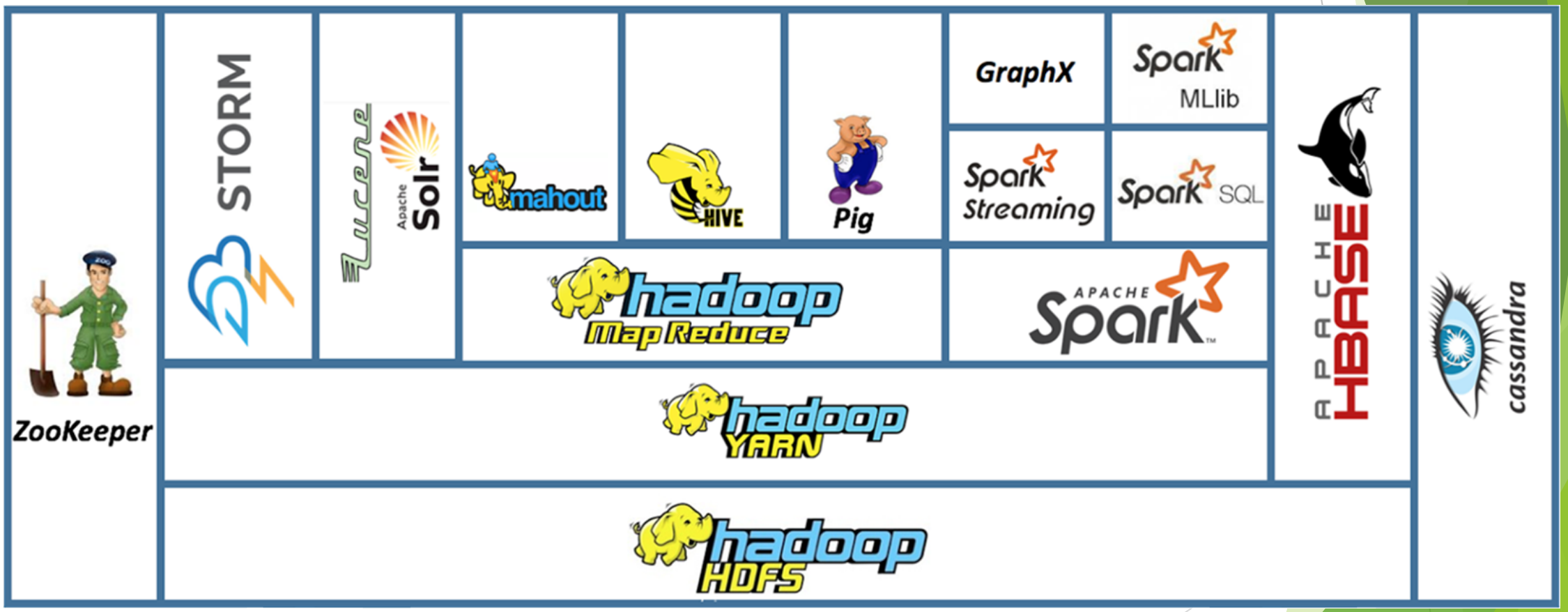
A MapReduce működése

Rusznák Attila

SZTAKI

Tartalomjegyzék

A Hadoop ökoszisztéma

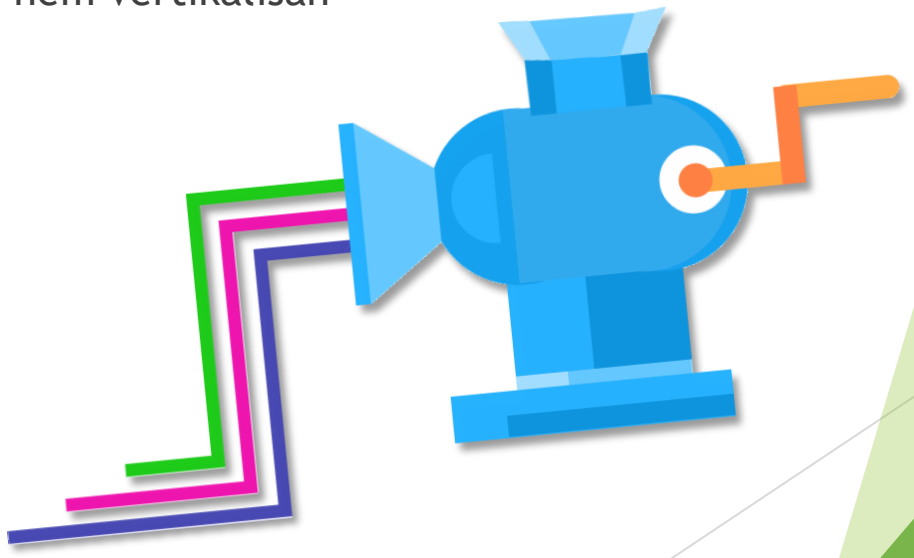


A MapReduce működése

Mi indukálta a MapReduce bevezetését?

