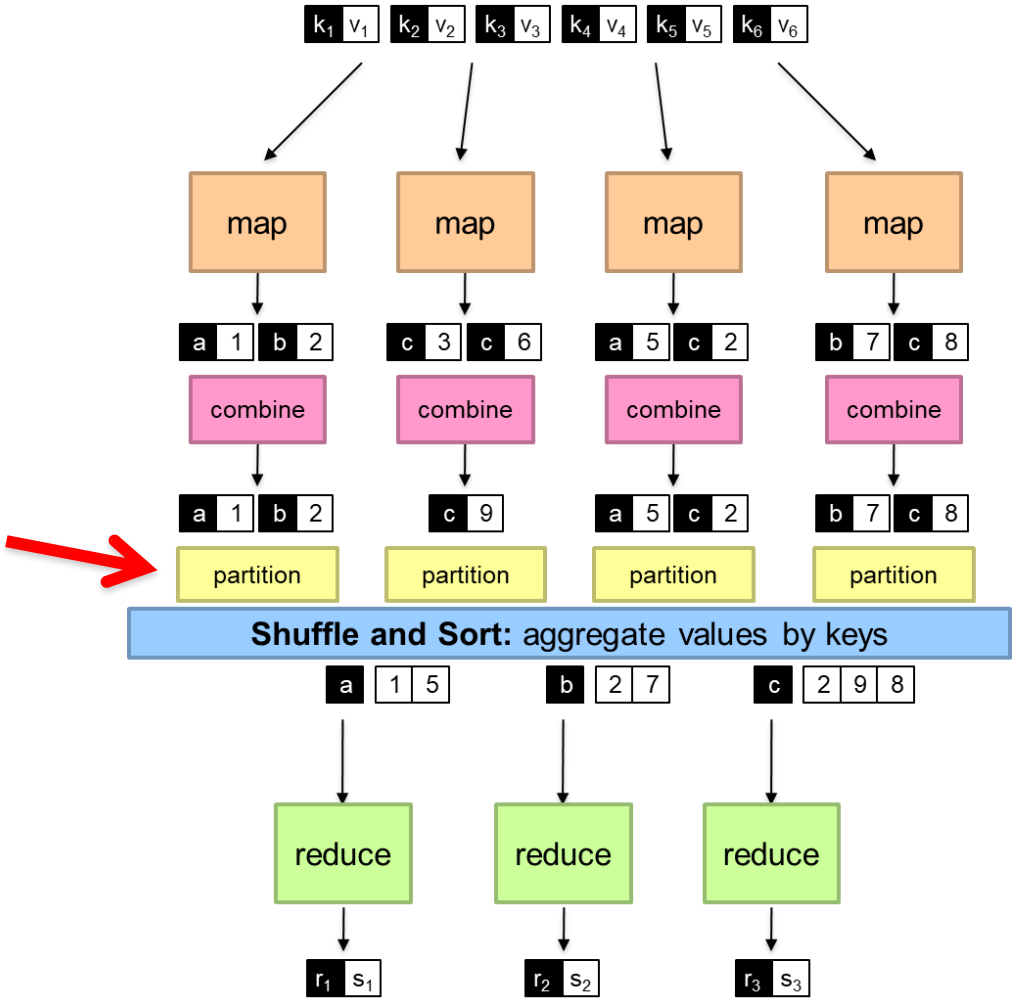
Big Data Architektúrák és Elemző módszerek

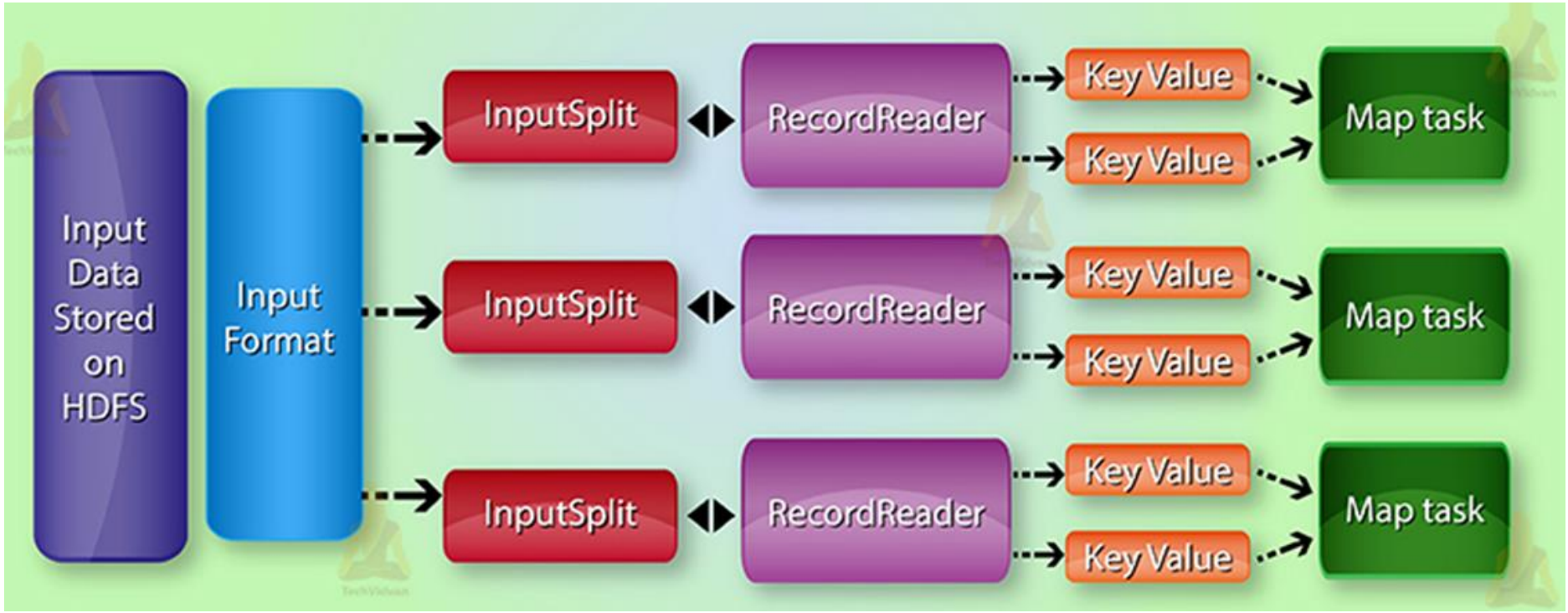
Gombos Gergő, Laki Sándor

MapReduce - Partitioner



- Kulcsok szétosztásáért felel
- Alap beállítása hash alapú, de felül lehet definiálni

