NÁSKOK
DÍKY
ZNALOSTEM

PROFINIT

BE0M33BDT
Big Data Technologies

MapReduce

Milan Kratochvíl, Sergii Stamenov, Jan Hučín

December 4th, 2019

MapReduce

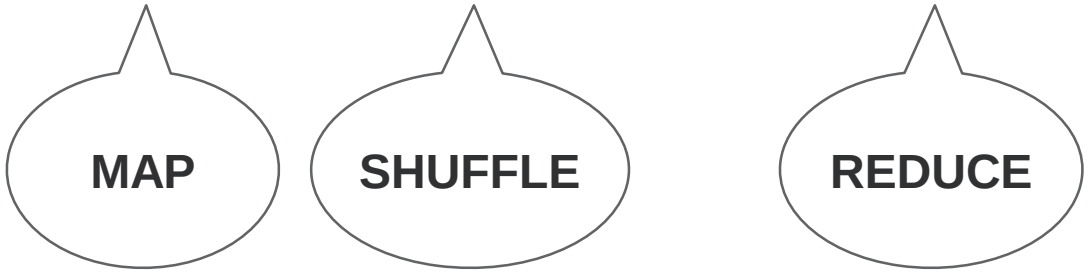
- Programing model for parallel computations
- Basic idea:
  - Data are distributed over cluster, I need put them together to groups
  - Mark of a group = **key** (same key = same group)
  - Exchange of data over cluster to get a group together (**shuffle**)
  - After grouping: aggregation (**reduce**)
  - Before grouping: preparation of data and key assignment (**map**)

PROFINIT

MapReduce

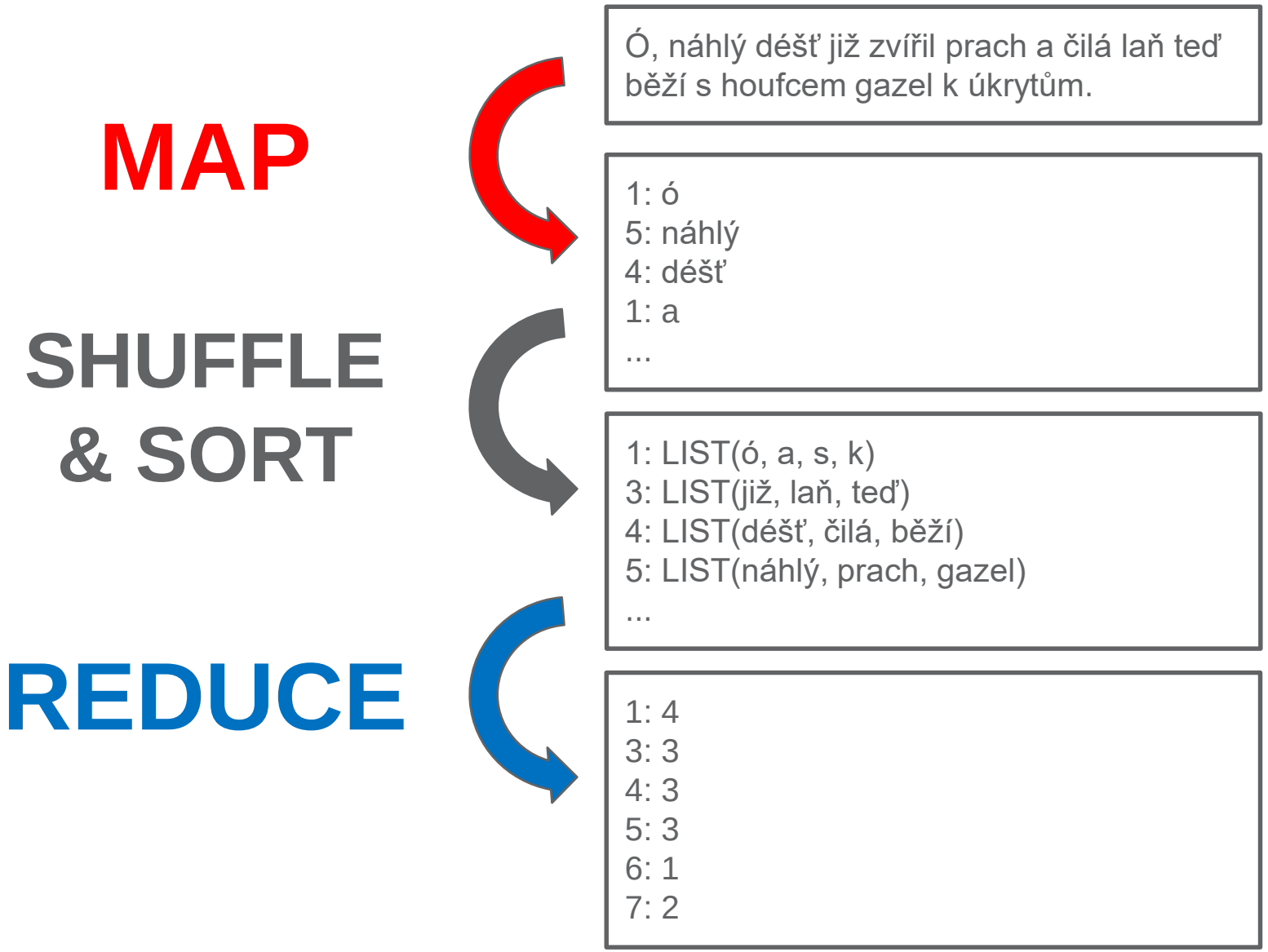
- Programing model for parallel computations
- Principle:
  - MAP
    - input data <VALUE1> converts to <KEY,VALUE2>
  - SHUFFLE & SORT
  - REDUCE
    - data <KEY, LIST(VALUE2)> aggregated to <KEY, VALUE3>

