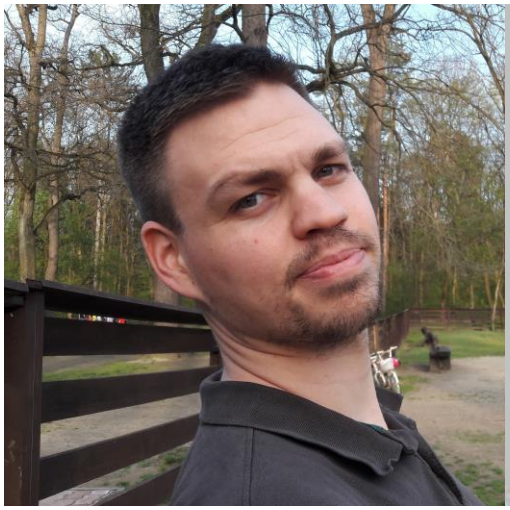



Dr. Hegedűs Péter, Dr. Ferenc Rudolf

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

A MapReduce programozási modell

Összefoglalás

Ez az olvasólecke bemutatja, hogy mi is az a MapReduce programozási modell, illetve hogyan működik ez a programozási modell a gyakorlatban az Apache Hadoop ökoszisztémán belül. Megtudhatjuk, hogyan kell egyszerű MapReduce algoritmust írni Java nyelven, valamint hogyan lehet azt lefordítani és futtatni Hadoop környezetben.
A lecke fejezetei:

- 1. fejezet: **A MapReduce programozási modell alapjai (olvasó)**
- 2. fejezet: **MapReduce folyamat működése lépésről lépésre, valamint implementációi (olvasó)**
- 3. fejezet: **MapReduce megvalósítása az Apache Hadoop ökoszisztémában (olvasó)**

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

📖 1. fejezet
A MapReduce programozási modell

A MapReduce egy programozási modell [9] és ahhoz tartozó implementáció, mely támogatja a batch alapú elosztott feldolgozást adatintenzív rendszerek esetén. Ezáltal kifejezetten alkalmas nagy mennyiségű adat párhuzamos feldolgozására, ahol az adatok akár többszáz különböző számítógépen lehet elosztva. További jellemzője a magasfokú hibatűrés, valamint a funkcionális programozás támogatása.

A koncepció 2004-ben jelent meg (Google), 2012-től az elosztott adatfeldolgozás „de facto” szabványává vált. Egy MapReduce programnak két fő fázisa van (két függvény):

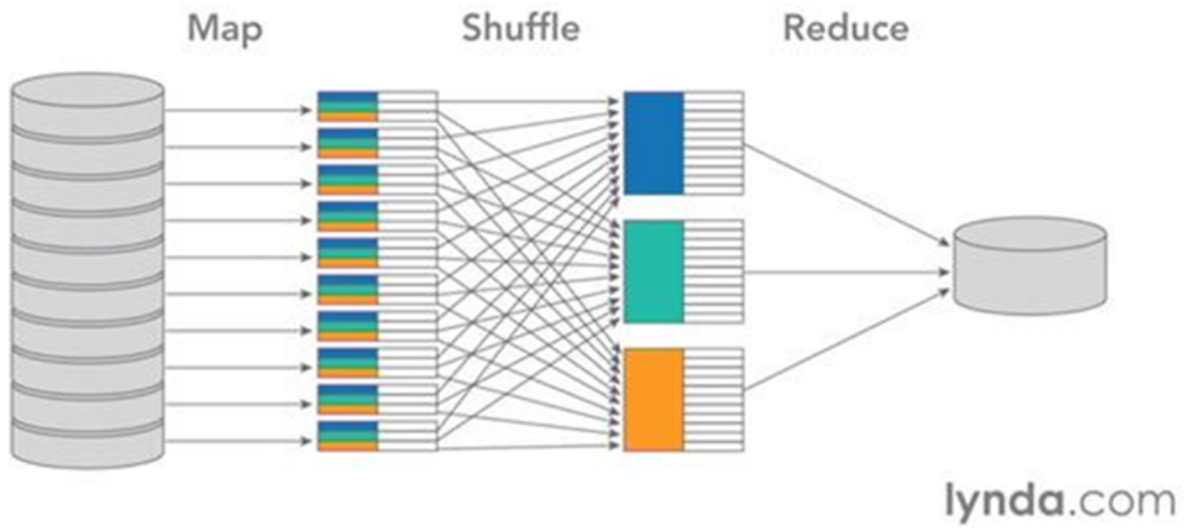
- **Map**: adatok szétválogatása és összerendelése
- **Reduce**: adatok átrendezése és tömörítése

