Az Apache Hadoop és a Hadoop elosztott fájlrendszer (HDFS)

Az Apache Hadoop

Az **Apache Hadoop** egy nyílt forráskódú, hibatűrő, elosztott rendszer nagy adathalmazok tárolásához és feldolgozásához.
A Hadoop keretrendszere elosztott adatmodellt, elosztott programozási modellt és klaszter erőforrás-kezelési megoldást (YARN) is biztosít.
Adatmodell: HDFS elosztott fájlrendszer
Egyszerű adatmodell, bármilyen típusú és formátumú adat illeszkedik hozzá.

Az Apache Hadoop - HDFS

A Hadoop Distributed File System (HDFS) egy elosztott fájlrendszer, ami a Google File System-en (GFS) alapul. A HDFS hasonló a többi elosztott fájlrendszerhez, de van néhány jelentős különbsége azokhoz képest:

- magas szintű hibatűrés;
- olcsó, általános hardveren történő működésre optimalizált;
- nagy áteresztőképességet biztosít;
- kifejezetten nagy adatmennyiségeket feldolgozó alkalmazásokhoz lett tervezve.

Az Apache Hadoop - HDFS

A HDFS jellemzői:

Egy létező fájlrendszer (pl. Ext3) felett működik.

Az adatok a betöltéskor kerülnek szétosztásra a klaszter csomópontjai között.

Az adatok frissítése legtöbbször hozzáfűzéssel történik („write once, read many”).

Támogatja a kötegelt és stream jellegű hozzáférést (emiatt néhány POSIX követelménynek nem felel meg).

Az Apache Hadoop - MapReduce

Programozási modell: MapReduce
Hadoop MapReduce egy keretrendszer, ami a MapReduce programozási paradigmát valósítja meg.

Jellemzői:

Terabájtos vagy nagyobb adatok olcsó hardverekből épített, több ezer csomópontot tartalmazó klasztereken történő feldolgozására lett tervezve.

Hibatűrés: az egyes taszkok külön-külön is újraindíthatóak.

Az Apache Hadoop - MapReduce

Legnagyobb előnye, hogy automatikus feladatelosztást és párhuzamosítást tesz lehetővé, így egyszerű absztrakciót biztosít a fejlesztők számára.

Minden MapReduce job két lépésből áll:

map: a feladat szétbontása részproblémákra
reduce: a részeredményekből előállítja a feladat megoldását

A HDFS és a MapReduce: egy MapReduce job esetében minden csomópont a rajta tárolt adatokon dolgozik, együttműködve a HDFS-sel, ami gyorsabb feldolgozást tesz lehetővé.

Az Apache Hadoop

Az Apache Hadoop a Google File System (2003) és a MapReduce programozási paradigma (2004) nyílt forrás-kódú implementációja, ami a Nutch keresőalgoritmus (2005) skálázási problémáinak megoldására készült.

2008: az Apache Software Foundation önálló, kiemelt projektje lett.
2009: Jim Gray's Sort Benchmark: 1 TB rendezése 62 másodperc alatt, 1 PB rendezése 16,25 óra alatt Hadoop-pal a Yahoo-nál.
Napjainkban az egyik legelterjedtebb rendszer nagyon nagy mennyiségű adat elosztott feldolgozásához.

Apache Hadoop – hatékonyság

A Hadoop hatékonyan alkalmazható,

ha a feldolgozandó adat nagy méretű vagy komplexitású, vagy gyorsan generálódik,

ha összetett adatelemzésre van szükség, és

ha az adatokon végzendő elemzési feladat időben változhat.

De: a Hadoop nem hatékony

relációs adatbázisok kiváltására,

ad-hoc jellegű elemzésekhez.

A Hadoop Distributed File System (HDFS)

A **Hadoop Distributed File System (HDFS)** egy elosztott fájlrendszer, amely a Google File System-en alapul.

A HDFS a Hadoop fájlrendszer komponense, amely egy már létező fájlrendszer (pl. Ext3) felett működik.