- `<V1>` → `<K,V2>` → `<K,LIST(V2)>` → `<K,V3>`



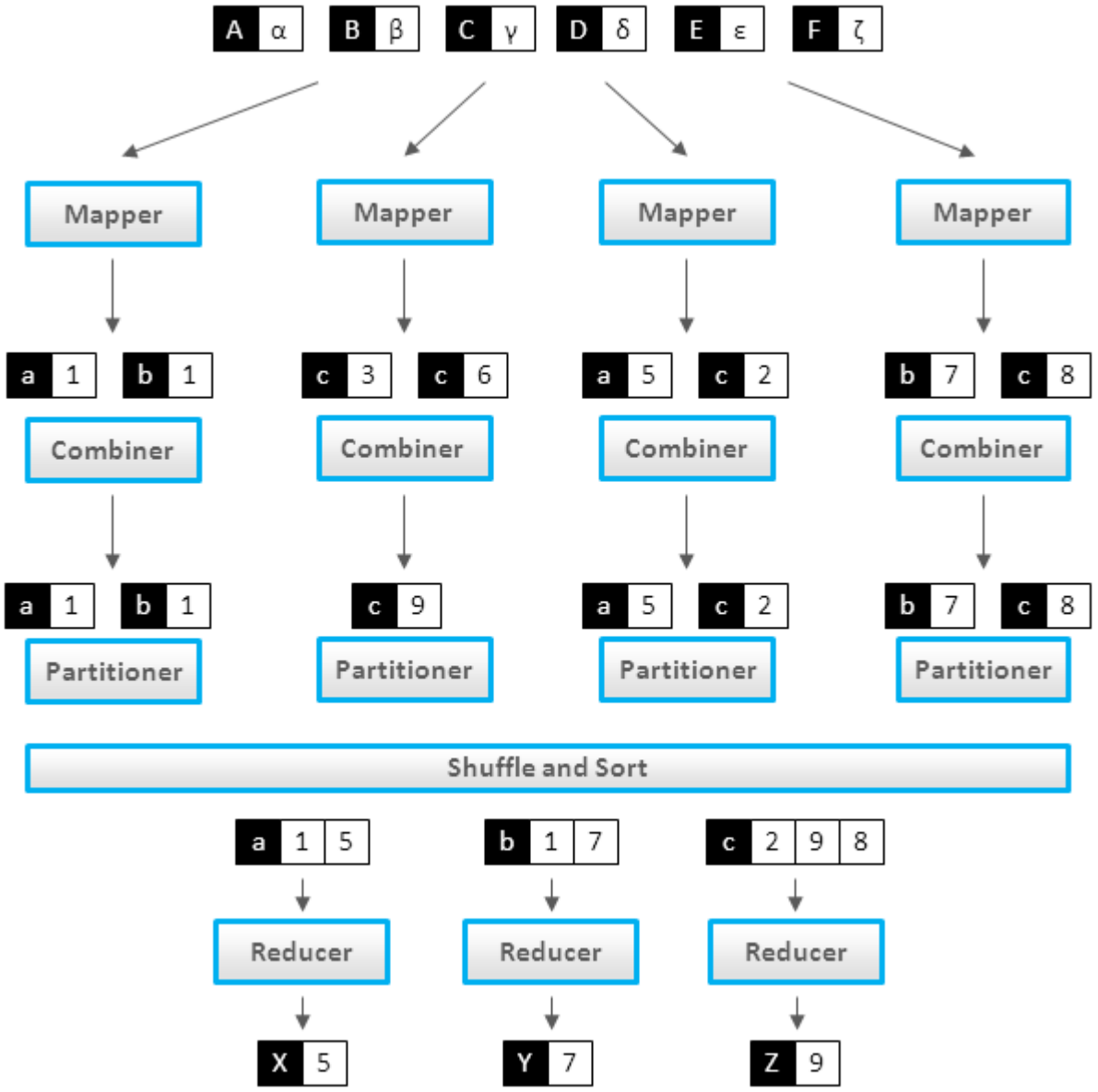
PROFINIT

Example: number of words by length



