University of Science and Technology in Zewail City

CIE 427

WDC PageRank MapReduce

2023

Noureldin Mohamed

s-noureldin.hamedo@zewailcity.edu.eg

Contents

1 Introduction

The World Wide Web is a vast network of interconnected domains, where hyperlinks serve as the backbone connecting information across different websites. Understanding the link structure of the web is crucial for various applications, including search engine optimization, web analytics, and information retrieval. In this report, we leverage the power of MapReduce, a parallel processing framework, to analyze the link structure of the web using the WDC Hyperlink Graphs dataset.

The dataset is represented in the Index/Arc format, consisting of two files: the index file, which provides information about each node (domain), and the arc file, which specifies the directed edges (hyperlinks) between nodes. Our objective is to perform comprehensive analysis on the link relationships within this dataset, specifically focusing on the **outlinks** and **inlinks** of each domain.

2 Data Description

2.1 Index File

The `index` file contains information about each node (**domain**) in the graph. Each line represents a single node, with two columns separated by tabs. The first column provides the node name, and the second column indicates the node index. The nodes are sorted by index, showcasing a total of 106 nodes in the dataset.

Node Name	Node Index
1000notes.com	0
100500.tv	1
abebooks.com	2
abebooks.de	3
amazon-presse.de	4
...	...
amazon.fr	12
amazon.it	13
angrybirds.com	14
animationplayhouse.com	15

Table 1: Sample data from the `index` file.

2.2 Arc File

The `arc` file specifies the directed edges (**hyperlinks**) between nodes. Each line in this file signifies a directed edge, with two columns denoting the origin and target nodes, respectively. The nodes are sorted by index, resulting in a total of 141 arcs in the dataset.

Origin Node	Target Node
7	5
7	6
7	8
7	9
7	10
...	...
10	8

Table 2: Sample data from the `arc` file.

These files collectively represent a graph with 106 nodes and 141 arcs, providing a structured view of the connectivity within the dataset. The data is sorted by index, facilitating efficient analysis and interpretation.

3 Setting Up Hadoop and Data Preparation

To begin the Hadoop environment, start the NameNode, DataNode, Secondary NameNode, and the ResourceManager by opening the terminal and entering the following command:

```
start-all.sh
```

Once Hadoop is running, create a user directory in the Hadoop Distributed File System (HDFS) using the following commands:

```
hadoop fs -mkdir /user
hadoop fs -mkdir /user/hadoopuser
hadoop fs -mkdir /user/hadoopuser/assignment1
```

If you need to remove a folder, you can use:

```
hadoop fs -rm -r <path>
```

The user directory is now set up. Next, move to the local source directory and copy the data to HDFS:

```
mkdir data
mv example_arcs data/
hadoop fs -copyFromLocal Downloads/data /user/hadoopuser/assignment1
hadoop fs -copyFromLocal Downloads/example_index /user/hadoopuser/assignment1
```

You can visualize the HDFS structure, including the data and example_index folders, by navigating to hadoop explorer.

