDFS-en tárolt adatok SQL-szerű lekérdezéséhez?
9. Sorolj fel néhány olyan NoSQL adatbázist, amelyek együtt tudnak működni egy Hadoop klaszterrel!

Referenciák

[1] https://hadoop.apache.org/

[2] https://en.wikipedia.org/wiki/Apache_Hadoop

[3] https://en.wikipedia.org/wiki/MapReduce

[4] https://en.wikipedia.org/wiki/Google_File_System

[5] https://research.google.com/archive/bigtable.html

[6] https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

[7] https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html