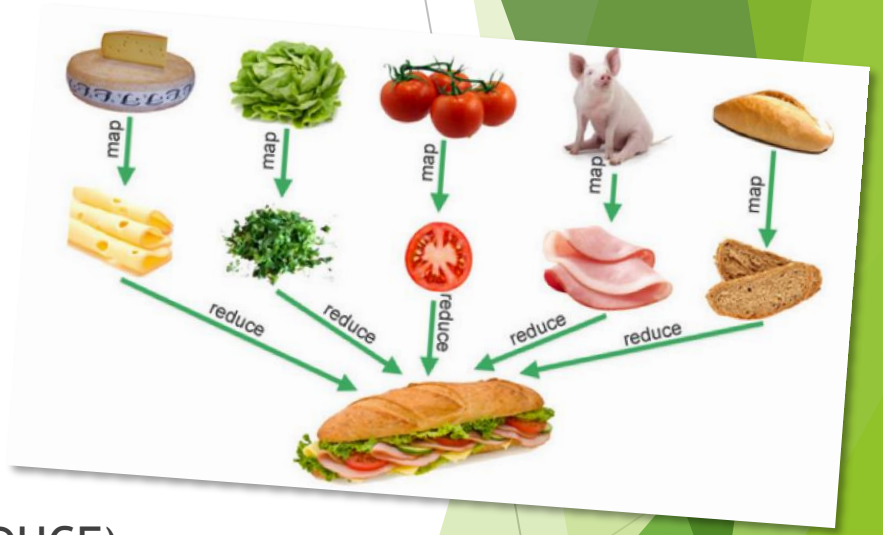
- Többféle formátumú, hatalmas mennyiségű adat érkezik
  - Pl.: írott szövegek, nehezen feldolgozható adatok

- Megjelent az igény a **párhuzamos programozás**ra
  - A feladatokat horizontálisan kell elosztani, nem vertikálisan
  - Ennek megvalósítása nem egyszerű



Mi az a MapReduce?

Olyan programozási paradigma, mely lehetővé teszi az adatok feldolgozását és a különféle folyamatok futtatását elosztott környezetben.



A MapReduce lépései:

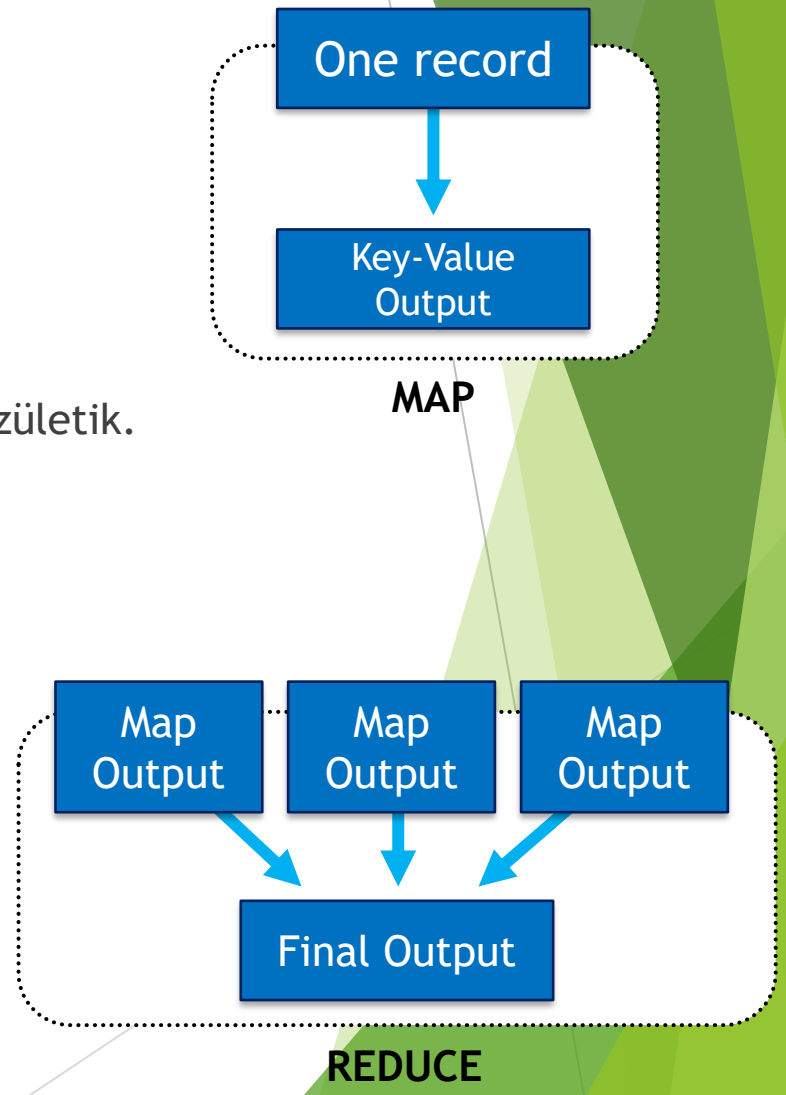
1. Felolvassuk az adatokat
2. Kivonatoljuk az adatokat egy logika mentén (MAP)
   - Különálló kulcs-érték párok állnak elő
3. Összegyűjtjük, rendezzük az előbb előállt kulcs-érték párokat
4. Elvégzünk a kulcs-érték pár halmazon egy összegző funkciót (REDUCE)
   - Pl.: aggregálás, filterezés, transzformálás
5. A végeredményt kiírjuk, azaz visszakapja a felhasználó