MapReduce

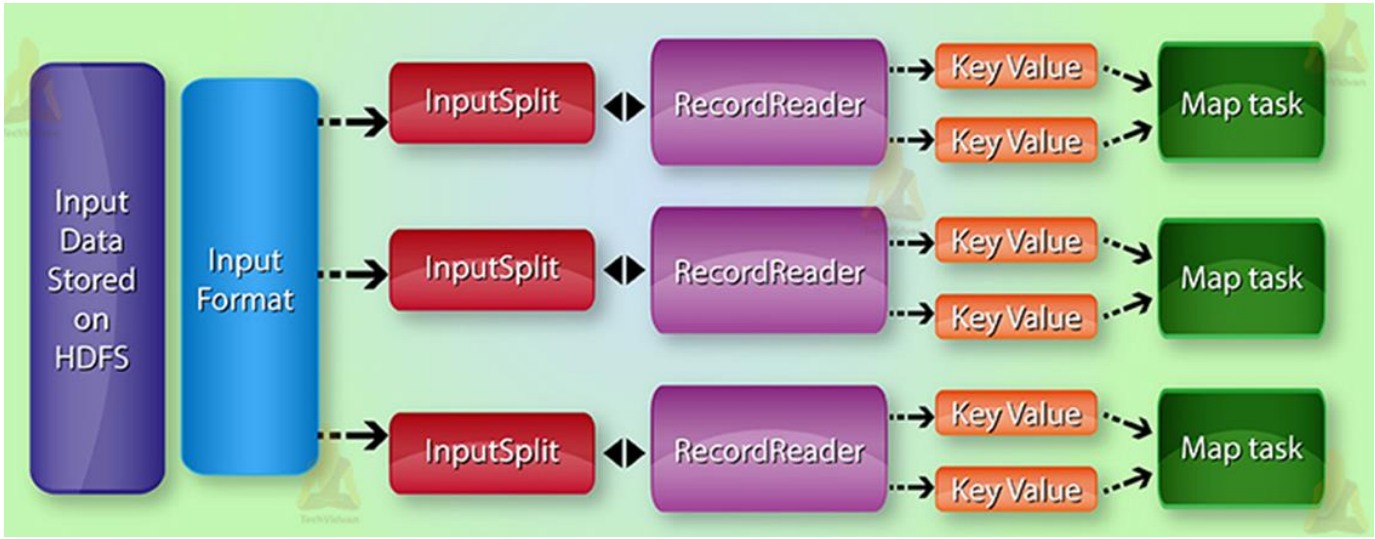


source: https://techvidvan.com/

MapReduce – Input Format

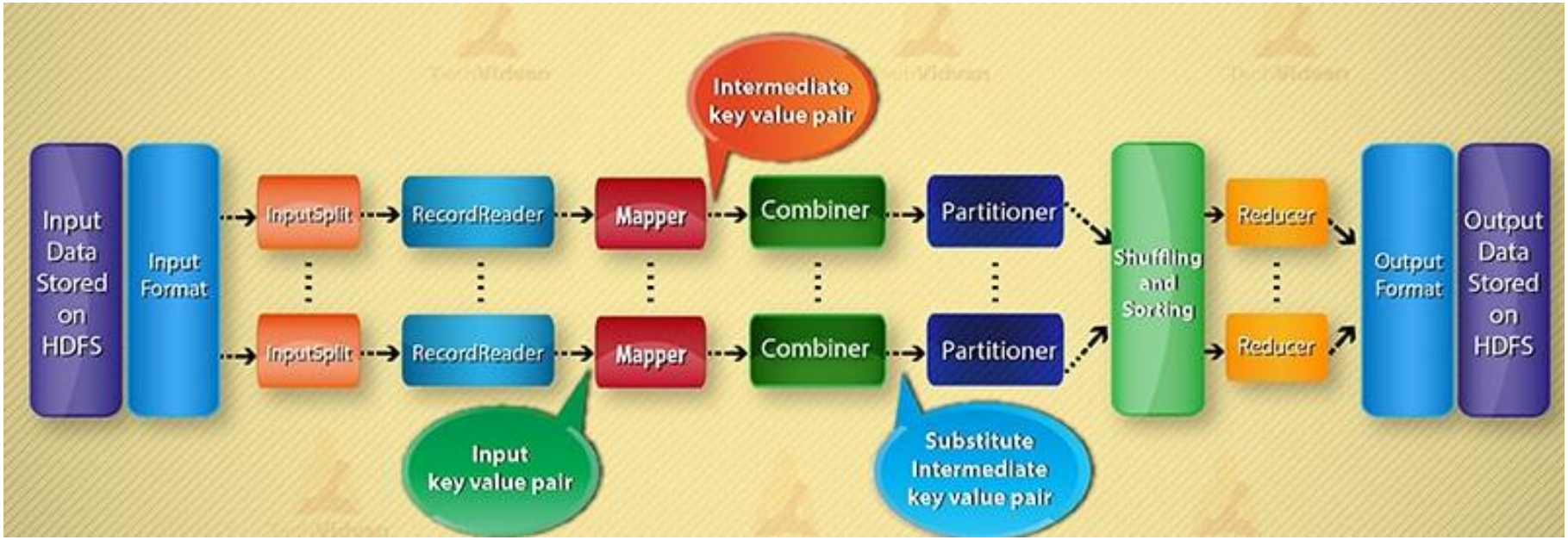
- Leírja a MapReduce Job bementi formátumát
- Alap típusok:
  - **FileInputFormat**: egy vagy több file elérését adja meg
  - **TextInputFormat**: alap beállítás, egy vagy több fájlból sorokat készít (offset a file elejétől, adott sor)
  - **KeyValueTextInputFormat**: hasonló mint a TextInputFormat, csak itt a sorokat tabulátor ('\t') mentén szétválasztja kulcs-értékre
  - **SequenceFileInputFormat**: bináris fájlok olvasása, a kulcs- értékeket a felhasználónka meg kell adni
  - **SequenceFileAsTextInputFormat**: hasonló min az előbbi, csak a kulcsból és a értékből Text típust csinál a toString() metódussal
  - **NlineInputFormat**: olyan TextInputFormat, ahol a map N sort kap meg

MapReduce



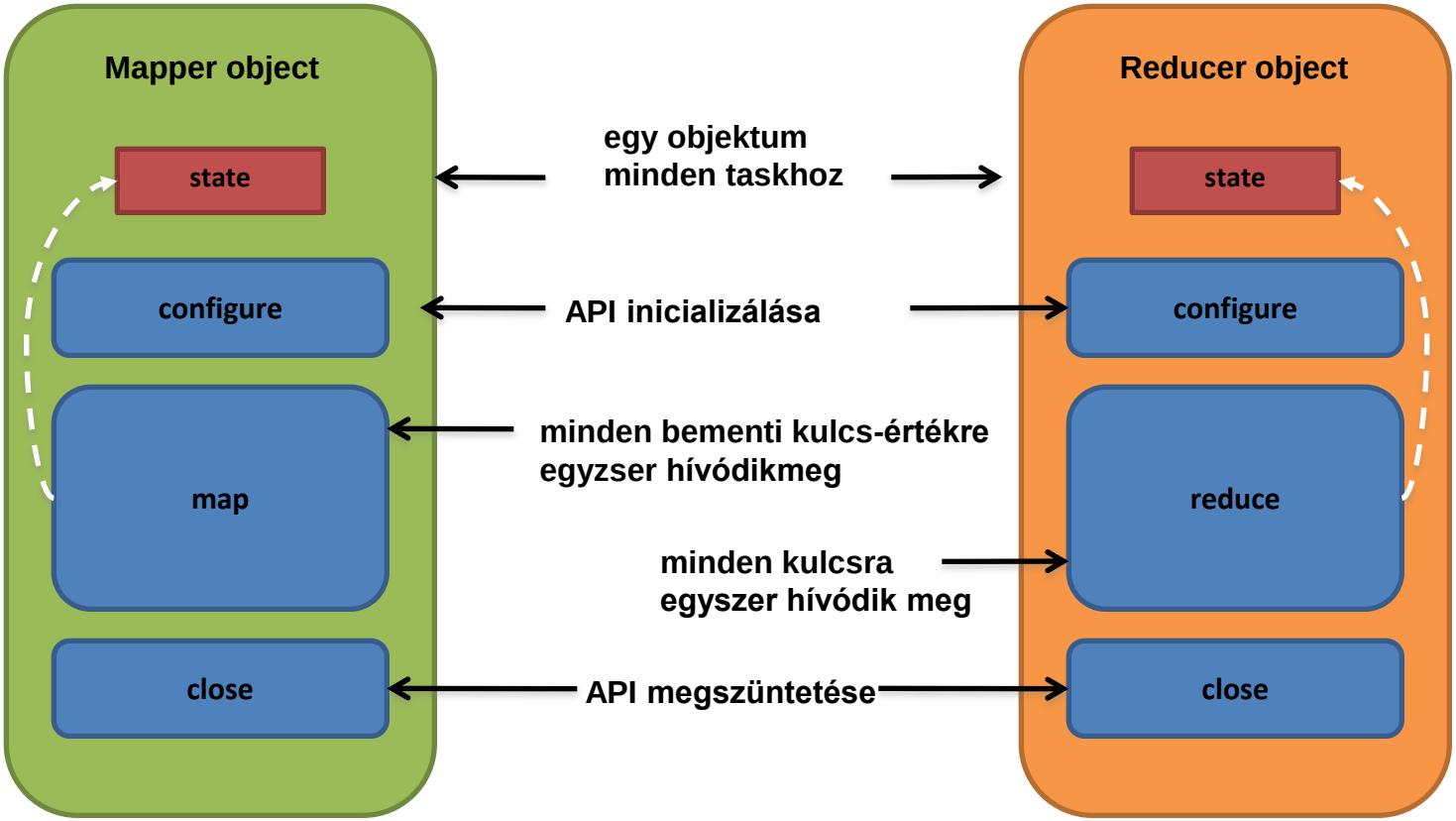
- InputSplit:
  - Logikai megfelelője az InputFormatnak, egy map taskhoz kerülő inputot készíti, a darabolt fájl blok méretének megfelelően
- RecordReader:
  - el kulcs-értéket készít az InputSplit kimenetéből