PROFINIT

MapReduce schema



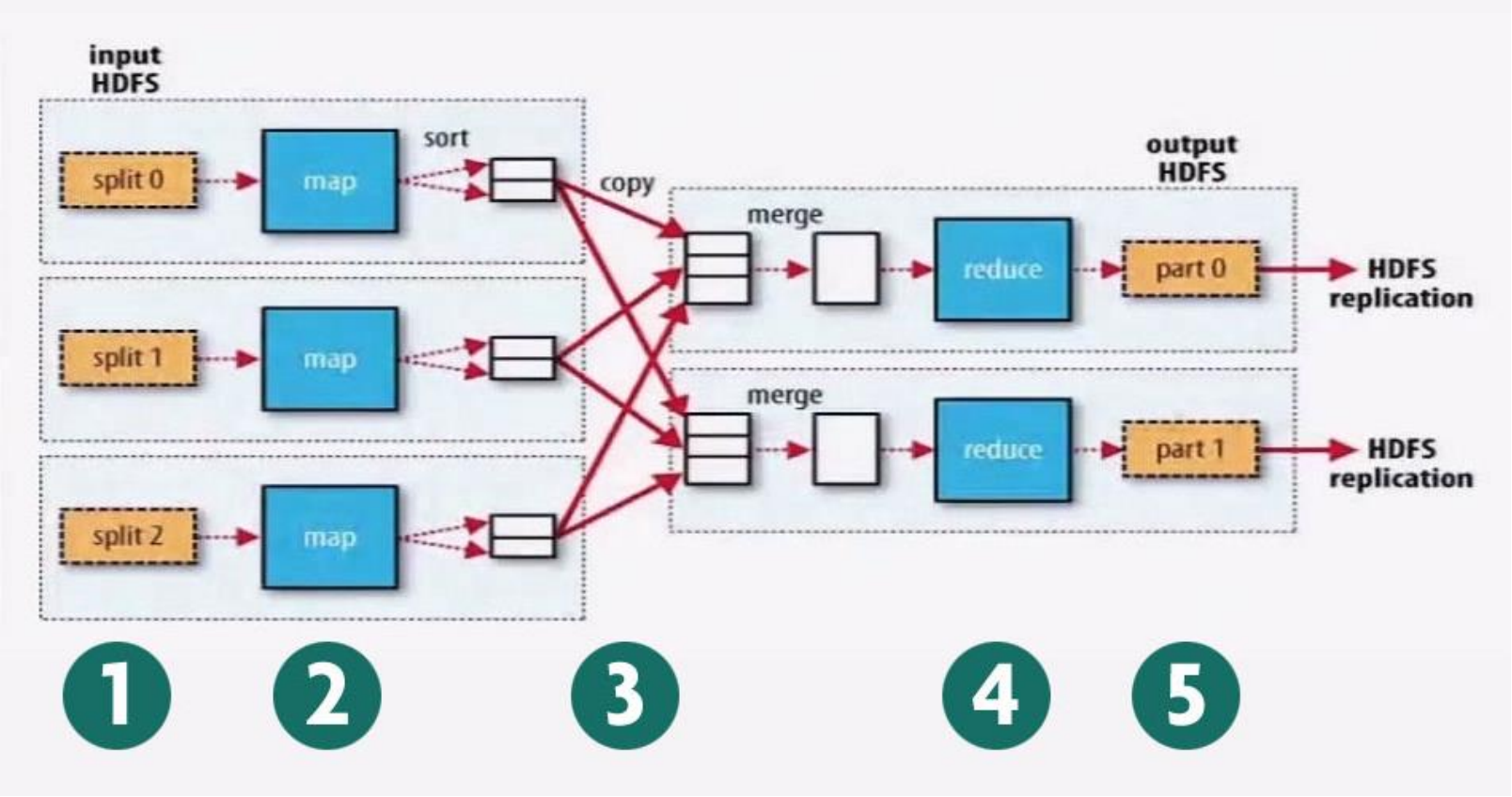
https://highlyscalable.wordpress.com/2012/02/01/mapreduce-patterns/