Mi az a MapReduce?

Két részre bontja fel a feldolgozandó feladatokat:

- **Map:** a feladatok végrehajtása párhuzamosan történik
  - Egyszerre több DataNode-on (worker-en) fut
  - Ezek a folyamatok az egyes gépeken eredményeznek 1-1 kimenetet.
  - Ezt követően összevonásra kerülnek, azaz egy közbenső eredmény születik.

- **Reduce:** a Map kimeneteken végez műveletet
  - Lefuthat egyszerre több DataNode-on is
  - Az egyes Map kimenetekből készít egy nagy, végleges kimenetet



Példa: szavak gyakoriságának előfordulása

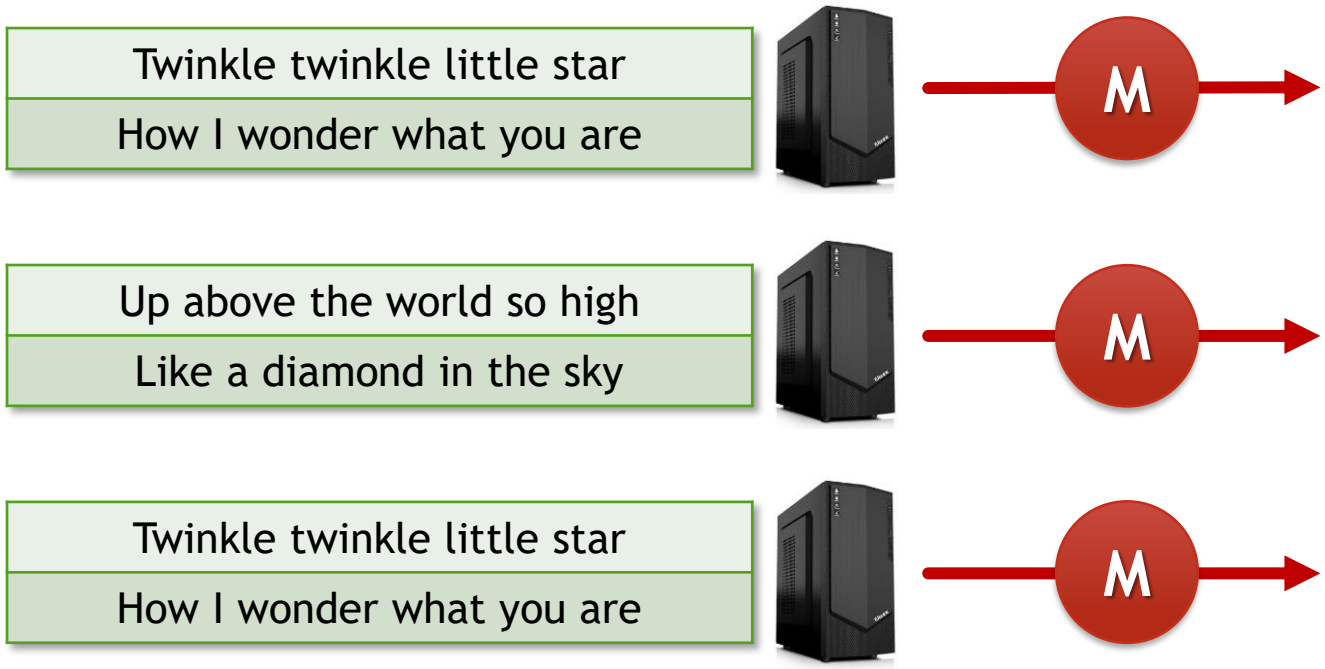
Twinkle twinkle little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky
Twinkle twinkle little star
How I wonder what you are
….