A fájlrendszer metaadatait egy dedikált csomóponton elkülönítve, míg az alkalmazások adatait a klaszter többi csomópontján tárolja.

A fájlrendszerbeli kommunikáció TCP/IP-alapú protokoll használatával történik.

A tárolás szervezésével támogatja a nagy sebességű elosztott feldolgozást.

HDFS architektúra - Tervezési célok

Hibatűrés:

Jellemzően nagy, olcsóbb szerverekből álló klaszterek támogatása, ahol a hardveres és hálózati hibák gyakoribbak.

Nagy fájlok kezelése:

Akár terabájtos nagyságú fájlok kezelésére is szükség lehet.

Feldolgozási teljesítmény:

Ahol csak lehetséges, ki kell használni az adatok és a feldolgozást végző csomópontok hálózaton belüli elhelyezkedését.

Hordozhatóság:

Heterogén hardver és szoftver platformok támogatása, könnyű bővíthetőség és skálázhatóság.

HDFS architektúra

Tervezési alapkoncepciók

Nagy áteresztőképesség előtérbe helyezése az alacsony késleltetéssel szemben:

A fájlrendszer szervezése a szokásos gyors véletlenszerű hozzáférés helyett stream hozzáférésre van optimalizálva.

Egyszerű konkurencia modell:

Egy stream író – több, párhuzamos stream olvasó

Zárolásmentes írás és olvasás

De: ezek miatt nincs teljes POSIX-kompatibilitás

Feldolgozás az adat mellett:

A feldolgozást végző programot általában kisebb költséggel lehet mozgatni, mint a feldolgozandó nagy mennyiségű adatot.

HDFS architektúra

Tervezési alapkoncepciók

A fájlok szervezésekor a minél nagyobb teljesítmény elérése érdekében figyelembe kell venni az általános klaszter-topológiát:

- Egy nagy klaszter jellemzően több rack-ből áll.
- A rack-ek közötti kommunikációhoz általában legalább két switch-en át kell menni a forgalomnak → A rack-ek közötti késleltetés nagyobb, mint a rack-en belüli.
- Az aggregált rack-en belüli sávszélesség sokkal nagyobb, mint a rack-ek közötti sávszélesség.

A fájlrendszert rack-en belüli párhuzamos hozzáférésre kell optimalizálni, ahol lehetséges.

HDFS – logikai szervezés

HDFS a mester-dolgozó szervezési mintát implementálja:

- Egy mester csomópont: NameNode
  - HDFS metaadatainak tárolása a helyi fájlrendszeren, a fájlrendszer menedzselése
- Több dolgozó csomópont: DataNode-ok
  - HDFS adatainak tárolása a helyi fájlrendszeren.
  - A kliens alkalmazások a DataNode-okon futnak és nem különálló gépeken.
  - Minden csomópont lehetőség szerint a helyi fájlrendszerén tárolt blokkokon dolgozik.

HDFS – fájlszervezés

NameNode

Kezeli a fájlrendszer névterét: hierarchikus névtér (fájl illetve könyvtár fa).

Kezeli a fájlokhoz és könyvtárakhoz tartozó metaadatokat (pl. a blokkok fizikai elhelyezkedése stb.).

Koordinálja az adatok DataNode-okon történő elhelyezését.

Limitált hozzáférés-vezérlést biztosít: a fájlrendszerhez történő hozzáférés szabályozása kevésbé kritikus, ha a kliensek alapvetően megbízhatóak.

NameNode

Egy NameNode jelenléte nélkülözhetetlen a HDFS működéséhez, ennek kiesése elérhetetlenné teszi a teljes fájlrendszert.

A névtér képe és a metaadatok a működés során a csomópont rendszermemóriájában tárolódnak, ezzel növelve a fájlrendszer teljesítményét.

Fontos: a tárolható metaadatok mérete, így a kezelhető fájlok száma, függ a csomópont rendszer-memóriájának méretététől. (Kevesebb, de nagyon nagy méretű fájl kezelésénél hatékonyabb ez a megközelítés.)