PROFINIT

Stages

- Planning
  - Do locally as most as you can – exploit HDFS replication
  - Transfer only necessary data over network
- Mapper saves its result to a local FS (not HDFS)
- Combine can do a useful aggregation on mapper side
- Shuffle & sort
  - Transfers all data with same key to same node
  - The most demanding stage of MapReduce (network load)
- Reduce
  - number of reducers << number of mappers
  - one reducer = one output file to HDFS

MapReduce data flow



https://content.pivotal.io/blog/hadoop-101-programming-mapreduce-with-native-libraries-hive-pig-and-cascading

PROFINIT

Task resolvable by MapReduce

- „Divide and conquer“

- **Formal requirements on reduce operation**
- ▲ must hold
  a) asociativity: (A ▲ B) ▲ C = A ▲ (B ▲ C)
  b) exists ◊ such that A ▲ ◊ = A
  c) commutativity: (A ▲ B) = (B ▲ A)

- Why?
  a) we have no control over order of operations
  b) node with no data must have no inluence on results
  c) mapper/shuffle can change data order

PROFINIT

Suitable tasks

- „School tasks“
  - word count
- Real tasks
  - summing up of partial reports
  - aggregating by items, clients, product types etc.
  - data filtering and validation
  - image processing
  - table joining
  - big data processing (matrix multiplication)

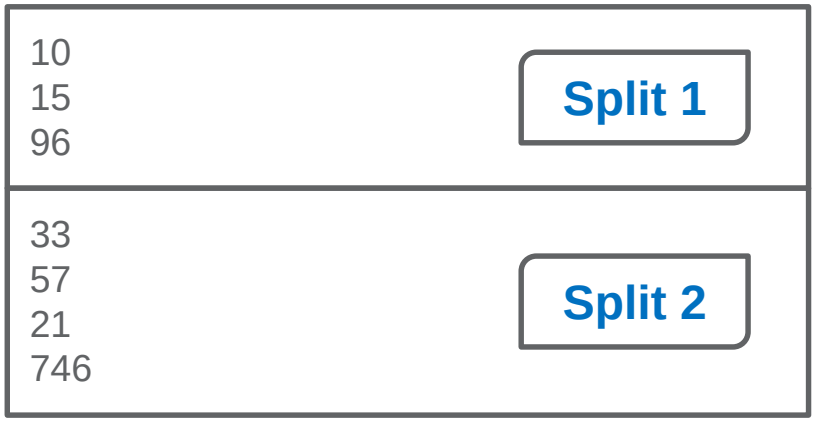
PROFINIT

Unsuitable tasks

- Median
  - does not hold associativity
- Average
  - average of partial averages ≠ overal average
- BUT: Task can be carried out smarter way.