Consider a large text file

How?

Word	Frequency
above	14
are	20
how	21
star	22
twinkle	32
…	…

A Map Flow működése



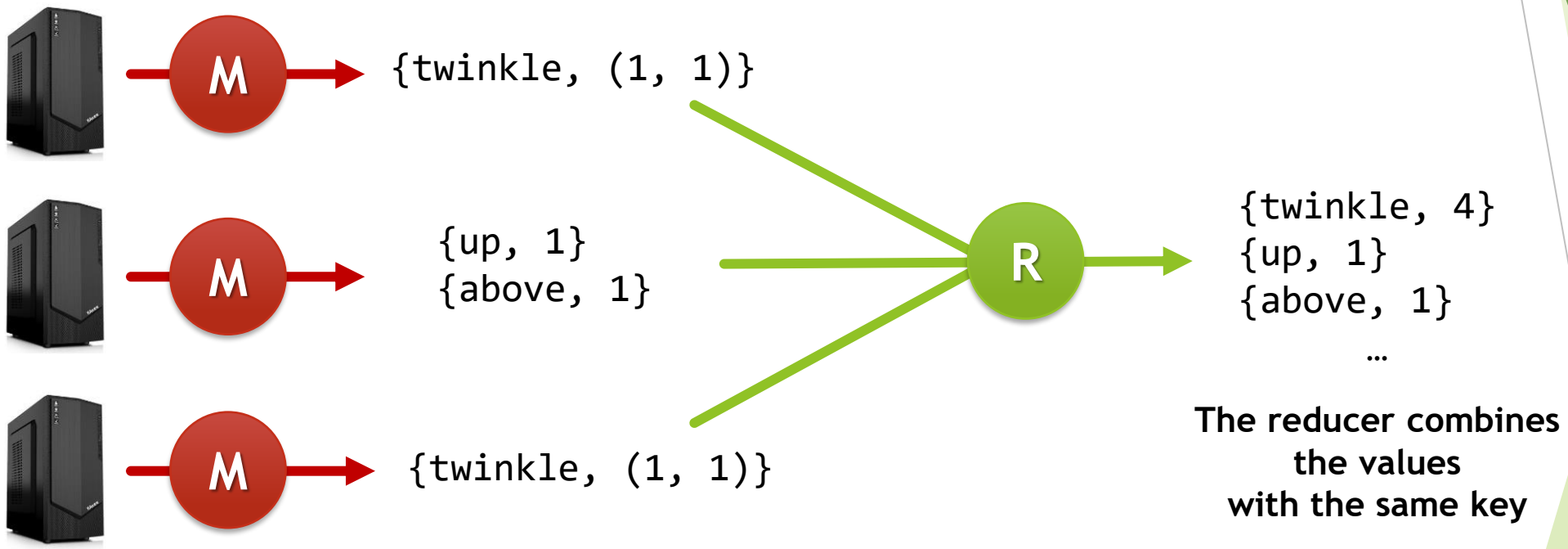
Word	# Count
twinkle	1
twinkle	1
little	1
star	1

Map phase

Minden worker gépen megtalálható a fájl egy részhalmaza

- Minden gépen fut egy Map fázis parallel módon
- Egy Mapper egyszerre egy rekordot dolgoz fel, a kimenet egy kulcs-érték pár

A Reduce Flow működése



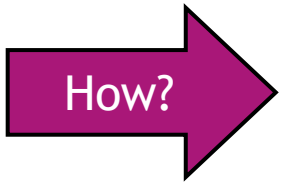
A Reduce megkapja az összes Map kulcs-érték pár kimenetét:

- Ezeket összegyűjti és az azonos szavakat összesíti
- Végül egy nagy list lesz a kimenet az egyes szavakkal és azok gyakoriságával