MapReduce

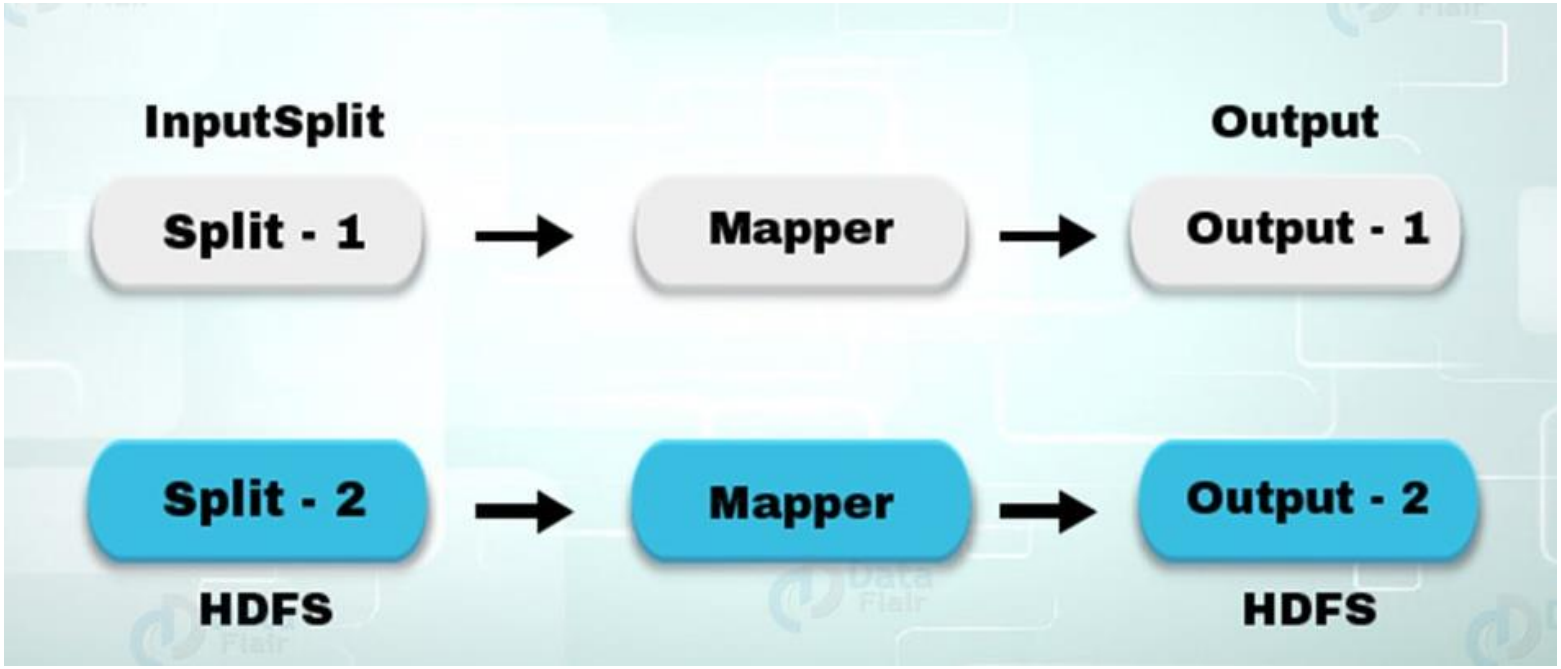


source: https://techvidvan.com/

Mapper, Reducer objects



Map only MapReduce



job.setNumreduceTasks(0)

MapReduce optimalizálás

1. LZO tömörítés használata
   - *mapred.compress.map.output* beállítása true-ra
2. map, reduce taskok száma, blokkméretek beállítása
3. Combiner használata
4. Használjuk újra a Writeable-ket

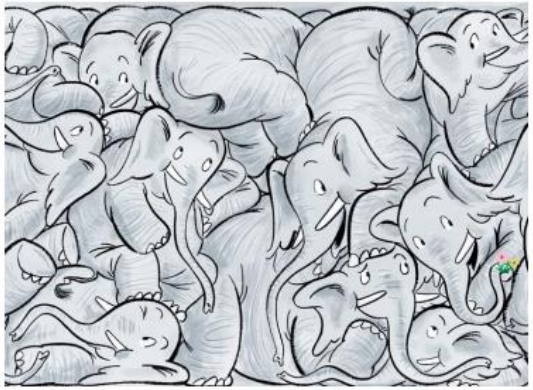
```
1.  public void map(...) {
2.  ...
3.  for (String word : words) {
4.  output.collect(new Text(word), new IntWritable(1));
5.  }
```

```
1.  class MyMapper ... {
2.  Text wordText = new Text();
3.  IntWritable one = new IntWritable(1);
4.  public void map(...) {
5.  ... for (String word : words)
6.  {
7.  wordText.set(word);
8.  output.collect(word, one); }
9.  }
10. }
```

Hadoop / Yarn ütemezés

- Definíció:
  - tenant = Felhasználók/Jobok
  - multi-tenant : Kölünböző Felhasználó / Különböző Jobok
- Probléma:
  - Több felhasználó, több Job-bal

Single tenant



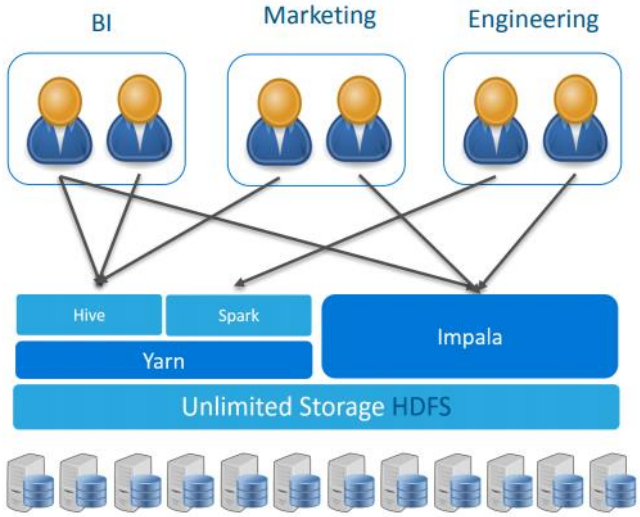
Free-for-all

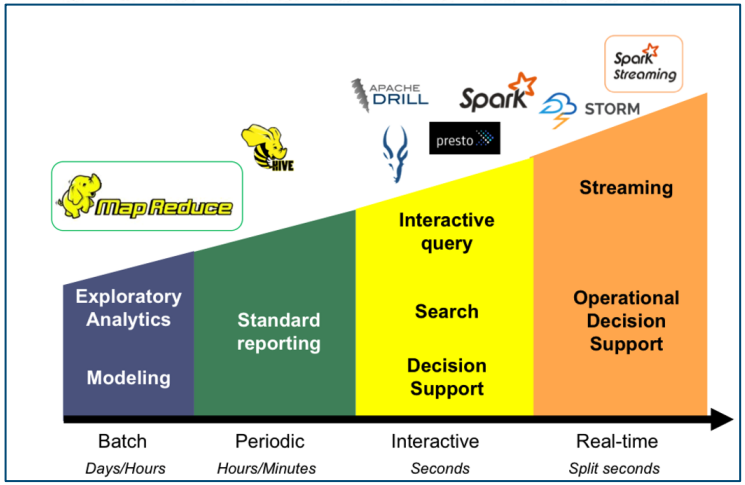
Multi-tenancy

source: Indranil Gupta & Cloudera (Dániel Schöberle)

Hadoop / Yarn ütemezés

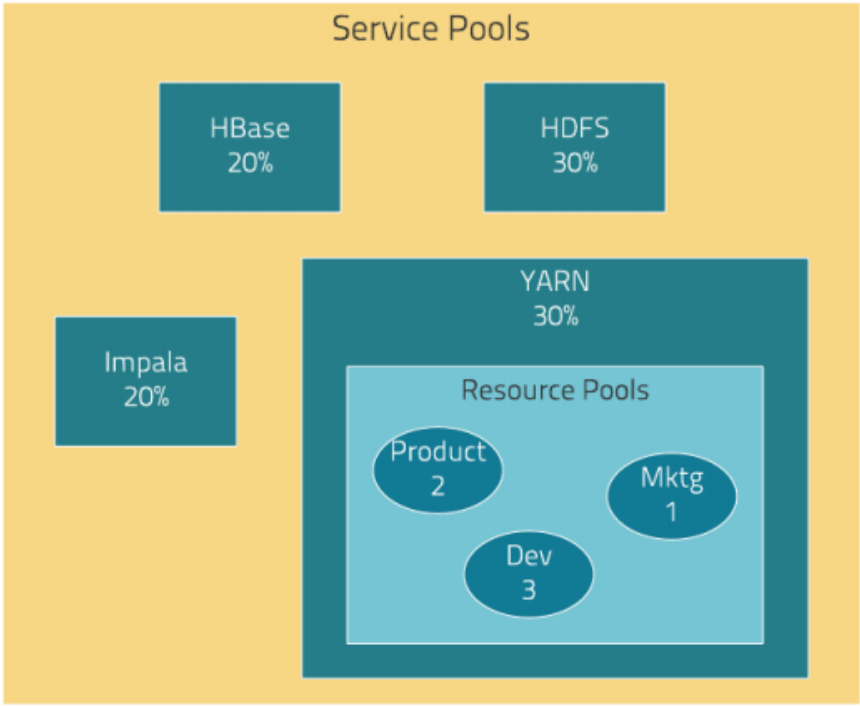
- Cél:
  - Optimális erőforrás kihasználás
  - Megosztani az infrastruktúrát
  - Különböző csoportoknak hozzáférést biztosítani az adatokhoz
  - Támogatni a különböző csoportokat (fejlesztők, data scientistek, elemzők különböző részlegekből)