PROFINIT

How to compute an average



- Combiner
  - average on map side
- Mapper output
  - SPLIT1:40.3333
  - SPLIT2:214.25
- Reducer output
  - AVG:127.2917

- Combiner
  - sum and count on map side
- Mapper output
  - SPLIT1:121:3
  - SPLIT2:857:4
- Reducer output
  - AVG:139.714

PROFINIT

MapReduce key features

- Stable, tested, reliable
  - „other components failing, MapReduce keeps running“
- Very small memory requirements
  - majority of operations is done on disks
- Very frequent disk usage
  - HDFS: only starting input and final output
  - local FS: intermediate results
  - many I/O operations
- Rather slow compared to other components
  - long initializing

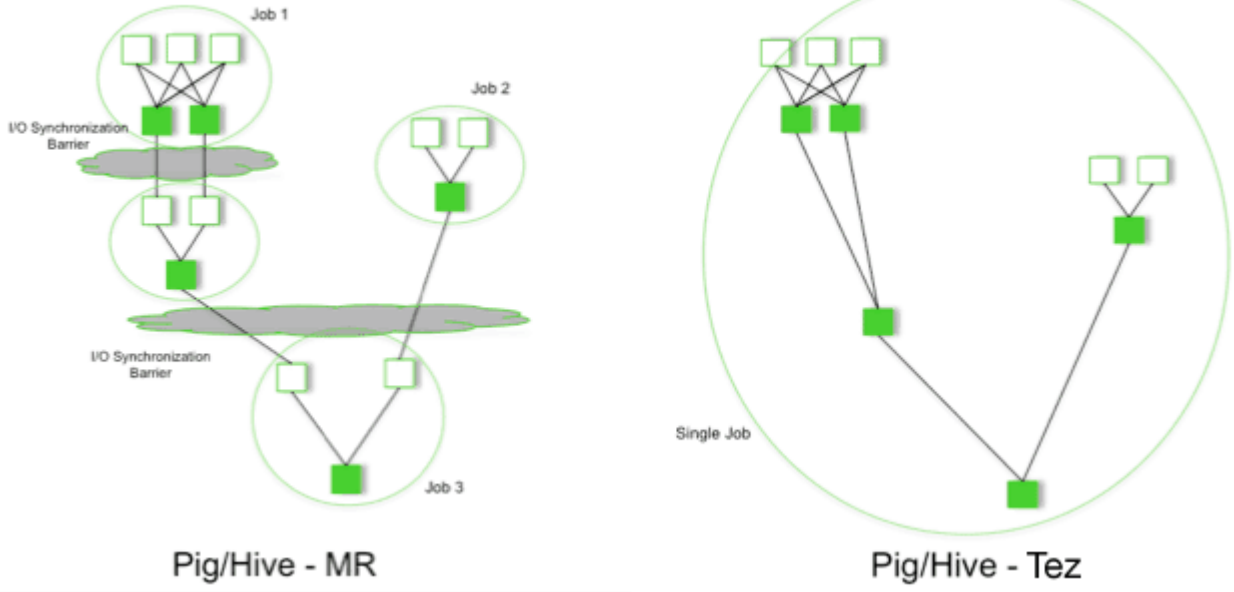
When to use MapReduce

- Task can run in parallel, „good“ reduce operation
- Processing big big data (petabytes)
- We have plenty of time
- Batch processing (not interactive)
- Lack of memory

PROFINIT

Variations on Hadoop

- Apache Tez
  - „optimized“ MapReduce
  - similar principle but combining of succesive steps
  - less I/O operations



https://hortonworks.com/blog/introducing-tez-faster-hadoop-processing/

PROFINIT

Variations on Hadoop

- Apache Spark
  - similar principle but intermediate results are kept in memory
  - enables interactive work

- ... next lecture

Thanks for your attention

PROFINIT
NÁSKOK DÍKY ZNALOSTEM

Profinit EU, s.r.o.
Tychonova 2, 160 00 Praha 6 | Telefon + 420 224 316 016

Web
www.profinit.eu

LinkedIn
linkedin.com/company/profinit

Twitter
twitter.com/Profinit_EU

Facebook
facebook.com/Profinit.EU

Youtube
Profinit EU