Példa: legmagasabb hőmérséklet

Date	Time	Temperature
2021/01/15	09:00	2
2021/01/15	10:00	3
2021/01/15	11:00	5
2021/01/15	12:00	8
2021/01/15	13:00	10
2021/01/15	14:00	11

Data is originally
Captured in this form



Date	Max Temp
2021/01/15	11
2021/01/16	13

A Map Flow működése

Date	Time	Temperature
2021/01/15	09:00	2
2021/01/15	10:00	3

M

Date	Temperature
2021/01/15	2
2021/01/15	3

Date	Time	Temperature
2021/01/15	11:00	5
2021/01/15	12:00	8

M

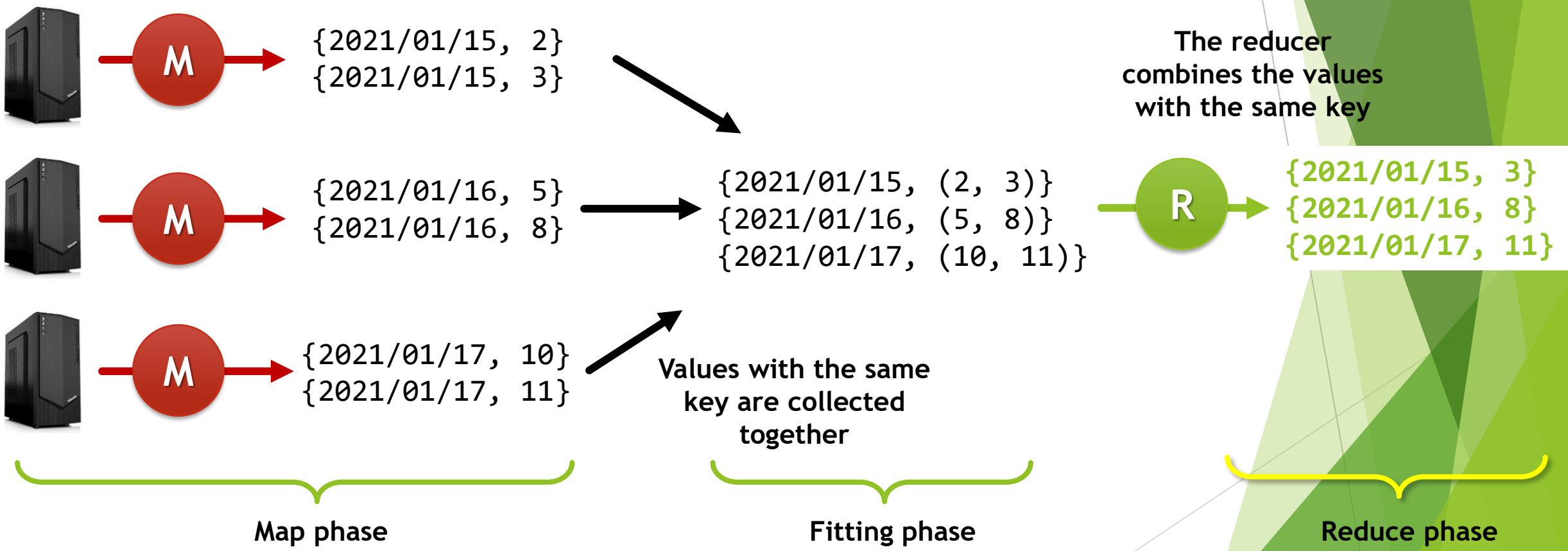
Date	Max Temp
2021/01/15	5
2021/01/15	8

Date	Time	Temperature
2021/01/15	13:00	10
2021/01/15	14:00	11

M

Date	Max Temp
2021/01/15	10
2021/01/15	11

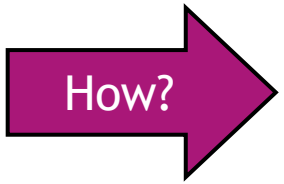
A Reduce Flow működése



Példa: profil látogatók száma

View ID	From Member	To Member
1	István	Benjámin
2	Lilla	István
3	Anikó	Csaba
4	Dóra	Ernő
5	Anikó	István
6	Dóra	Csaba

