UNIVERSITÀ DI PISA

Artificial Intelligence and Data Engineering

Cloud Computing

PageRank

Project Documentation

TEAM MEMBERS:
Daniele Cioffo
Edoardo Fazzari
Federica Baldi
Mirco Ramo

Academic Year: 2020/2021

Contents

1 — PageRank

1.1 Introduction

In this section a description of the MapReduce implementation of *Page Rank* is given. The algorithm is carried out in **four distinct steps**:

1. *Nodes counting* phase
2. *Graph Construction* phase
3. *Page Rank Computation* phase
4. *Sorting* phase

HADOOP: A cleanup function is runned after each step in order to taper down the memory usage as much as possible.

1.2 First Phase: Nodes Counting

To compute PageRank the total number of nodes is required. Considering that the number of nodes is unknown at the beginning –and may be huge–, this is assessed by using a MapReduce approach for optimization reasons in the following way:

Algorithm 1 Nodes Counter Mapper

```
1: procedure MAP(pageid id, page p)
2:     if p is not empty then
3:         EMIT(uniqueKey, 1)
```

Algorithm 2 Nodes Counter Reducer

```
1: procedure REDUCE(key k, values [v_1, v_2, ...])
2:     for all value in values do
3:         sum ← sum + value
4:     EMIT(k, sum)
```

1.3 Second Phase: Graph Construction

In this phase we parse the information in the input file removing all the uninterested fields (e.g., content of the webpage). Also, to each page, we provide the initial PageRank thanks to the already calculated total number of nodes.

Algorithm 3 Graph Construction Mapper

```
1: procedure MAP(key k, page p)
2:     outgoingEdges new AssociativeArray
3:     title ← getTitle(p)
4:     outgoingEdges ← getOutgoingEdges(p)
5:     EMIT(title, outgoingEdges)
```

Algorithm 4 Graph Construction Reducer

1: **procedure** INITREDUCE(Configuration c)
2: $N \leftarrow c.numberOfNodes$
3: **procedure** REDUCE(title t, edges $[e_1, e_2, \dots]$)
4: $initialPageRank \leftarrow \frac{1}{N}$
5: $edges \leftarrow e_1$
6: EMIT(title, {initialPageRank, edges})

In *Algorithm 4* only e_1 is considered because each page is never reduplicated in the dataset, thus in the mapper the key produced is always unique (i.e., the list of values in the reducer will be made up just by one item).

1.4 Third Phase: PageRank Estimation

In this section, the relaxed pagerank iteration is presented. In the computation, we did not redistribute the probability mass lost by dangling nodes since it was not requested by the project specification, but a special key (i.e., **DANGLING**) is used for taken into account the total mass lost.

The number of iteration is fixed at the start of the execution. We do not converge to a (or a more or less) consistent state, because the presence of dangling nodes will cause importance (i.e., pagerank mass) to leak out.

Algorithm 5 PageRank Computation Mapper

1: **procedure** MAP(key k, formattedPage p)
2: EMIT(p.title, {0, p.outgoingEdges})
3: **if** p.outgoingEdges **is not** empty **then**
4: **for all** outgoingEdge **in** p.outgoingEdges **do**
5: EMIT(outgoingEdge, $\{\frac{p.pagerank}{p.outgoingEdges.length}, NULL\}$)
6: **else**
7: EMIT(DANGLING, {p.pagerank, NULL})

Note: *outgoingEdge* is a title itself.

Algorithm 6 PageRank Computation Reducer

```
1: procedure InitReduce(Configuration c)
2:     N ← c.numberOfNodes
3:     D ← Damping
4: procedure Reduce(title t, pages [p_1, p_2, ...])
5:     if title = DANGLING then
6:         for all page in pages do
7:             s ← s + page.pagerank
8:         EMIT(t, {s, NULL})
9:     else
10:        p new Page
11:        for all page in pages do
12:            if page.hasOutgoingEdges() then
13:                p.outgoingEdges = page.outgoingEdges
14:            else
15:                s = s + page.pagerank
16:        p.pagerank = (1−D)/N + D * s
17:        EMIT(t, p)
```

1.5 Fourth Phase: Sorting

The final step is sorting the webpages by decreasing rank, this is done making advantage of the sorting mechanism of MapReduce.

Algorithm 7 Sorting Mapper

```
1: procedure Map(key k, formattedPage p)
2:     title ← p.title
3:     pagerank ← p.pagerank
4:     EMIT(pagerank, title)
```

Algorithm 8 Sorting Reducer

```
1: procedure Reduce(pagerank rank, titles [t_1, t_2, ...])
2:     for all title in titles do
3:         EMIT(title, rank)
```

2 — Hadoop Implementation

3 — Spark Implementation in Java

4 — Spark Implementation in Python