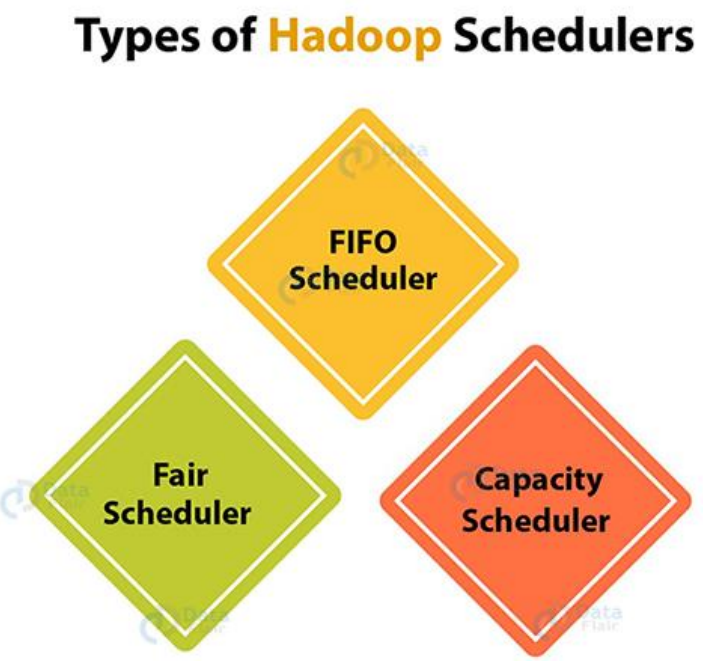
Hadoop / Yarn ütemezés

- Statikus megoldás:
  - Cgroups
  - Yarn-on kívüli erőforrás limitáció
  - Statikus paraméterek:
    - CPU megosztás
    - I/O súly
    - Memória



Hadoop / Yarn ütemezés

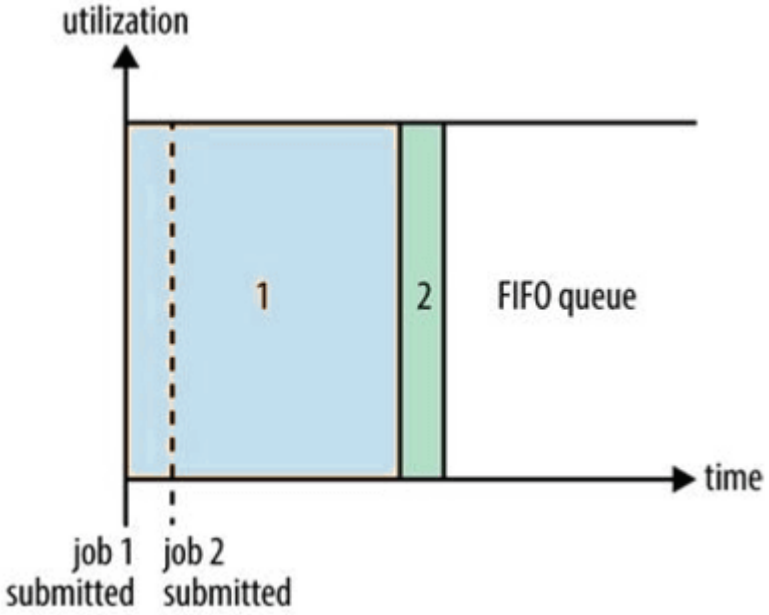
- Dinamikus megoldások:
  - Capacity vagy Fair scheduler
  - Erőforrások dinamikusan állítódnak a sorok (queue) alapján
  - Erőforrások meg vannak osztva a sorok között. Ha egy sor nem használja az erőforrást, akkor a többiek használhatják.
  - Sorok hozzáférését lehet szabályozni felhasználó / csoport szinten
  - Működik: MapReduce, Spark, Hive,….



source: https://data-flair.training/blogs/hadoop-schedulers/ & Cloudera

FIFO Scheduler

- First In First Out (FIFO)
- Jobok ütemezése az indításuk sorrendjében
- Minden Job az egész clustert használja, nem hatékony
- Egy lassú Job kiéhezteti a többit
- Előnye, hogy nem kell konfigurálni

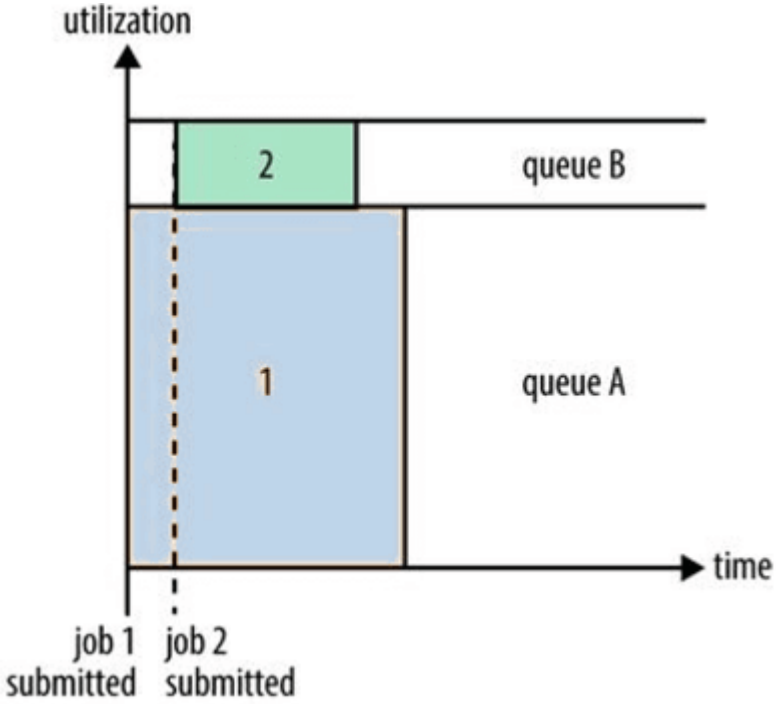



```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fifo.FifoScheduler</va
lue>
</property>
```

source: http://www.corejavaguru.com/bigdata/hadoop-tutorial

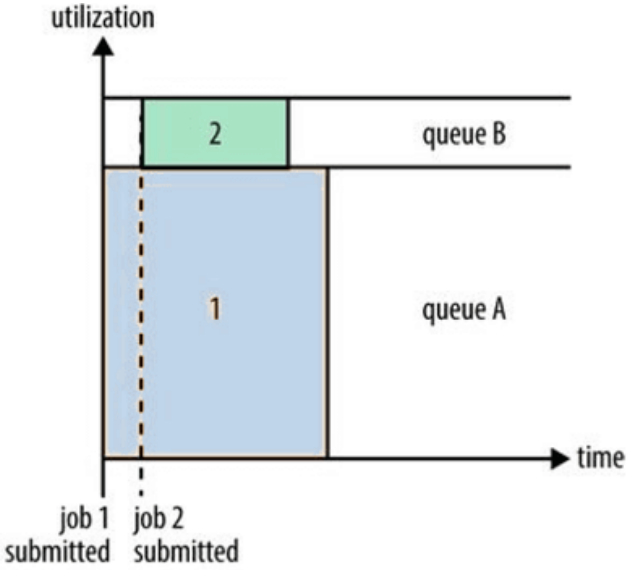
Capacity Scheduler

- Egyszerű klaszter megosztás
- Százalékos formában lehet sorokat létrehozni
- A megosztás garantálja a minimum erőforrást
- A soroknak lehetnek al-sorai is
- Lehet konfigurálni, hogy mennyi legyen a maximális kapacitás is
  - pl: Ha A sor üres, akkor B sor használhatja a teljes klasztert



Capacity Scheduler

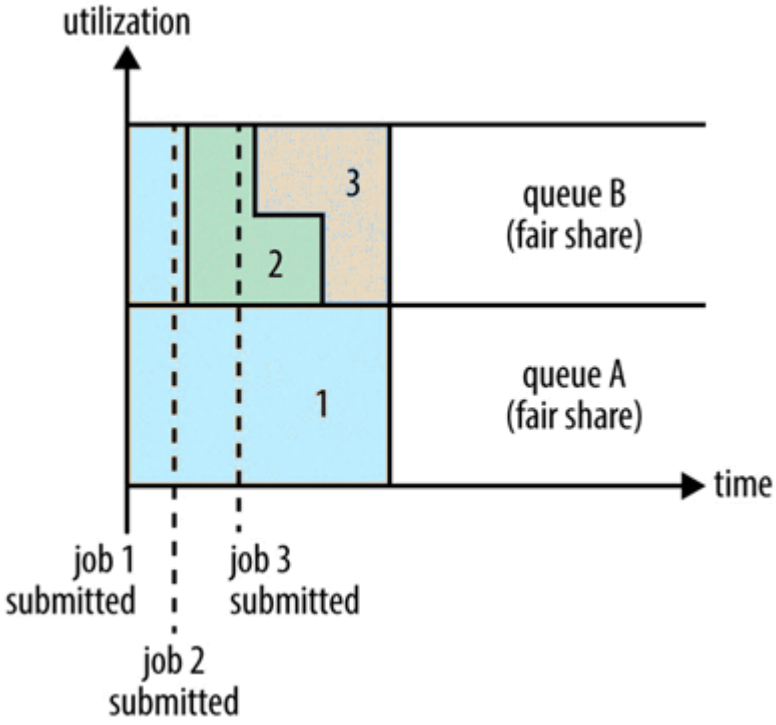
- root queue (100%)
  - prod (40%)
  - dev (60%), max: (75%)
    - job1 (50%)
    - job2 (50%)




```xml
<?xml version="1.0"?>
<configuration>
  <property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>prod,dev</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.dev.queues</name>
    <value>job1,job2</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.prod.capacity</name>
    <value>40</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.dev.capacity</name>
    <value>60</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.dev.maximum-capacity</name>
    <value>75</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.dev.job1.capacity</name>
    <value>50</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.dev.job2.capacity</name>
    <value>50</value>
  </property>
</configuration>
```