Data is originally Captured in this form



Member	# Profile Views
István	50
Lilla	15
Ernő	22
Anikó	23
Benjámin	32
Csaba	10

User-ViewCount Map

A Map Flow működése

View ID	From Member	To Member
1	István	Benjámin
2	Lilla	István

M

Member	# Profile Views
Benjámin	1
István	1

View ID	From Member	To Member
3	Anikó	Csaba
4	Dóra	Ernő

M

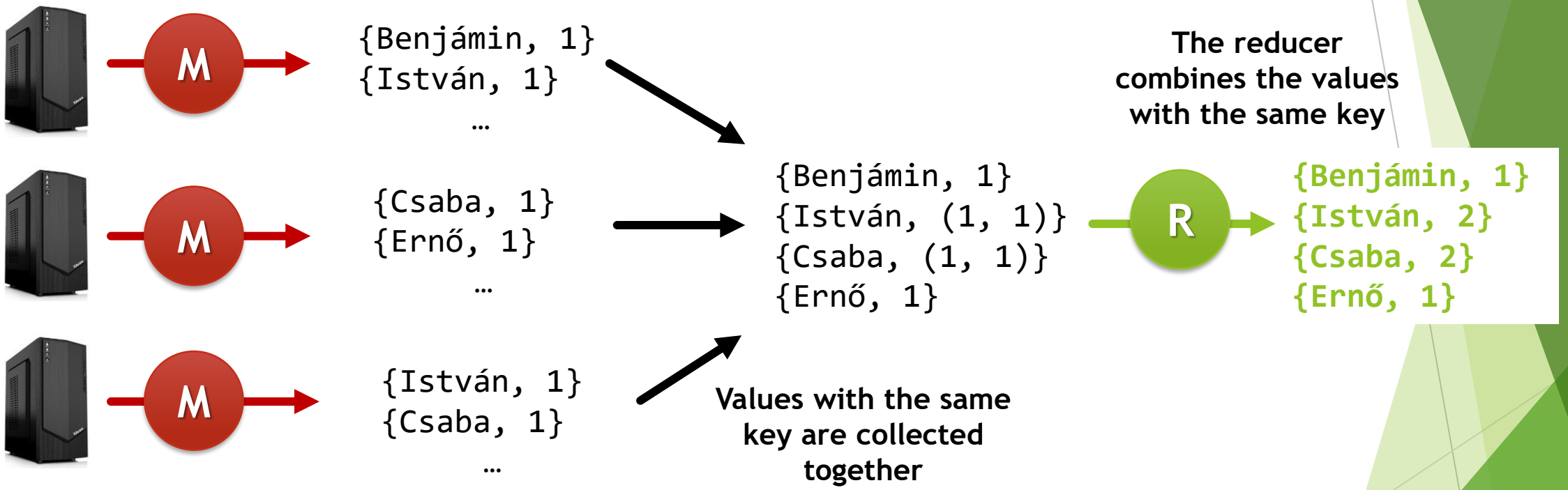
Member	# Profile Views
Csaba	1
Ernő	1

View ID	From Member	To Member
5	Anikó	István
6	Dóra	Csaba

M

Member	# Profile Views
István	1
Csaba	1

A Reduce Flow működése



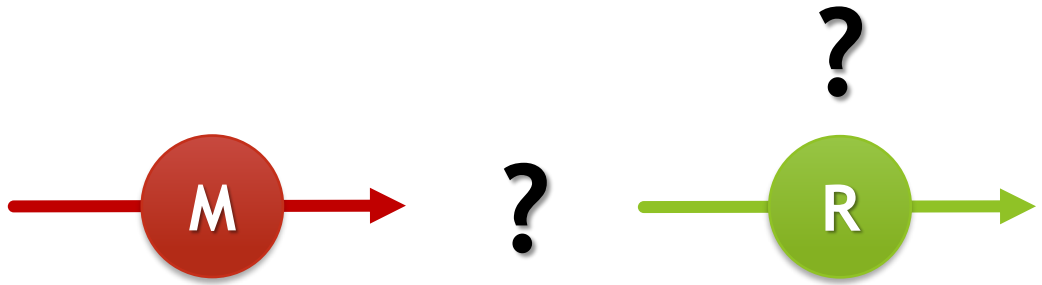
A MapReduce program tervezése

Arra kényszeríti a programozót, hogy parallel módon gondolkodjon

- Csak a paradigmára kell koncentrálni, a háttérfolyamatokat elvégzi a Hadoop

Mindig két kulcskérdést kell megválaszolnunk:

1. Mely kulcs-érték párokat kellene hogy kimenetként létrehozza a map lépés?
   - A kulcsok az inputként beérkező szavak lesznek egyesével, az érték pedig az 1.
2. Hogyan történjen az azonos kulcsok kombinálása a kapcsolódó értékekkel?
   - Minden egyes szó előfordulását összegezzük, hogy megkapjuk a szavak gyakoriságát.