NameNode

Ezek periódikusan egy-egy perzisztens fájlba kerülnek mentésre, amelyek egy szerkesztési napló segítségével vannak naprakészen tartva.

A fájlok nagy méretű blokkokban kerülnek tárolásra, az egész klaszteren elosztva:

- Általában 64 vagy 128 MB egy blokk mérete (fájlonként konfigurálható)
- Minden blokk a DataNode-ok helyi fájlrendszerén, fájlként kerül tárolásra

NameNode

Minden blokk *n*-szeresen replikált
- a replikációs faktor fájlonként konfigurálható (alapesetben *n=3*)
- a replikák is a teljes klaszteren elosztva helyezkednek el
- a replikáció célja:
  - nagyobb rendelkezésre-állás elérése
  - teljesítmény növelése
- a replikációs stratégia kritikus mindkét cél esetében

A NameNode kiesésének kiküszöbölésére stand-by NameNode másolatokat lehet alkalmazni, de ehhez külső eszköz szükséges; a HDFS jelenleg nem biztosítja.

DataNode

A fájlok tárolását végzi egy dolgozó csomóponton.
Két fájl reprezentál egy-egy blokk-replikát egy DataNode-on:

- az adatot magát tartalmazó fájl
- az ehhez a fájlhoz tartozó ellenőrzőösszegeket és verzió azonosítót tartalmazó fájl

Egy DataNode a NameNode-hoz történő csatlakozás során egy kézfogási hitelesítést végez. Ennek során:

- ellenőrzi a névtér azonosítóját és a HDFS verziót
- újonnan csatlakozó DataNode-ok esetén megkapja azt a névtér azonosítót, amihez tartozni fog.

DataNode

Minden DataNode esetén szükséges azok regisztrációja a NameNode-on:

Ekkor egy tároló azonosító (storage ID) kerül hozzárendelésre a DataNode-hoz, ami egy egyedi belső azonosító és nem változik.

DataNode – NameNode kommunikáció:

Blokk riport: a tárolt blokk-replikák azonosítására szolgál

Tartalma: blokkazonosító, verzió-azonosító, és a blokk hossza

Először a regisztráció során kerül elküldésre, majd adott periodicitással (alapesetben óránként)

DataNode – NameNode kommunikáció

Szívverés: az elérhetőség jelzésére szolgáló üzenet

- Az alapértelmezett időintervallum 3 másodperc.
- Egy DataNode kiesettnek tekintendő, ha egy adott ideig nem küld szívverés üzenetet (alapesetben 10 percig).
- Ezek a rendszeres üzenetek egyben a skálázhatóságot is biztosítják.

Egy szívverés üzenet a tárhelyről és a terheléseloszlásról tartalmaz információkat:

- a teljes tárkapacitás
- a felhasznált tárhely mérete
- a folyamatban lévő írási és olvasási műveletek száma

A NameNode ezek alapján utasításokat küld a DataNode-nak válaszként.

HDFS – I/O műveletek és blokk kezelés

Fájlok írása és olvasása

Egy alkalmazás egy fájl létrehozásával és az abba történő írással vihet be adatot a HDFS tárba.
Minden fájl esetén a hozzáfűzés, olvasás és törlés műveletek engedélyezettek programozottan.

