Map-Reduce

Igor Yakushin
ivy2@uchicago.edu

December 9, 2019

MapReduce

- **map** - convert each record into (*key*, *value*) pairs
- **reduce** - apply some reduce operation to the resulting set of (*key*, *value*); for example, sum values for each key, or find max, or min, etc.
- Obviously, maps can be done in parallel, independently from each other, on different nodes, for each record
- Between local map and global reduce, perhaps, apply reduce locally on each node before doing it globally and sort the results
- The native way to write your own MapReduce is to extend the corresponding Java classes from MapReduce API
- These days people rarely have to implement their own MapReduce but rather use high level tools like Pig, Hive, Spark, HBase, etc. unless you are a programmer implementing such a high level tool

Examples of applying MapReduce approach

- Count how many times each word occurs in a text
  - map - for each word w_i in a record (for example, a line in a text file) output $(w_i, 1)$
  - reduce - for each w_i sum the values
- Count the average number of social contacts a person has grouped by age
  - map - for each person p_i of an age a_i in a record, output $(a_i, (1, contacts_i))$
  - reduce - for each age a_i, sum separately the first and second component of the value to obtain number of people of a given age and total number of contacts, divide the latter by the former

Examples of applying MapReduce approach

- Finding common friends, like in Facebook. For each person a list of friends is given:

```
A -> B C D
B -> A C D E
C -> A B D E
D -> A B C E
E -> B C D
```

  - map - key is a person and a friend, sorted; value is the list of friends. For example, for C the output from map is:

```
(A C) -> A B D E
(B C) -> A B D E
(C D) -> A B D E
(C E) -> A B D E
```

  - reduce - group the results by key, for example:

```
(A B) -> (A C D E) (B C D)
```

    and find the intersection of value lists:

```
(A B) -> (C D)
```