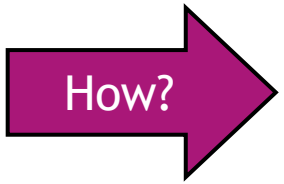
Egy MapRedpuce alkalmazás implementálása

ELKH Cloud

Példa: profil látogatók száma

View ID	From Member	To Member
1	István	Benjámin
2	Lilla	István
3	Anikó	Csaba
4	Dóra	Ernő
5	Anikó	István
6	Dóra	Csaba

Data is originally Captured in this form



Member	# Profile Views
István	50
Lilla	15
Ernő	22
Anikó	23
Benjámin	32
Csaba	10

User-ViewCount Map

A MapReduce program implementálása

A Hadoop Java nyelven íródott, így a MapReduce programozása is így történik:

- Az implementáláshoz számos beépülő library-t használhatunk
- Minden program három osztályból áll:
  - **Map:** olyan folyamat implementálása, melyek parallel módon futnak
  - **Reduce:** logika, mely egyesíti a Map által előállított kulcs-érték párokat
  - **Main:** egy vezérlőosztály, mely összeállítja a job-ot

Fontos hogy a Map osztály kimenete egyezzen meg a Reduce bemenetével!

A Map osztály felépítése

input key type,
input value type

<Key, Value>

Map

Input kulcs-érték párok:

```
<1, Benjámin>
<2, István>
<3, Csaba>
    …
```

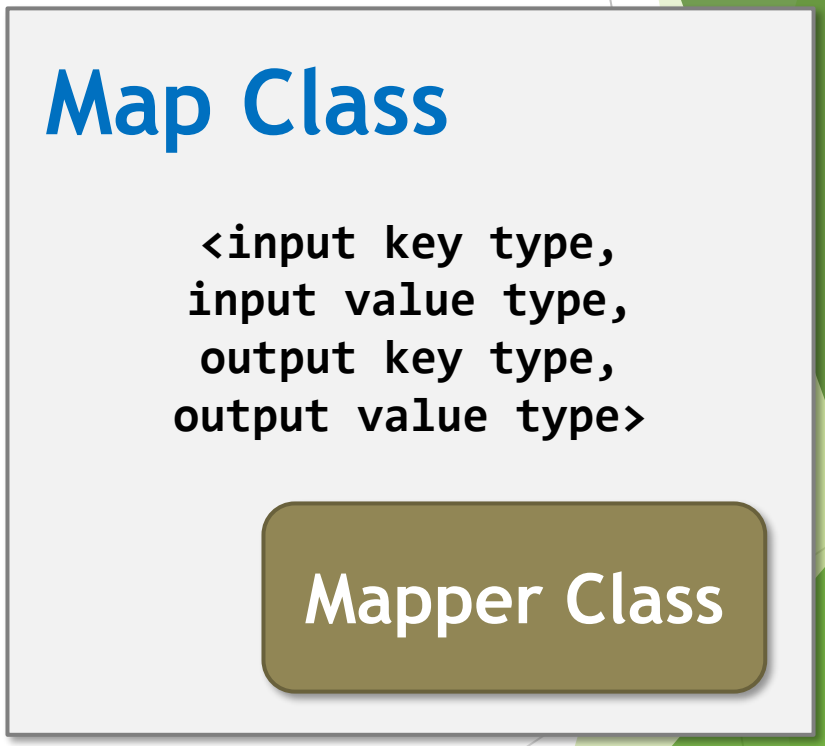
output key type,
output value type

<Key, Value>

Map

Output kulcs-érték párok:

```
<Benjámin, 1>
<István, 1>
<Csaba, 1>
<Ernő, 1>
    …
```



A Mapper egy generikus osztály és 4 típusparamétere van.

A Reduce osztály felépítése

input key type,
input value type

<Key, Value>

Reduce

Input kulcs-érték párok:

```
<Benjámin, 1>
<István, (1, 1)>
<Csaba, (1, 1)>
<Ernő, 1>
```

output key type,
output value type

<Key, Value>

Reduce

Output kulcs-érték párok:

```
<Benjámin, 1>
<István, 2>
<Csaba, 2>
<Ernő, 1>
```

Reduce Class

```
<input key type,
input value type,
output key type,
output value type>
```

Reducer Class

A Reducer egy generikus osztály és 4 típusparamétere van.