Egy MapReduce program esetén minden map és reduce függvény elosztott módon, különböző node-okon fut (elosztott adattárolásra használt cluster csomópont, pl. Hadoop node). Minden függvény a teljes adathalmaz csak egy kis szeletén dolgozik, így a terhelés eloszlik a cluster-en belül. A map fázis a következő lépésekből tevődik össze:

1. Adatok betöltése, szűrése
2. Adatok transzformálása kulcs-érték párokká

Míg a reduce fázis lépései a következők:

1. Egy-egy map feladat kimenetének feldolgozása
2. Köztes adat átmásolása



A MapReduce folyamata az adatok különböző adat csomópontokról történő betöltésével kezdődik. Az adatok tárolódhatnak egy, de akár több különböző node-on is. Az adat rekordok annyi halmazra kerülnek szétosztásra, ahány map node elérhető a cluster-en belül, ezt a lépést `split`-nek nevezik. A map feladatok párhuzamosan lefutnak, és mindegyik a számára kijelölt rekordokból egy kulcs-érték párost képez.

Az ábrán az azonos kulcsok azonos színnel vannak jelölve. A map feladat eredményeként előálló kulcs-érték párokat egy ún. `shuffle` lépés során átrendezzük. Minden azonos kulcsú elem egy számítási node-ra kerül, majd az azonos kulcsú elemekre lefut a reduce feladat. Az egyes reduce feladatok által adott kimenetek együttesen alkotják a MapReduce folyamat végeredményét (`combine` lépés).

📖 2. fejezet
A MapReduce folyamat részletei és megvalósításai

Az első fejezetben áttekintettük, hogy magas szinten hogyan néz ki egy MapReduce folyamat. Tekintsük most újra át a map és reduce fázisokat kicsit részletesebben.

Map fázis:

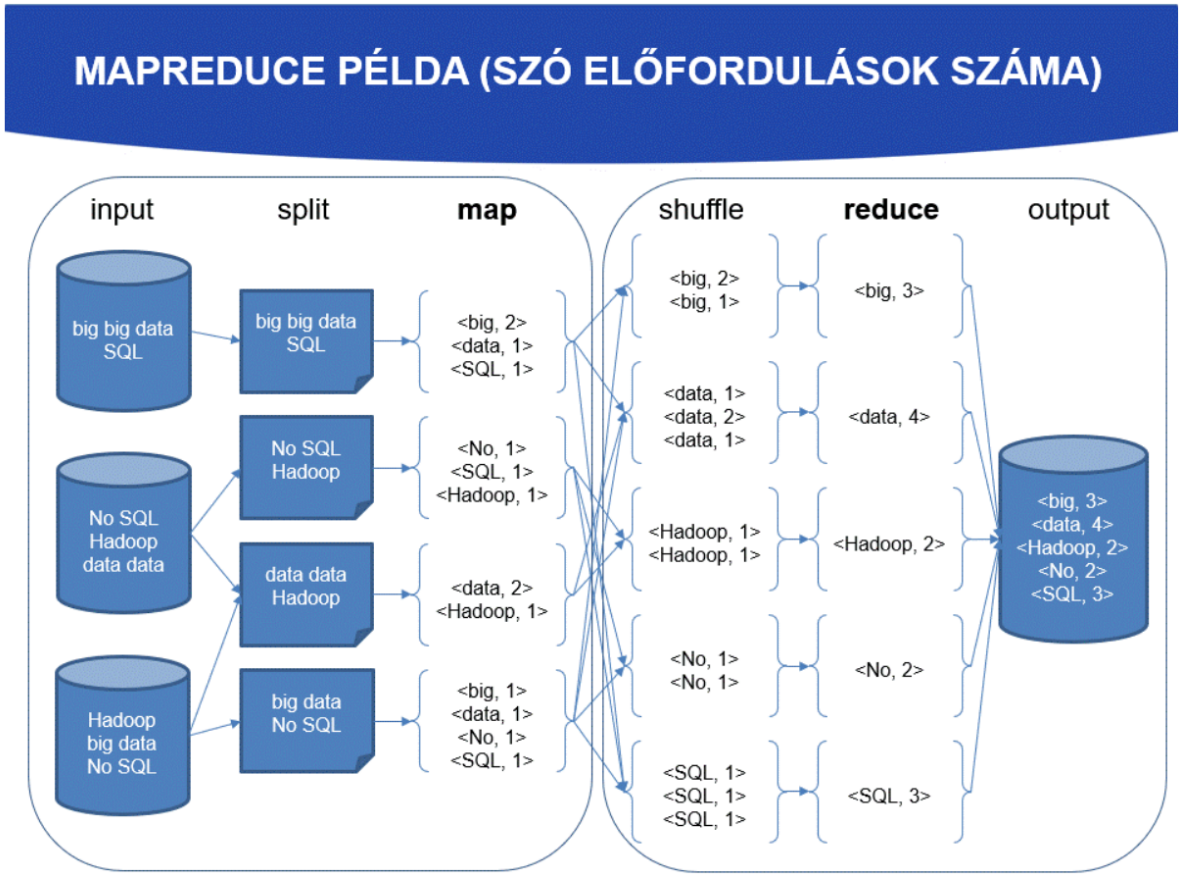
- Input olvasás (record reader)
- Szétosztás (split)
- Map függvény meghívása

