Some solutions to Homework

COSC–254 – March 18, 2019

HW 02 – Ex. 1

Design a *MapReduce* algorithm to compute, given a file containing *one integer per line*, the *count* of the number of distinct integers.

HW 02 – Ex. 1

Design a *MapReduce* algorithm to compute, given a file containing *one integer per line*, the *count* of the number of distinct integers.

Example:

Input:

0
2
1
5
6
2
6

Output: 4

Things to remember

MapReduce does not allow for *global* variables, because of the distributed nature.

The input and output of *map* and *reduce* are *always pairs*.

Things to remember

MapReduce does not allow for *global* variables, because of the distributed nature.

The input and output of *map* and *reduce* are *always pairs*.

The *map* function operates only on one element at a time.

Things to remember

MapReduce does not allow for *global* variables, because of the distributed nature.

The input and output of *map* and *reduce* are *always pairs*.

The *map* function operates only on one element at a time.

Variables that persist across calls to `map` can exist in Hadoop, but not in the MapReduce model.

Things to remember

MapReduce does not allow for *global* variables, because of the distributed nature.

The input and output of *map* and *reduce* are *always pairs*.

The *map* function operates only on one element at a time.

Variables that persist across calls to `map` can exist in Hadoop, but not in the MapReduce model.

Sending all the input to a single reducer is not a good use of MapReduce

Idea

The algorithm will need *two rounds*:

$$\mathsf{map}_1 \to \mathsf{reduce}_1 \to \mathsf{map}_2 \to \mathsf{reduce}_2$$

Idea

The algorithm will need *two rounds*:

$$\text{map}_1 \rightarrow \text{reduce}_1 \rightarrow \text{map}_2 \rightarrow \text{reduce}_2$$

In the 1st round, we *identify the distinct integers* in the input

In the second round, we *compute the count* of distinct integers

First Map

Domain: $\mathbb{N} \times \mathbb{N}$ Codomain: $\mathbb{N} \times \{*\}$

The domain and codomain are *always Cartesian products* of two sets, because the input and output are always *pairs*.

map_1 takes a pair (offset, z) and outputs $(z, *)$

The specific value $*$ in the in output pair is *irrelevant*.

First Reduce

The shuffle phase ensures that the input to a reducer is in the form

$$(\text{key}, (\text{list of values } v \text{ emitted in pairs } (\text{key}, v) \text{ by mappers})$$

First Reduce

The shuffle phase ensures that the input to a reducer is in the form

$$(\text{key}, (\text{list of values } v \text{ emitted in pairs } (\text{key}, v) \text{ by mappers})$$

The *domain* of reduce is always related to the *codomain* of the corresponding map:
if the codomain of map is $A \times B$, the domain of reduce is $A \times (B \times \cdots \times B))$

The co-domain of map_1 is $\mathbb{N} \times \{*\}$, then the domain of reduce_1 is $\mathbb{N} \times (\{*\} \times \cdots \times \{*\})$

First Reduce

The shuffle phase ensures that the input to a reducer is in the form

$$(\text{key}, (\text{list of values } v \text{ emitted in pairs } (\text{key}, v) \text{ by mappers})$$

The *domain* of reduce is always related to the *codomain* of the corresponding map:
if the codomain of map is $A \times B$, the domain of reduce is $A \times (B \times \cdots \times B))$

The co-domain of map_1 is $\mathbb{N} \times \{*\}$, then the domain of reduce_1 is $\mathbb{N} \times (\{*\} \times \cdots \times \{*\})$

The co-domain of reduce_1 is $\mathbb{N} \times \{*\}$.

reduce_1 takes a pair $(z, (*, \ldots, *))$ and outputs $(z, *)$.

We can use reduce_1 in the *combiners*!

First Round Recap

$\text{map}_1 : (\text{offset}, z) \to (z, *)$
$\text{reduce}_1 : (z, (*, \ldots, *)) \to (z, *)$

The "global" output of reduce_1 is a file of pairs in the form $(z, *)$.

How many of these pairs are there?

First Round Recap

$\text{map}_1 : (\text{offset}, z) \to (z, *)$
$\text{reduce}_1 : (z, (*, \ldots, *)) \to (z, *)$

The "global" output of reduce_1 is a file of pairs in the form $(z, *)$.

How many of these pairs are there? As many as the number of distinct integers (k)

Idea

The algorithm will need *two rounds*:

$$\mathsf{map}_1 \to \mathsf{reduce}_1 \to \mathsf{map}_2 \to \mathsf{reduce}_2$$

✔ In the 1st round, we *identify the distinct integers* in the input

In the second round, we *compute the count* of distinct integers.

Second Map

The domain of the second map is the codomain of the first reduce: $\mathbb{N} \times \{*\}$

map_2 maps $(k, *)$ to $(*, 1)$.

The codomain is $\{*\} \times \{1\}$

Second Reduce

The domain is $\{*\} \times (\{1\} \times \cdots \times \{1\})$, or better $\{*\} \times (\mathbb{N} \times \cdots \times \mathbb{N})$ (for combiners!)

The codomain is $\{*\} \times \mathbb{N}$

reduce$_2$ takes $(*, (v_1, \ldots, v_\ell))$ and outputs $(*, \sum_{i=1}^{\ell} v_i)$

We can again use reduce$_2$ in the combiners.

Second Round Recap

$\text{map}_2 : (z, *) \to (*, 1)$
$\text{reduce}_2 : (z, (v_1, \ldots, v_\ell)) \to (*, \sum_{i=1}^{\ell} v_i)$

The "global" output of reduce_2 is a single pair $(*, k)$.