Java import utasítások

Map class:

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
```

Reduce class:

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
```

Main class:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
```

A Map class

```
public class Map extends
Mapper<LongWritable, Text, Text, IntWritable> {

@Override
public void map(LongWriteable key, Text value, Context context)
throws IOException, InterruptedException {

    String[] row = value.toString().split("\t");
    context.write(new Text(row[2]), new IntWritable(1));

}}
```

View ID	From Member	To Member
1	István	Benjámin
2	Lilla	István
0. index	1. Index	2. index

Generikus típusparaméterek:

- Speciális Hadoop csomagolóosztályok
  - **Input:** egy ID, ami nagy szám is lehet (LongWritable) és a beolvasott név (Text)
  - **Output:** a megtekintett profil neve (Text) és a látogatások száma, azaz mindig egy (IntWritable)
- A map metódus paraméterei:
  - A kulcs-érték pár, illetve a kontextus.
  - A kontextus teszi lehetővé a kommunikációt az osztály és a külső MapReduce keretrendszerrel (híd)

A Reduce class

```
{Benjámin, 1}
{István, (1, 1)}
{Csaba, (1, 1)}
{Ernő, 1}
```

```
public class Reduce extends
Reducer<Text, IntWritable, Text, IntWritable> {
@Override

public void reduce(Text key, Iterable<IntWritable> values,
Context context) throws IOException, InterruptedException {
     int count = 0;
     for (IntWritable value : values) {
          count += value.get();
     }
     context.write(key, new IntWritable(count));
}}
```

Generikus típusparaméterek:

- Speciális Hadoop csomagolóosztályok
  - **Input:** a Map által visszaadott név (Text) és a konstans egyes látogatás (IntWritable)
  - **Output:** a kimenet a név lesz ami bejött (Text), illetve egy számérték az összes látogatásról (IntWritable)
- A reduce metódus paraméterei:
  - Egy névérték (Text key), és egy iterálható értékek kollekciója (Iterable<IntWritable> values)
  - A kontextus teszi lehetővé a kommunikációt az osztály és a külső MapReduce keretrendszerrel (híd)

A Main class

```
public class Main extends Configured implements Tool {

    @Override
    pubic int run(String[] args) throws Exception {…}

    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new Main(), args);
        System.exit(exitCode);
    }
}
```

A Job működése:

- A Configured egy olyan Hadoop-tól örökölt osztály, ami minden alkalmazás futtatásához kell
- A Tool interfész segít a futtatott alkalmazás parancssori argumentumának átadásában
- A run metódus a Tool interfészből származik, itt konfiguráljuk a MapReduce Job-ot
- A main kibontja a megfelelő parancssori beállításokat, amire a MapReduce-nek szüksége van.

A Job run metódusa

```
@Override
pubic int run(String[] args) throws Exception {
    Configuration conf = this.getConf();

    Job job = Job.getInstance(conf);
    job.setJobName("viewCount");
    job.setJarByClass(Main.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    Path inputFilePath = new Path(args[0]);
    Path outputFilePath = new Path(args[1]);
    FileInputFormat.addInputPath(job, inputFilePath);
    FileOutputFormat.setOutputPath(job, outputFilePath);

    return job.waitForCompletion(true) ? 0 : 1;
}
```

Az alaposztályból kell a konfigurációs objektum.

Ezt átadjuk a Job-nak, hogy kapjunk belőle egy példányt. Kell neki egy név és egy vezérlőosztály (main).

Megadjuk a kimenet kulcs-érték párjainak osztályait amit a típusokhoz tudunk használni a Reduce esetén.

Beállítjuk a Map és Reduce osztályneveket.

A parancssori argumentumból kiolvassuk az input / output elérési útvonalakat. Futtatáskor fogjuk ezeket majd megadni.

A Job-nak lehetővé tesszük hogy tudjon írni és olvasni fájlokat.

Végül futtatjuk a MapReduce Job-ot.

A bemutatott programból alkalmazás fejlesztése és futtatása

A cloud-ban végrehajtható program legyen, a következő lépéseket kell végrehajtani:

1. Helyi PC-n történő alkalmazásfejlesztés
2. A Hadoop referencia architektúra segítségével a Hadoop klaszter létrehozása az ELKH Cloud-on
3. A helyi PC-n kifejlesztett alkalmazás telepítése a cloud-ban futó Hadoop klaszterre
4. A kifejlesztett alkalmazás végrehajtása a Hadoop klaszteren
5. A végrehajtás monitorozása webes felületen

Mindezt később fogjuk megmutatni a Hadoop referencia architektúrát bemutató előadást követően.