Reduce fázis

- Map eredmények átrendezése (shuffle/sort)
- Reduce függvény meghívása
- Eredmény összeállítása és kiírás (combine/output format)

```
Megjegyzés: az egyes konkrét MapReduce megvalósítások más/eltérő feldolgozó
lépéseket is implementálhatnak!
```

Lássunk egy példát egy egyszerű MapReduce feladatra. Adott egy szövegrészlet, és a feladatunk az, hogy összeszámoljuk az egyes szavak előfordulásainak a számát ezen szövegen belül. A példa kedvéért feltételezzük, hogy a teljes szöveg nem egyben (egy adat node-on) kerül tárolásra, hanem szétosztva a klaszteren belül több csomópontra. Ennek megfelelően egy szó összeszámláló MapReduce program az alábbi módon futna le.



Az adatok betöltése után azok szétosztásra kerülnek, ez az ún. `split` lépés. A `split` során előálló adatrészletek száma nem kell, hogy megegyezzen az adatokat tároló csomópontok számával, konfigurálható. Triviális esetben egy darab adat képződik `split` során, ám ekkor elveszítjük a párhuzamos futtatás adta előnyöket, hiszen minden `split`-hez készül egy map feladat, amelyek párhuzamosan futhatnak. A programozó által biztosított map függvény meghívásra kerül a `split` során keletkezett adat rekordokra, ami növeli a hatékonyságot, hiszen a rekordok száma egy-egy `split` -en belül sokkal kisebb, mint a teljes adathalmaz, ráadásul az egyes map hívások így párhuzamosan történhetnek. Ezáltal könnyen megvalósítható a terhelés elosztás is, az egyes map folyamatokat más és más számítási node is végezheti. Vigyázni kell azonban, mert a túl kis `split` méret sem előnyös, hiszen akkor a szétosztás/összegyűjtés overhead-je lesz meghatározó, és performancia növekedés helyett inkább lassulást okoz.

A map hívás eredményei köztes eredmények, amelyek a reduce függvény bemenetét képezik. Fontos megjegyezni, hogy a reduce fázis csak az összes map végeztével kezdődhet el, így a map műveletek végrehajtási ideje egy bottleneck. A teljes feladat lefutásakor a map kimenetek eldobhatók, a reduce függvény kimenete lesz a teljes MapReduce feladat eredménye, ami letárolásra kerül pl. elosztott fájlrendszeren (Hadoop), stb. További részletek: https://www.guru99.com/introduction-to-mapreduce.html

MapReduce implementációk

Eddig a MapReduce modell elvi koncepciójáról volt szó. De hol használják ezt ténylegesen? Az alábbiak példák a MapReduce programozási modell implementációira:

- **Apache Hadoop** elosztott fájl tárolás big data feldolgozással
  - Java, de bármely nyelven implementálható a map és reduce függvény
- **Apache CouchDB**
  - NoSQL adatbázis lekérdezéséhez JavaScript MapReduce megvalósítás
- **Disco**
  - Big data Python framework

    - Nokia
- **Infinispan**
    - Elosztott cache/NoSQL adatbázis
    - In-memory adatok feldolgozása MapReduce segítségével
- **Riak**
    - NoSQL adatbázis
    - JavaScript és Erlang MapReduce támogatás lekérdezésekhez

Ahogy az implementációkból is látszik, két területen játszik főbb szerepet az MapReduce programozási modell:

1. Big data és elosztott fájlrendszerek szerves része
    - Apache Hadoop
    - Disco
2. Elosztott NoSQL adatbázisok esetén komplex lekérdezésekhez (MapReduce adatforrása)
    - Apache CouchDB
    - Infinispan
    - Riak
    - MongoDB
    - Apache HBase

📖 3. fejezet
A MapReduce Apache Hadoop megvalósítása

A Hadoop MapReduce [1] egy keretrendszer, amely segítségével könnyen készíthetők olyan programokat, melyek hatalmas mennyiségű adatot (akár több terra-bájt) dolgoznak fel, és amelyek hagyományos eszközökből álló klaszteren (több ezer csomópont/node) párhuzamosan, megbízható és hibatűrő módon futtathatók.