Blokk kezelés

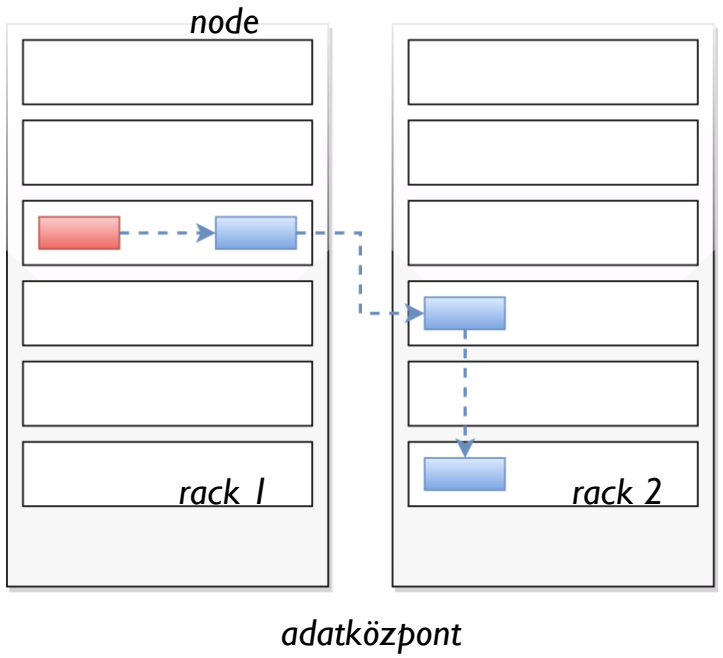
A blokkok elhelyezését és a replikák kezelését a NameNode végzi a DataNode-ok által küldött blokk riportok és a szívverés üzenetek alapján.
Amikor egy új blokkra van szükség, a NameNode egy új blokk azonosítót hoz létre és eldönti, hogy mely DataNode-ok fogják tárolni a blokk replikáit.

HDFS – I/O műveletek és blokk kezelés

Az adatot a DataNode-ok pipeline-jellegűen kapják meg, azaz az első kiválasztott DataNode továbbítja a csomagokat a másodiknak, majd az a harmadiknak stb.

Az írt adat nem áll rendelkezésre az olvasók számára, ameddig a blokk létrehozása le nem zárul minden DataNode-on.

A folyamat végét acknowledgement üzenetekkel jelzik a DataNode-ok a NameNode felé.



HDFS – Replikációs stratégiák

Az alapértelmezett stratégia folyamatosan biztosítja:

- egy DataNode nem tartalmaz egynél több blokk replikát egyik blokk esetében sem;
- legalább egy replika ugyanazon a csomóponton van, amelyik a blokkot írja;
- ha lehetséges, egy rack sem tartalmazza ugyanannak a blokknak kettőnél több replikáját;
- ha lehetséges, egy replika legyen ugyanazon a rack-en, amiben az a csomópont van, amelyik írja a blokkot;
- a blokkok egyenletesen kerüljenek eloszlásra a teljes klaszteren.

Ezeken kívül további stratégiák is definiálhatóak.

HDFS – Kiegyensúlyozás

A fenti stratégiai célok és új DataNode-ok hozzáadása kiegyensúlyozatlanná teheti a klasztert:

- az új blokkok egyformán kerülnek elosztásra a régi és az új DataNode-okon, de a régi blokkok megmaradnak a korábbi helyükön,
- az újabb DataNode-okra így sokkal kisebb terhelés jut kezdetben,
- a feldolgozás folyamán ezek a csomópontok nagyobb hálózati forgalmat generálhatnak, mivel más csomópontokról kell adatot olvasniuk.

A HDFS nem végez automatikus kiegyensúlyozást.

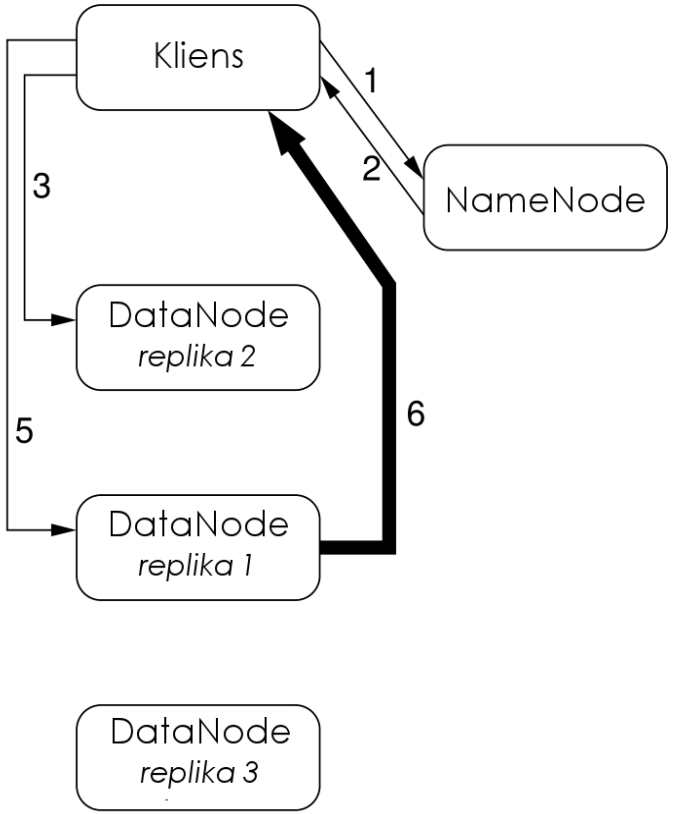
Megoldás: *balancer* segédprogram, amely klaszter szintű kiegyensúlyozást tud végrehajtani.