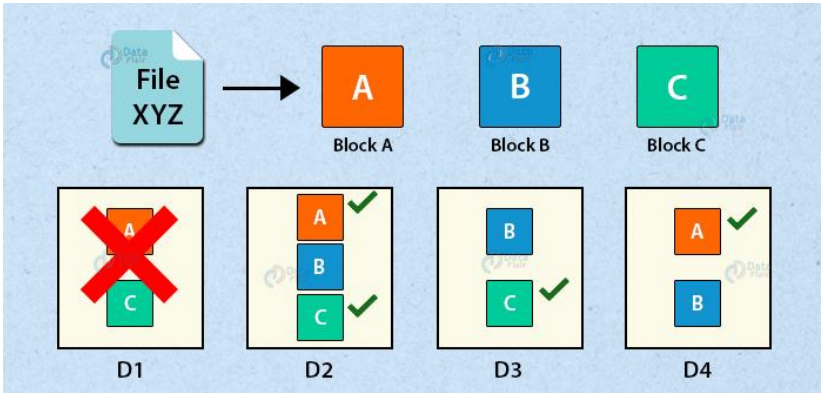
Fair Scheduler

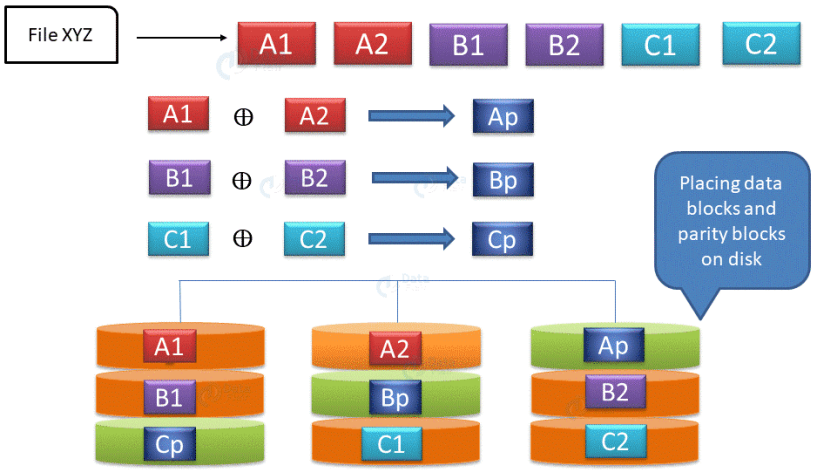
- Az indított Job-ok fair módon osztoznak a klaszter erőforrásain
- Legtöbbször használt ütemező
- Előnye: mindig van egy minimum erőforrás, amit megkap a Job
- A beküldött Job nem indul addig, amig nem szabadul fel erőforrás (nem futnak le taskok)
- Van lehetőség 'preemption' opcióra, ami lehetőséget a schedulernek, hogy töröljön futó containereket



HDFS hibakezelés

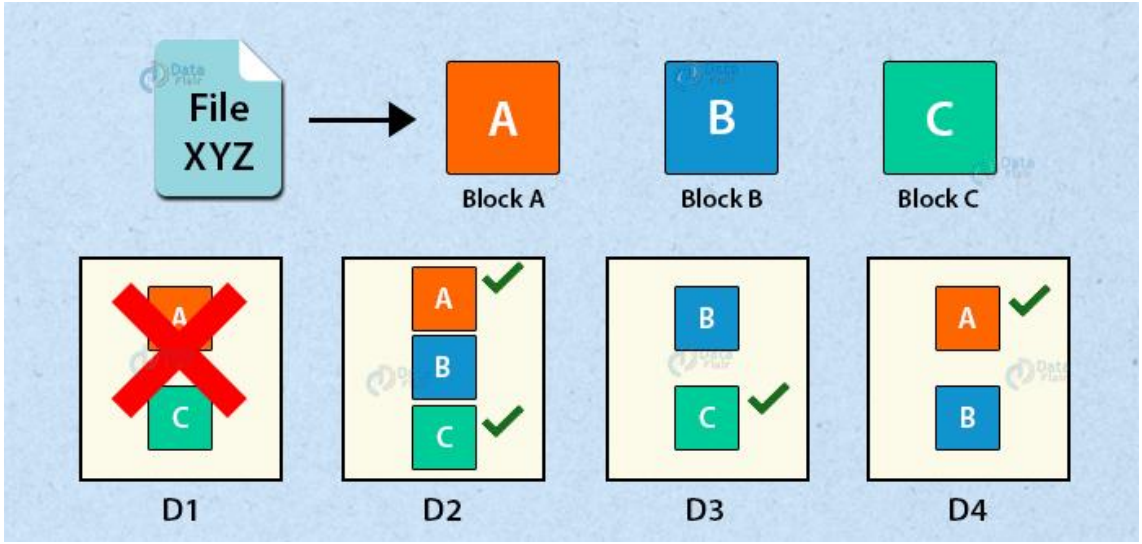
- Hadoop v2
  - replika
- Hadoop v3
  - erasure kód





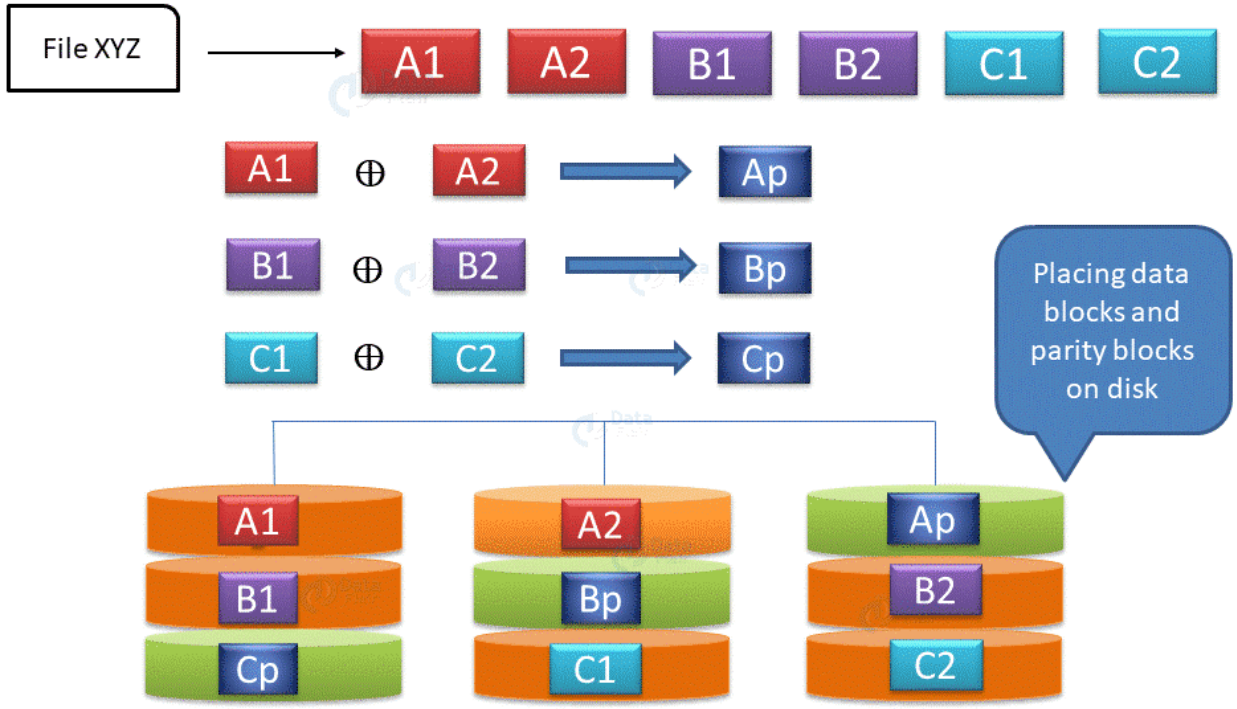
HDFS replika - hibakezelés

- Ha egy DataNode kiesik az ott tárolt blockok más Datanode-okon még elérhetőek
- Ha a DataNode nem lesz újra elérhető, akkor a NameNode új replikákat készít a hiányzó replikák helyett



HDFS Erasure Coding

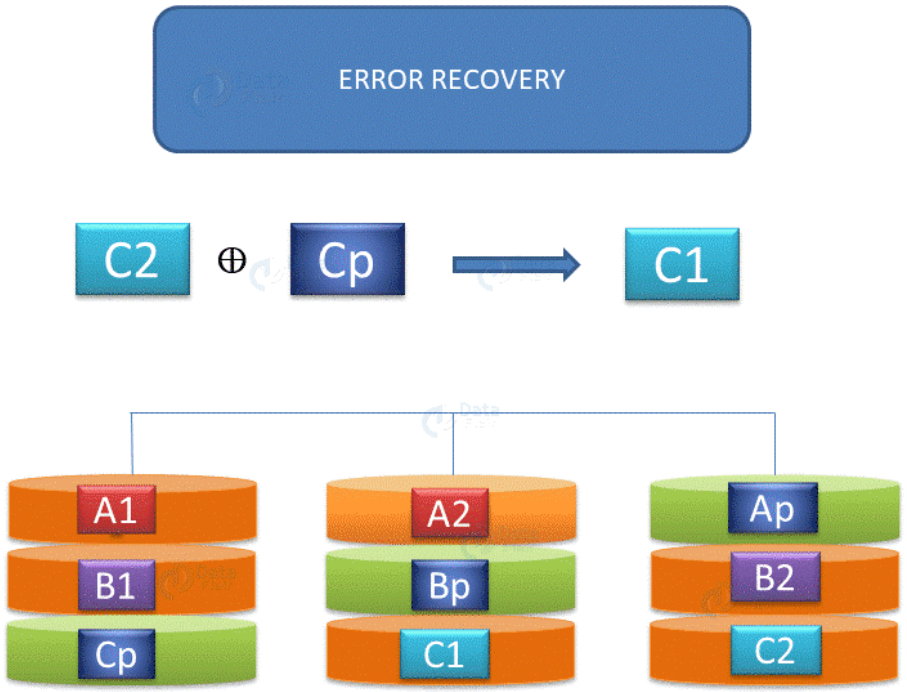
- Az input filet blokkokra bontjuk
- A blokkból paritás értéket számolunk (XOR, Reed-Solomon)



source: https://data-flair.training/blogs/hadoop-hdfs-erasure-coding/

HDFS Erasure Coding

- Ha C1 blokk elveszett, akkor a C2 és a paritás alapján vissza tudjuk állítani
- Előny: kevesebb tárhely, csak 50%-os overhead
- Hátrány: a kód kiszámítás miatt további CPU és hálózati költség



MapReduce hibakezelés

- MapReduce v1:
  - Task hiba
  - TaskTracker hiba
  - JobTracker hiba
- Yarn hiba
  - Task hiba == MapReduce v1 Task hiba
  - AppMaster hiba
  - ResourceManager hiba

source: Jari-Pekka Voutilainen

MapReduce v1 (Task hiba)

- TaskTracker hibásnak jelöli a taskot,
  - ha kivétel van a map vagy a reduce folyamatban
  - ha a JVM leállt
  - ha nem a task nem küld státusz állapotot
    - ekkor a TaskTracker lövi le a JVM-t
- JobTracker újraindítja a taskot egy másik TaskTrackeren ha lehetséges
- Ha 4 vagy többször nem sikerül lefuttatni a taskot, akkor az egész Job hibásnak lesz jelölve

MapReduce v1 (TaskTracker hiba)

- A JobTracker heartbeat üzenetek alapján látja, hogy egy adott TaskTracker működik
- Ha egy TaskTracker nm működik, akkor a JobTracker másik TaskTrackerre ütemezi be a futó és már lefutott taskokat
- A JobTracker figyeli hogy a Taskracker mennyi taskot nem tudott elvégezni
- Ha a hibák száma elér egy limitet, akkor a TaskTracker feketelistára kerül és a JobTracker nem ad neki feladatot
- A kizárás egy idő után (pl: naponta) elévül és TaskTrackernek újra kap taskokat

MapReduce v1 (JobTracker hiba)

- MapReduce v1 kritikus pontja
- Minden futó Job elveszik
- Minden futó Jobot újra kell indítani

Yarn hiba (AppMaster hiba)

- ResourceManager figyeli az AppMaster-t
- Hiba esetén egy új containerben indít egy új AppMastert
- Ha a kliens nem éri el az AppMastert, akkor megkérdezi az új címét a ResourceManager-től

Yarn hiba (ResourceManager hiba)

- ResourceManager checkpointokat tárol egy perzisztens háttértáron (HDFS, ZooKeeper)
- Hiba esetén:
  - adminisztrátor indít egy új ResourceManager-t
  - egy standby ResourceManager átveszi a szerepet

Hadoop limitációk

1. Probléma kis fájlokra (< blokk méret)
   - A HDFS nagy fájlokra lett tervezve, kis fájlok beolvasására nem hatékony
   - A NameNode több adatot tárol mint maga a fájl
2. Lassú feldolgozás
   - Taskok elindítása költséges, sok ideig tart az inicializálás
   - Részeredmények más másolása a taskok között lemezen és hálózaton történik

source: https://data-flair.training/blogs/13-limitations-of-hadoop/

Hadoop limitációk

3. Csak batch feldolgozás van
   - a Hadoop nem hatékony a stream jellegű adatok feldolgozására, csak a statikusan tárolt adatokra
4. Nem hatékony iteratív feldolgozásra
   - Hadoopban iterációt Jobokkal lehet megvalósítani
   - 1 Job = 1 iteráció
   - Költséges az adatok mozgatása a Jobok között

Hadoop limitációk

5. Nehezen használható
   - Nincs interaktív mód
   - Nehezen debug-olható a kód
6. Biztonság
   - Az adat nincs kódolva a közös tárterületen
   - Nincs megfelelő user-kezelés a HDFS-en
7. Nincs cache

Apache Pig

- Célja:
  - könnyebb felületet biztosítani a MapReduce technikához
- History
  - 2006 – kutatási projekt a Yahoo-nál
  - 2007 – opensource lett
  - 2008 – első release megjelenése

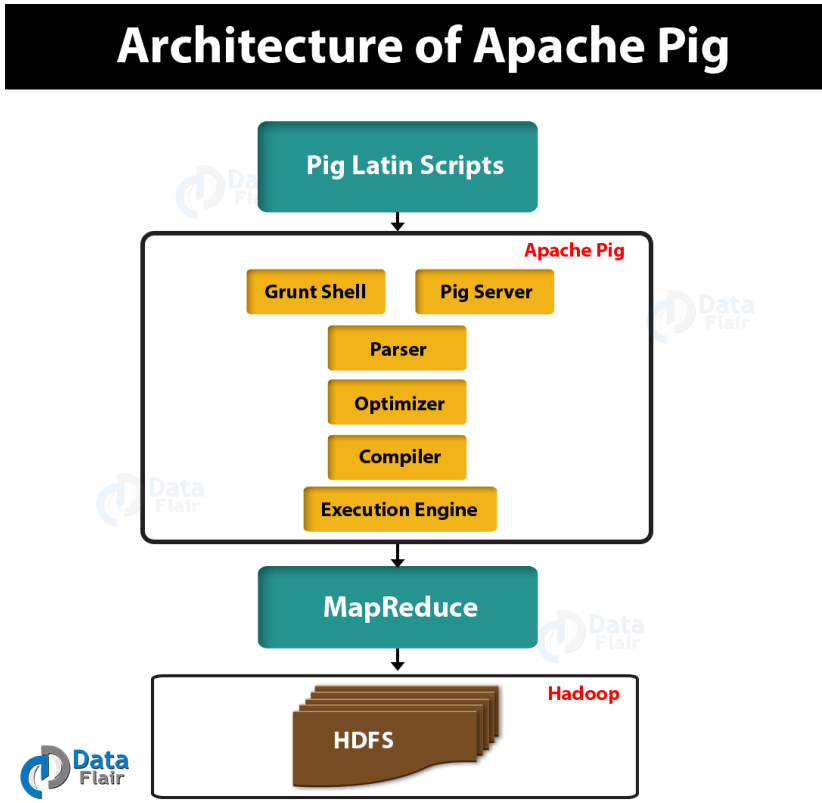
source: https://data-flair.training/blogs/hadoop-pig-tutorial/

Apache Pig

- Tulajdonságai:
  - Java helyett Pig Latin nyelven kell a scripteket megfogalmazni
    - Pig Engine átalakítja a scriptet MapReduce jobok-ká
  - Pig Latin hasonlít az SQL nyelvhez
    - van benne: join, filter, sort,…
  - Van optimalizáló lehetőség, amely a taskok sorrendjét optimalizálja
  - Kiegészíthető
    - Saját függvények írhatóak hozzá, különböző nyelveken, amelyet aztán használhatunk a scriptünkben