Egy MapReduce program (Hadoop terminológia szerint *job*) a bemenő adathalmazt általában több kisebb darabra (*chunk*) osztja fel (*split*), amelyeket egy *map* feladat (*task*) dolgoz fel teljesen párhuzamos módon. A keretrendszer rendezi a map feladatok kimeneteit, amelyek aztán a *reduce* feladat bemenetei lesznek. Jellemzően mind a job kimenetei és bemenetei fájlrendszeren tárolódnak. A keretrendszer végzi a feladatok ütemezését, monitorozását és hiba esetén azok újrafuttatását.

Jellemzően a Hadoop klaszter számítási és tárolási csomópontjai megegyeznek, azaz a MapReduce keretrendszer és a Hadoop elosztott fájlrendszer (HDFS) [2] ugyanazon csomópontokon futnak. Ez a konfiguráció lehetővé teszi a keretrendszer számára a hatékony feladat ütemezést a csomópontokon, ahol az adatok is tárolódnak, nagyon magas átviteli sebességet elérve ezáltal a klaszteren belül.

A MapReduce keretrendszer egyetlen mester csomópontot (master node) tartalmaz, amely elnevezése `ResourceManager`, valamint minden egyes klaszter csomóponthoz egy `NodeManager` nevű worker node-ot és alkalmazásonként egyetlen `MRAppMaster` node-ot [3].

Egy minimális MapReduce alkalmazás megadja a bemeneti és kimeneti adatok helyét, valamint tartalmaz egy *map* és egy *reduce* függvényt a megfelelő interfészek és/vagy absztrakt osztályok megvalósításával. Ezek és minden egyéb job paraméter alkotja a job konfigurációját (*configuration*).

Ezek után a Hadoop job kliens elküldi a job-ot (jar/futtatható kód, stb.) és a konfigurációját a `ResourceManager` számára, amely szétosztja a szoftvert/konfigurációt worker csomópontokra, és elvégezi a feladatok ütemezését, monitorozását és állapot visszajelzését a kliens számára.

Noha a Hadoop keretrendszer Java nyelven íródott, nem feltétlenül szükséges a MapReduce programokat is Java-ban írni:

- A Hadoop Streaming [4] egy olyan eszköz a Hadoop keretrendszeren belül, melynek segítségével a felhasználók tetszőleges futtatható állományokat használhatnak job futtatásához (pl. shell script, azaz map és/vagy reduce megvalósításként.
- Lehetőség van a Hadoop Pipes [5] C++ API használatára is MapReduce alkalmazás megvalósításához.

A MapReduce keretrendszer kizárólag adat párokon dolgozik (*pair*), azaz egy job összes bemeneti adatát adatpárok halmazaként dolgozza fel, és adatpárokat állít elő kimenetként is, ám ezek típusának nem kell feltétlenül megegyeznie.

A `kulcs` és `érték` (`key, value`) párok Java osztályait a keretrendszer kiszerializálja, ezért ezeknek implementálniuk kell a `Writeable` interfészt. Ezen felül a kulcs osztályoknak meg kell valósítaniuk még a `WritableComparable` interészt is, hogy a keretrendszer rendezni tudja őket.

Egy MapReduce job bemeneti és kimeneti típusai a következők:

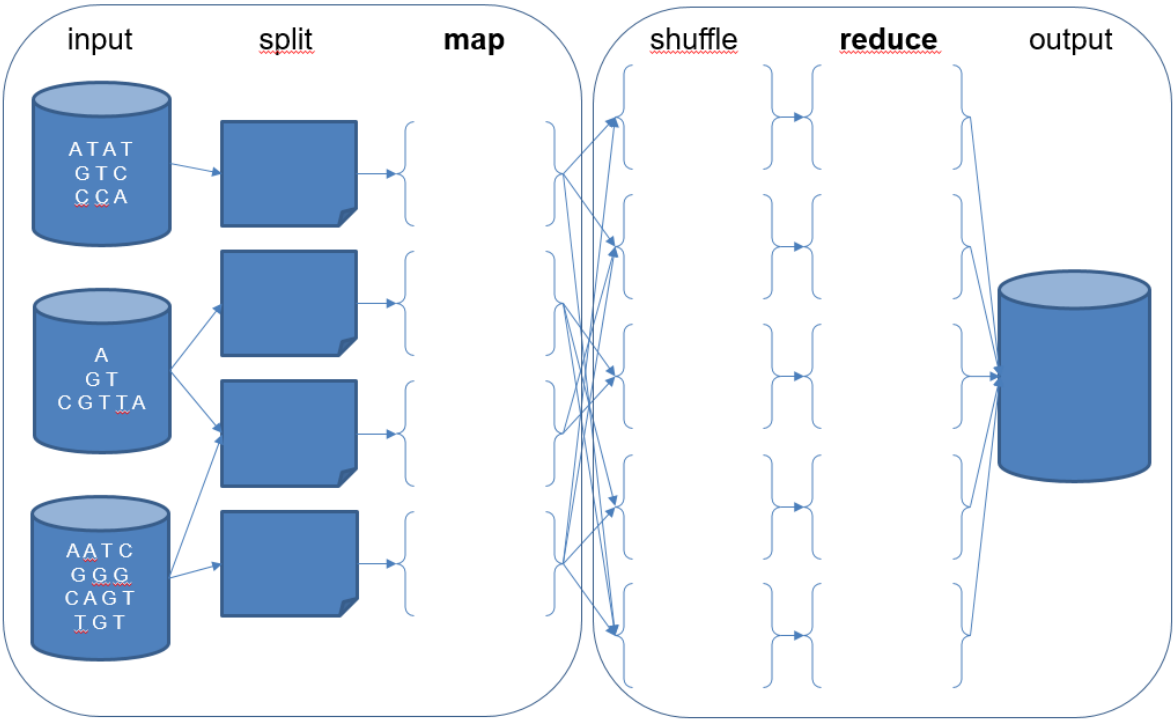
(input) `<k1, v1>` `->` **map** `->` `<k2, v2>` `->` **combine** `->` `<k2, v2>` `->` **reduce** `->` `<k3, v3>` (output)

Jól látszik, hogy minden fázisban lehetnek más-más típusúak a kulcsok és értékek is, kivéve a combine esetében. Erre a műveletre ugyanis megszorítás, hogy pont ugyanolyan kulcs-érték típusokat kell használnia, mint amit a map is használ.

Bővebb olvasnivaló: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

Ellenőrző kérdések

1. Melyik a MapReduce programozási modell két fő fázisa?
2. Milyen művelet és mire szolgál a `split`?
3. Hol és milyen célra használnak MapReduce implementációkat?
4. Mi a különbség a `reduce` és a `combine` művelet között egy Hadoop MapReduce program esetén?
5. Előfordulhat az, hogy egy Hadoop MapReduce programban ugyanaz a `reduce` és `combine` művelet is?
6. Milyen kötelező elemei vannak egy Hadoop klaszternek?
7. Milyen nyelven írhatunk MapReduce programot Hadoop környezetben?
8. Milyen egyezésnek kell lennie a `map`, `combine` és `reduce` által használt bemeneti és kimeneti kulcs-értékek típusai között?
9. Adott az alábbi DNS szekvencia adathalmaz elosztottan tárolva 3 adat node-on. Feladatunk egy olyan MapReduce program készítése, amely összeszámolja, hogy a szekvenciákban az egyes nukleinsavak hányszor fordulnak elő összesítve. Az ábrán jelöld ezen MapReduce program által elvégzett lépéseket és műveleteket a fentiekhez hasonlóan!



Referenciák

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

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

[3] https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[4] https://hadoop.apache.org/docs/r1.2.1/streaming.html

[5] https://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/pipes/package-summary.html

[6] https://www.datasciencecentral.com/profiles/blogs/how-to-install-and-run-hadoop-on-windows-for-beginners

[7] https://bigdataproblog.wordpress.com/2016/05/20/developing-hadoop-mapreduce-application-within-intellij-idea-on-windows-10/

[8] https://data-flair.training/blogs/hadoop-combiner-tutorial

[9] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51, no. 1 (2008): 107-113.

[10] https://github.com/ahmedfarhat/software-development-ebooks-1 [MapReduce Design Patterns Building Effective Algorithms and Analytics for Hadoop and Other Systems Kindle Edition by Donald Miner - 2013].pdf

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

[12] https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm