University of Pisa

Master in Computer Engineering
Project for Electronic systems

PageRank implementation using hadoop

Author:
Tommaso Burlon

Academic year
2020/2021

Summary

1 Introduction

In this project I implemented the PageRank algorithm using the MapReduce paradigm in Hadoop. Unfortunately I've only got one virtual machine so the framework was set to work in pseudo-distributed mode in this way I haven't obtain the speedup that I could have gained using the distributed mode.

2 The algorithm

PageRank is an algorithm used to calculate an index of "importance" of web pages (or more in general the index of importance of a specific node in a graph). This index is calculated following the idea that a node is more important than another if by following a random path in the graph it is more likely to pass through this node. This algorithm can also be expressed in a recursive formula:

$$P_n(a) = \alpha \cdot \frac{1}{N} + (1 - \alpha) \cdot \sum_{b \in L(n)} \frac{P_{n-1}(b)}{C(b)}$$

where $P_n(a)$ is the index that compute the importance of the node a at iteration n, N is the number of nodes in the graph, α is the convergence factor, $C(b)$ are the total number of outcoming edges of node b and $L(n)$ is the set of nodes that have incoming edges coming from n. This formula is recursive in terms of the vector $P(n)$, so it is possible to implement this formula using an iterative paradigm.

3 MapReduce

Mapreduce is a famous programming paradigm useful to compute data in a distributed system. This algorithm implement two Main entity the Mapper and the Reducer that can communicate using messages. The Mapper receives as input a pair (K_1, V_1) and return in output a list of pairs $[(K_2, V_2), (K_3, V_3), ..., (K_m, V_m)]$ this pairs than get shuffle and sorted using the K element of the pair. Then the data is given to the Reducer which get as input a pair $(K, [V_1, V_2, V_3, ..., V_n])$ and return in output a pair K, V. Using this paradigm it is possible to implement the PageRank algorithm. Unfortunately the algorithm has a strong data dependence between step of

iteration this dependence is stronger when the graph is highly connected while is weaker in the opposite situation (if two part of the graph are disconnected they can be computed in parallel with no data dependence). Using the MapReduce paradigm unfortunately it is really complex to develop an algorithm that exploit the topology of the graph to send less messages to other parts of the distributed system.

4 Pseudocode

Mapper code:

```
procedure MAPPER(key, node)
    P ← node.rank/len(node.adjacencyList)
    for N in node.adiacencyList do
        SEND(N, P)                          ▷ send data to node
    end for
    SEND(node.id, node)                     ▷ mantain the graph structure
end procedure
```

Reducer:

```
procedure REDUCER(key, list)
    RES ← 0
    node ← null
    for N in list do
        if isNODE(N) then
            node ← N                        ▷ Restore graph structure
        else
            RES ← RES + N
        end if
    end for
    node.rank ← α/SIZE + (1 − α) ∗ RES      ▷ Update node rank
    SEND(node.id, node)
end procedure
```

5 Implementation

Using the Hadoop framework I've implemented a version of the PageRank algorithm. My application is divided into 3 parts: the first and the second

part are useful to parse the graph and compress it while the third part compute the PageRank index. The first part receive as input a file (or more files) that in every line contains the information of a node like an unique key of the node (for example an URL) and the list of outcoming link from that node (an ¡a¿ HTML tag for example) of course the name of the link and the name of the node must have the same format, the output of this part is a more easily representation of the graph: a file that in every line is stored the information regarding a single node removing all the text from the input that do not contain useful information (for example the body o the page). The second part of the program compress the file through a substitution of the key of the nodes with a unique number (the offset in byte of the line that contains the information of the node) in this way nodes with long name (like for example string/URL) are replaced with number. The third part compute the PageRank index as it is described in the Pseudocode paragraph.

6 Optimization

Using a single virtual machine using more reducer could bring no benefit. However some optimization could also be possible. For example I have implemented a serializable class to transport topology information of the graph with reduced overhead, the default methods to send complex information to the mapper is by using the Text class which use a String serialize and deserialize protocol which is slower. To reduce the number of messages sent by the Mapper a Combiner class was implemented, this class it is useful because it is common for web pages to contain more than one link to other pages, in this way the Combinator class aggregate this links reducing the total number of messages. Other optimization mechanisms could be implemented like for example using a user-defined file reader so the file exchanged between iteration of the algorithm could be binary files instead of text file reducing the parsing time of the application. A more clever optimization is also possible by exploiting the specific topology of the graph, this kind of optimizations however are difficult to implement following a MapReduce approach.

7 Results

I tested the algorithm using two different dataset: the "wiki-micro.txt" file containing a smaller version of the Wikipedia english website, the other dataset was obtained by scraping 500 webpages of the site "Pokemon Central" that contains information of the popular franchise "Pokemon". Here the results for the first dataset:

NODE NAME	RANK VALUE	RANK EPSILON
[Special:DoubleRedirects|double-redirect]	0.000311657	0
[Wikipedia:GFDL standardization]	0.000222569	0
[Wikipedia:Non-free content/templates]	0.000200351	0
[WP:AES|←]	9.90539e-05	0
[Project:AutoWikiBrowser|AWB]	7.9455e-05	0
[Wikipedia:Deletion review|deletion review]	7.33228e-05	0
[user:freakofnurture|...]	6.03987e-05	0
[Template:R from title without diacritics|R from title without diacritics]	6.03987e-05	0
[User:AWeenieMan/furme|FURME]	4.6576e-05	0
[Wikipedia:Articles for deletion/PAGENAME (2nd nomination)]	4.45189e-05	0

As we can see every name in this list is a user page or an utility page we could foresee it because these kind of pages have got a lot of incoming links. Here the results for the secodn dataset:

NODE NAME	RANK VALUE	RANK EPSILON
[/Normale]	0.000196975	7.45945e-11
[/Categoria_danno#Fisico]	0.000183752	2.32282e-11
[/Pok%C3%A9mon_Spada_e_Scudo]	0.000177489	1.07264e-10
[/Categoria_danno#Stato]	0.000177148	2.17261e-11
[/Pok%C3%A9mon_Oro_HeartGold_e_Argento_SoulSilver]	0.000170319	1.89818e-10
[/Categoria_danno#Speciale]	0.000136419	1.89409e-11
[/Pok%C3%A9mon_Diamante_Lucente_e_Perla_Splendente]	0.000135241	7.57222e-11
[/Pok%C3%A9mon_Oro_e_Argento]	0.000125447	2.46488e-10
[/Dollari_Pok%C3%A9mon]	9.88654e-05	5.45022e-12
[/Pok%C3%A9mon_Cristallo]	9.65376e-05	1.73104e-10

As we can see the page with the highest rank are the ones that we can expect: pages regarding the games, pages regarding pokemon types or general abilities shared by a lot of pokemon.