Apache Pig

- Komponensek:
  - Parser: szintakszis és típus ellenörzés, szkript átalakítása az optimalizálónak
  - Optimizer: a logikai tervet optimalizálja, hasonlóan mint a relációs adatbázisoknál
  - Compiler: A logikai tervből MapReduce Jobokat készít
  - Execution Engine: A MapReduce Jobokat indítja el amegfelelő sorrendben



Apache Pig

- Pig Latin adatmodell
  - Atom: atomi típusok: int, float, long, double, char array, byte array pl: "elte", 1635
  - Tuple: rendezett halmaz pl ("elte",1635)
  - Bag: tuple-k halmaza pl: {("elte",1635),("beac", 1898)}
  - Map: kulcs-érték párok halmaza pl [nev#elte, alapitva#1635]

Apache Pig

- Pig futtatási lehetőségek:
  - Non-iteraktív vagy script mód
  - Grunt shell vagy interaktív mód
  - Embedded, ami JDBC-n keresztül érhető el

Apache Pig

- Hello World

```
lines = LOAD '/user/hadoop/HDFS_File.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
DUMP wordcount;
```

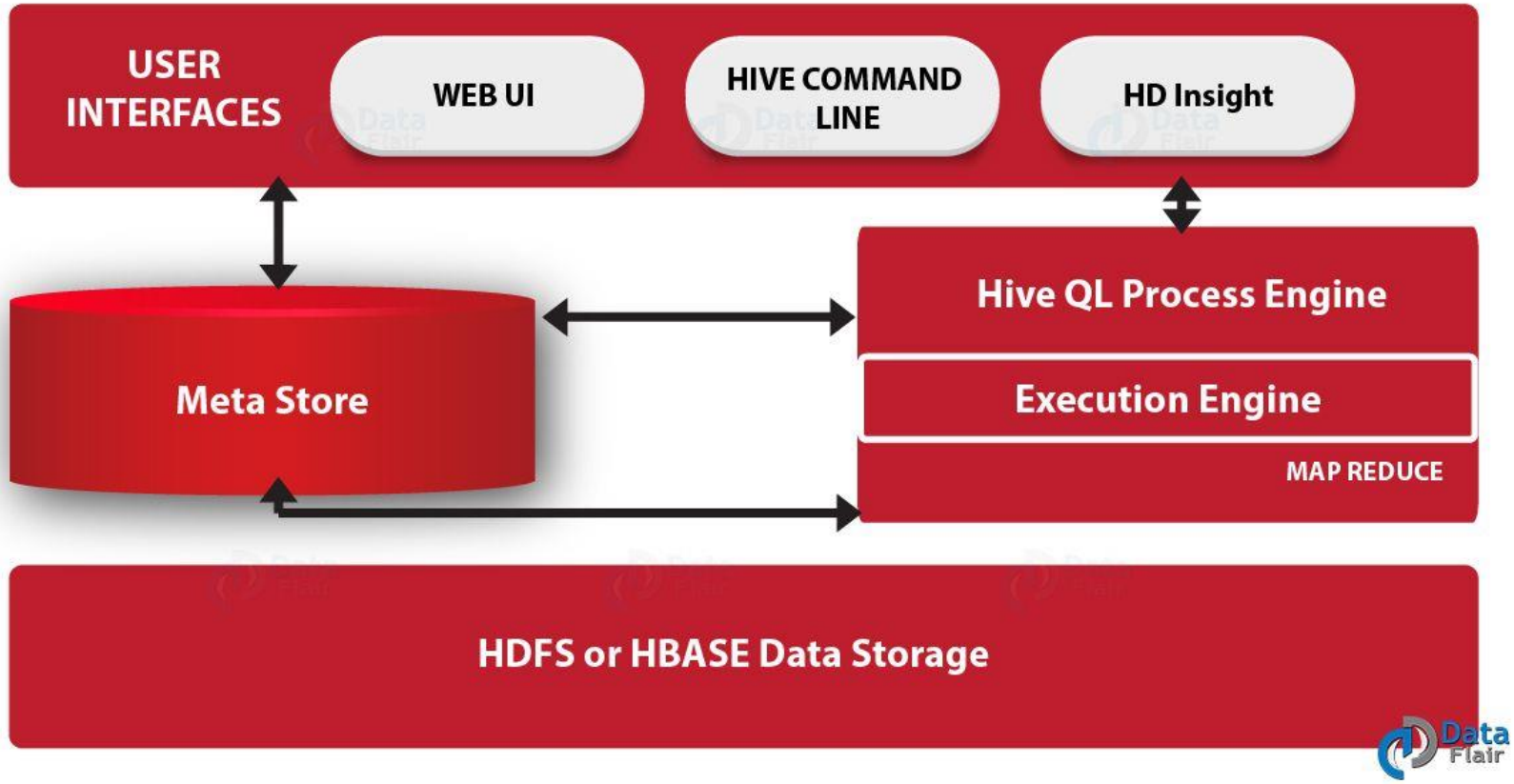
Apache Pig

- Limitációk
  - hibaüzenetek nem beszédesek
  - lassan fejlődik (évente 1-2 release, 2017-ben volt az utolsó)
  - lassú futtatás a MapReduce Jobok miatt

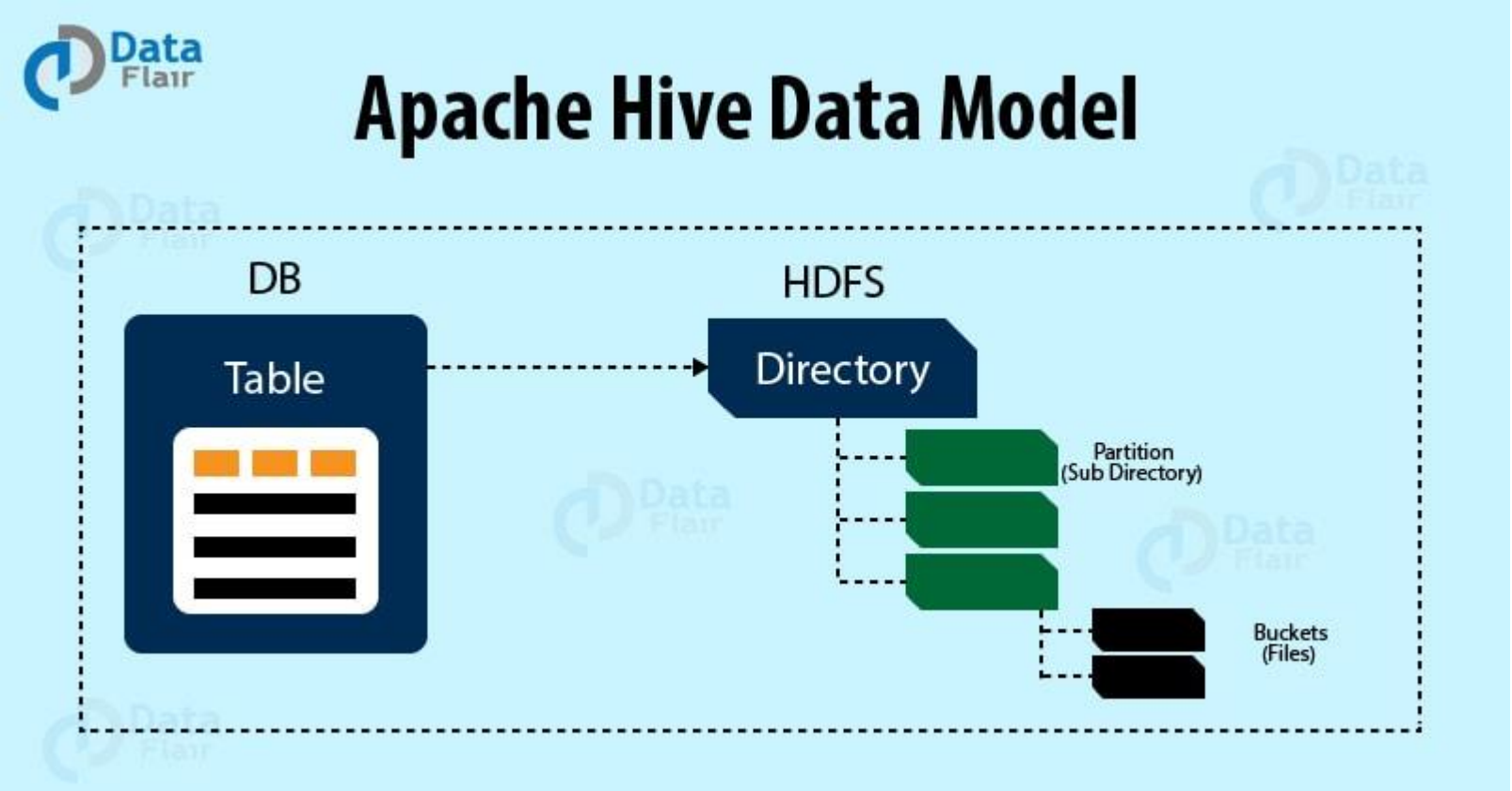
Hive

- Mi a Hive?
  - adattárház Hadoop alapokra
  - lekérdezéseket SQL-szerűen lehet megfogalmazni
    - HiveQL (HQL)
    - A lekérdezések MapReduce Jobok-ként futnak
- Története
  - Facebook fejlesztette
  - Később több cég is beszált: IBM, Yahoo, Netflix, Amazon

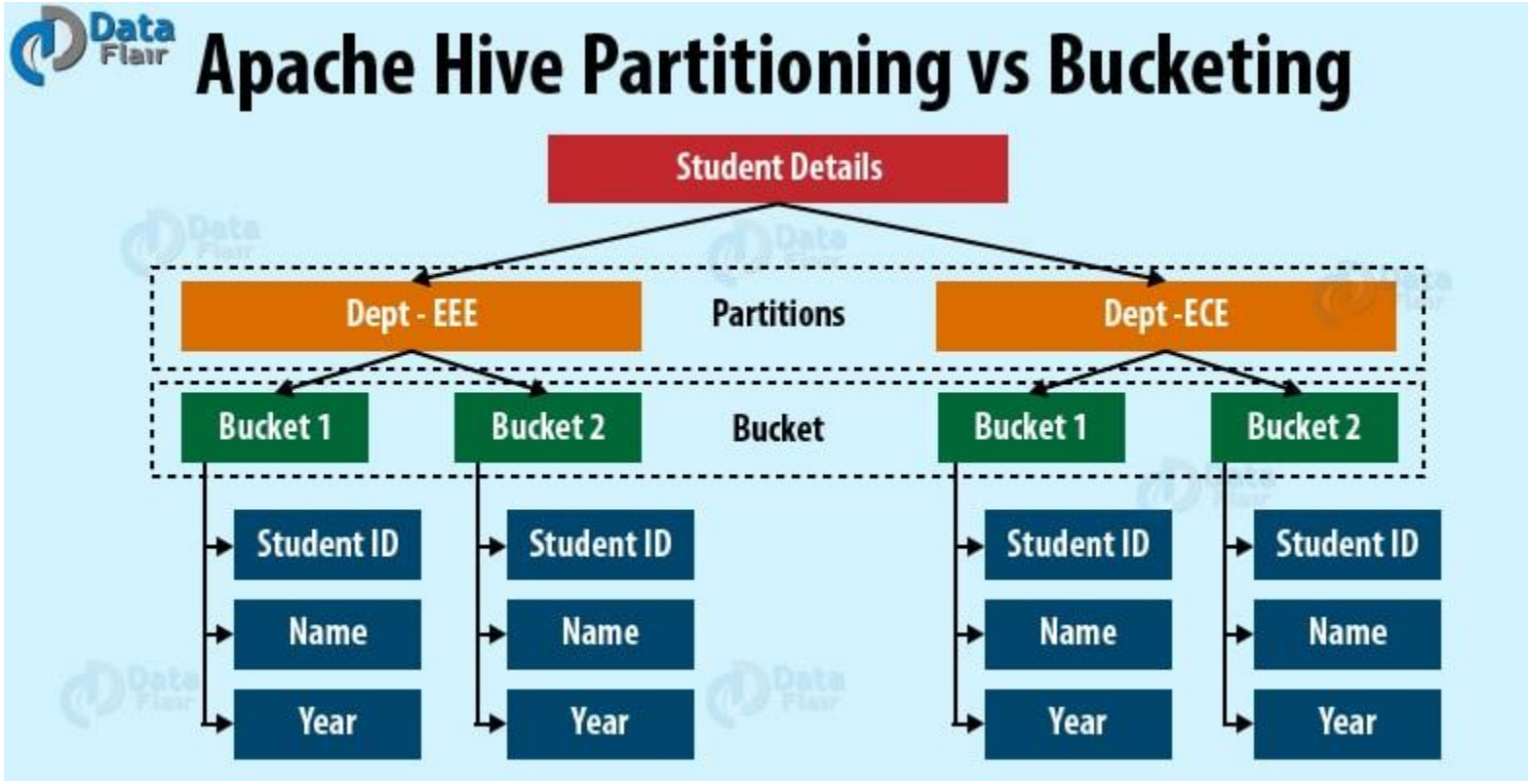
Hive


USER INTERFACES
WEB UI
HIVE COMMAND LINE
HD Insight
Meta Store
Hive QL Process Engine
Execution Engine
MAP REDUCE
HDFS or HBASE Data Storage
Data Flair

Hive


Data Flair
Apache Hive Data Model
DB
Table
HDFS
Directory
Partition (Sub Directory)
Buckets (Files)

Hive


Data Flair
Apache Hive Partitioning vs Bucketing
Student Details
Dept - EEE
Partitions
Dept -ECE
Bucket 1
Bucket 2
Bucket
Bucket 1
Bucket 2
Student ID
Name
Year
Student ID
Name
Year
Student ID
Name
Year
Student ID
Name
Year

Hive

Table-1 Hive DDL commands

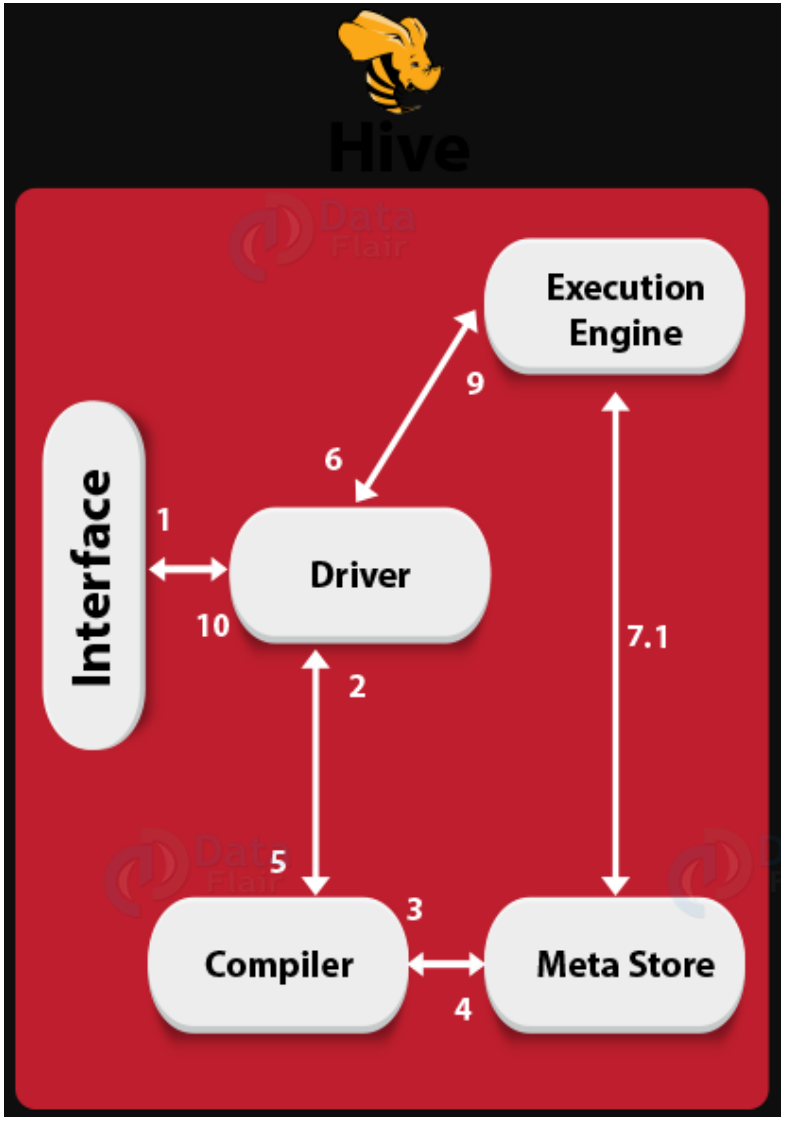
DDL Command	Use With
CREATE	Database, Table
SHOW	Databases, Tables, Table Properties, Partitions, Functions, Index
DESCRIBE	Database, Table, view
USE	Database
DROP	Database, Table
ALTER	Database, Table
TRUNCATE	Table

The various Hive DML commands are:

1. LOAD
2. SELECT
3. INSERT
4. DELETE
5. UPDATE
6. EXPORT
7. IMPORT

Hive

1. Lekérdezés küldése a Drivernek
2. Compiler értelmezi és ellenőrzi a lekérdezést
3. 4. Compiler metinformációt kérdez le a Meta Storeból
5. A lekérdezési tervet elküldi a Compilernek
6. Execute Engine indítja és lefuttatja a MapReduce Jobokat



Hive

- Limitációk
  - Valós idejű lekérdezésekre nem képes
  - nincs sor szintű frissítés
  - Hive lekérdezések általában lassúak

Köszönöm a Figyelmet!