Fájl olvasás fájlrendszer-hiba esetén

1. A kliens kérést küld az *F* fájl *i* blokkjának olvasására
2. Visszakapja a blokk handler-ét és az elérési helyét
3. Az adat lekérése

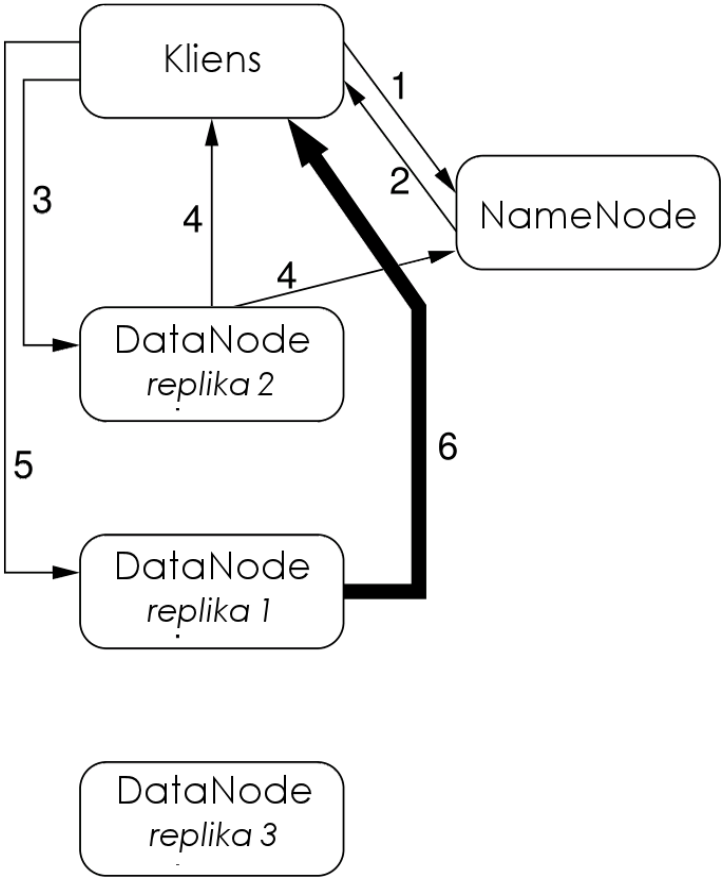
   A legközelebbi replika olvasása.
4. A művelet időtúllépés miatt nem hajtódik végre. Ennek oka, pl. hálózati hiba, DataNode hiba, vagy a DataNode túlterhelt.
5. Az adat újrakérése

   A következő legközelebbi replika olvasása
6. Adatátvitel



Fájl olvasás adathiba esetén

1. A kliens kérést küld az *F* fájl *i* blokkjának olvasására
2. Visszakapja a blokk handler-ét és az elérési helyét
3. Az adat lekérése
4. Hiba észlelése (pl. ellenőrzőösszeg alapján):
   A kliens és a NameNode értesítése a hibáról.
5. Az adat újrakérése
   A következő feltételezhetően jó replika olvasása.
6. Adatátvitel



Dr. Hajdu András, Tóth János: Nagy adathalmazok elosztott feldolgozása című tananyaga alapján készült