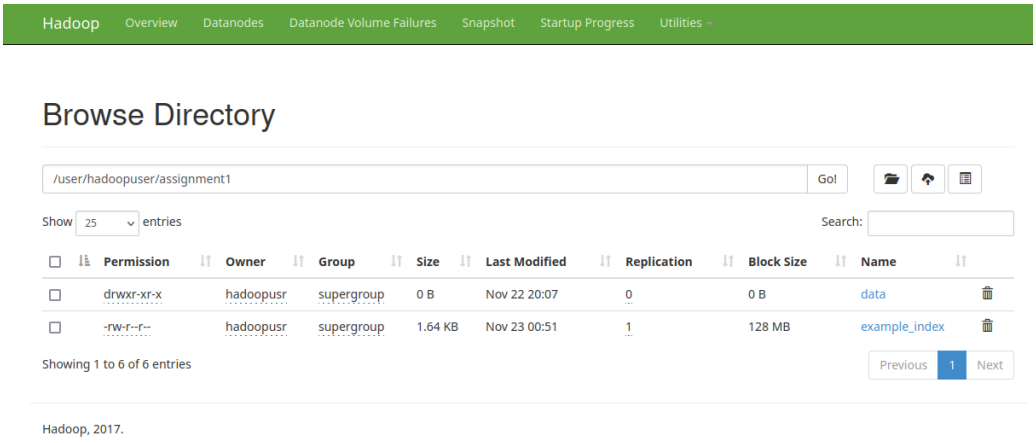

Figure 1: Hadoop Explorer

4 MapReduce Implementation

4.1 Mapper: `mapper_wdc.py`

The Mapper script reads input lines from the standard input (STDIN), which is typically the output of a previous MapReduce stage. Each line represents a directed edge in the graph, with the origin and target nodes separated by a tab. The script extracts the origin node and prints key-value pairs where the key is the origin node, and the value is set to 1. This is the initial step in counting the outlinks for each domain.

Listing 1: Mapper for the Origin Node

```
#!/usr/bin/env python3
"""mapper_wdc.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    if not line:
        continue

    # Split the line into values using tab as the delimiter
    values = line.split('\t')
    origin = values[0]
    target = values[1]

    # Output key-value pairs for the origin node
    print('%s\t%s' % (origin, 1))
```

To validate the Mapper, execute the following command:

Listing 2: Validate Mapper Output

```
hadoop fs -cat /user/hadoopuser/assignment1/data/* | ./mapper_wdc.py
```

If a permission error is encountered, change the permission of the Mapper script:

Listing 3: Change Script Permission

```
sudo chmod 777 mapper_wdc.py
```

The output of the Mapper should resemble the following:

Origin Node	Count
7	1
7	1
7	1
7	1
7	1
...	...
105	1

Table 3: Sample data from the Mapper output.

Repeat a similar process for the target by modifying, we will just change what we return

Listing 4: Modification for Target Mapper

```
print('%s\t%s' % (target, 1))
```

this line statement outputs key-value pairs to the standard output. that will then be used by the reducer.

Once again validate the output of the Mapper through executing the following command:

Listing 5: Validate Mapper Output

```
hadoop fs -cat /user/hadoopuser/assignment1/data/* | ./mapper_target.py
```

The output of the Target Mapper should resemble the following:

Origin Node	Count
5	1
6	1
9	1
10	1
12	1
...	...
25	1

Table 4: Sample data from the Target Mapper output.

4.2 Reducer: `reducer_wdc.py`

The Reducer script receives key-value pairs from the Mapper, where the key is the node (domain) and the value is the count of occurrences from the Mapper. The script processes these pairs and accumulates the counts for each node. It takes advantage of the fact that Hadoop sorts the input by key before passing it to the Reducer.

Here's a step-by-step explanation:

- **Initialization**: Initialize variables to keep track of the current node (‘current_node‘) and its count (‘current_count‘).
- **Input Processing**: Iterate through the input received from the Mapper. Each line is split into the node and count. The count is converted to an integer.
- **Count Accumulation**: If the current node is the same as the node from the input, accumulate the count. If the nodes are different, check if there was a previous node. If yes, write the result to STDOUT and update the current node and count for the new node.
- **Final Output**: Ensure the last node's count is outputted if needed.

This Reducer essentially sums up the counts for each node, providing the total count of links for each domain. Since it is just summing up, we just need to implement a signle reducer for the **Origin** and the **Target**.

Listing 6: Reducer for Counting Nodes

```
#!/usr/bin/env python3
"""reducer_wdc.py"""

from operator import itemgetter
import sys

# Initialize variables to keep track of the current node and count
current_node = None
current_count = 0
node = None

# Input comes from STDIN
for line in sys.stdin:
    # Remove leading and trailing whitespace
    line = line.strip()

    # Parse the input received from the mapper
    node, count = line.split('\t', 1)

    # Convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # Count was not a number, so silently ignore/discard this line
        continue

    # Check if the current node is the same as the node from the input
    if current_node == node:
        # If yes, accumulate the count
        current_count += count
    else:
        # If not, check if there was a previous node
        if current_node:
            # If yes, write the result to STDOUT
            print('%s\t%s' % (current_node, current_count))
        # Update current node and reset count for the new node
        current_count = count
        current_node = node

# Do not forget to output the last node if needed!
if current_node == node:
    print('%s\t%s' % (current_node, current_count))
```

Lets run the map reduce as a whole and save the output in dataoutput.

Listing 7: Run Mapreduce for Origin(Outlinks)

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar
-file /home/hadoopusr/mapper_wdc.py -mapper /home/hadoopusr/mapper_wdc.py
-file /home/hadoopusr/reducer_wdc.py -reducer /home/hadoopusr/reducer_wdc.py
-input /user/hadoopuser/assignment1/data/*
-output /user/hadoopuser/assignment1/dataoutput
```

Doing the same to the Taget nodes(**Inlinks**)

Listing 8: Run Mapreduce for Target(Inlinks)

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar
-file /home/hadoopusr/mapper_target.py -mapper /home/hadoopusr/mapper_target.py
-file /home/hadoopusr/reducer_wdc.py -reducer /home/hadoopusr/reducer_wdc.py
-input /user/hadoopuser/assignment1/data/*
-output /user/hadoopuser/assignment1/target_output
```

Lets have a look on a block diagram, to know what we have accomplished, and what do we need to accomplish.

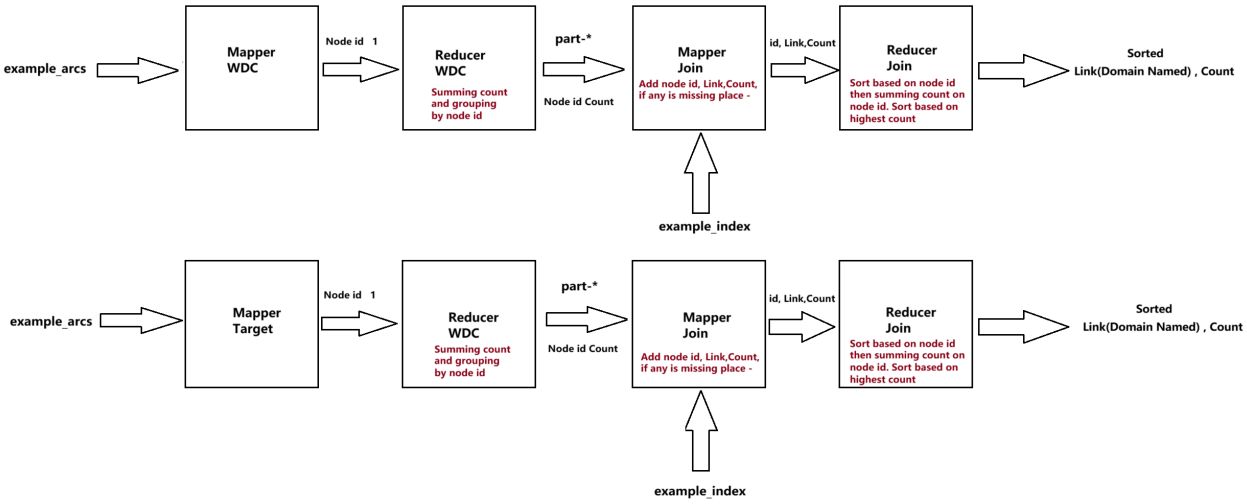

Figure 2: Hadoop Explorer

Outlink Node ID	Count
10	16
100	3
102	16
105	1
15	1
20	1
22	2
24	1
25	23
31	1
37	1
40	1
44	3
48	1
54	1
59	1
64	1
65	1
69	1
7	10
73	1
76	1
77	1
79	1
81	1
88	7
9	25
90	3
91	1
93	1
95	2
96	1
98	10

Table 5: Outlink Node Counts

Inlinks Node ID	Count	Inlinks Node ID	Count
0	1	1	1
10	2	100	2
101	1	102	3
103	1	104	1
11	1	12	3
13	1	14	1
16	2	17	1
18	1	19	1
2	1	21	1
22	1	23	1
25	14	26	2
27	1	28	1
29	2	3	1
30	1	32	3
33	1	34	1
35	1	36	1
38	2	39	1
4	1	41	1
42	1	43	2
44	3	45	1
46	1	47	1
48	3	49	1
5	3	50	1
51	1	52	2
53	1	55	1
56	1	57	1
58	1	6	3
60	1	61	1
62	1	63	1
66	1	67	1
68	1	7	2
70	3	71	1
72	1	73	1
74	1	75	1
78	1	79	1
8	3	80	1
81	1	82	3
83	1	84	1
85	1	86	1
87	1	88	3
89	2	9	2
90	2	91	1
92	1	94	1
95	4	97	1
98	2	99	1

Table 6: Inlink Node Counts

5 Map Reduce for Join

The current task aims to incorporate an index file to the output, providing domain names along with counts. The index file includes mappings of node IDs to domain names.

5.1 Mapper: `mapper_join.py`

Listing 9: Mapper Code

```
#!/usr/bin/env python3
"""mapper_join.py"""

import sys

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # split the input line into key and value
    data = line.split('\t')
    if len(data) == 2:
        key, value = data
        if key.isdigit():
            link_id = key
            sum_result = value
            link = '-'
        else:
            link = key
            link_id = value
            sum_result = '-'
        print('%s\t%s\t%s' % (link_id, link, sum_result))
    elif len(data) == 1:
        print('%s\t%s\t%s' % (data[0], '-', '-'))
```

5.1.1 Explanation

The mapper processes input data and extracts link ID, link, and sum result. If the key is a digit, it is considered a link ID, and the link is set to '-'. If the key is not a digit, it is considered a link, and the link ID is set to '-'. The output includes link ID, link, and sum result.

5.2 Reducer: `reducer_join.py`

Listing 10: Reducer Code

```
#!/usr/bin/env python3
"""reducer_join.py"""

import sys

current_key = None
current_link = None
current_sum_result = 0

# Create a list to store records
```

```
records = []
outputs = []
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # split the input line into key, link, and sum_result
    splitted = line.split('\t', 2)
    if len(splitted) == 3:
        records.append(splitted)

# Sort the records based on the id
records.sort(key=lambda x: x[0])

# Process the sorted records
for record in records:
    key, link, sum_result = record

    # convert sum_result to int
    if sum_result.isdigit():
        sum_result = int(sum_result)
    else:
        sum_result = 0

    # process the input
    if current_key == key:
        current_sum_result += sum_result
        if link != '-':
            current_link = link
    else:
        if current_key:
            # write result to STDOUT
            outputs.append([current_link, current_sum_result])
            #print('%s\t%s' % (current_link, current_sum_result))
        current_key = key
        current_link = link
        current_sum_result = sum_result


if current_key == key:
    outputs.append([current_link, current_sum_result])

outputs.sort(key=lambda x: x[1], reverse=True)
for output in outputs:
    print('%s\t%s' % (output[0], output[1]))
```

5.2.1 Explanation

The reducer sorts the records based on the key and processes them sequentially. It accumulates the sum results for the same key and outputs the final results, including the link and sum.

Lets run the map reduce as a whole and save the output for the **outlinks** in joined_output.

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar
-files /home/hadoopusr/mapper_join.py,/home/hadoopusr/reducer_join.py
-mapper "/usr/bin/env python3 mapper_join.py"
-reducer "/usr/bin/env python3 reducer_join.py"
-input /user/hadoopuser/assignment1/dataoutput/part-* /user/hadoopuser/assignment1/example_index
-output /user/hadoopuser/assignment1/joined_output
```

Lets run the map reduce as a whole and save the output for the **inlinks** in joined_target_output.

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar \
-files /home/hadoopusr/mapper_join.py,/home/hadoopusr/reducer_join.py
-mapper "/usr/bin/env python3 mapper_join.py"
-reducer "/usr/bin/env python3 reducer_join.py"
-input /user/hadoopuser/assignment1/target_output/part-* /user/hadoopuser/assignment1/example_index
-output /user/hadoopuser/assignment1/joined_target_output
```

Lets check our hadoop explorer. hadoop explorer.

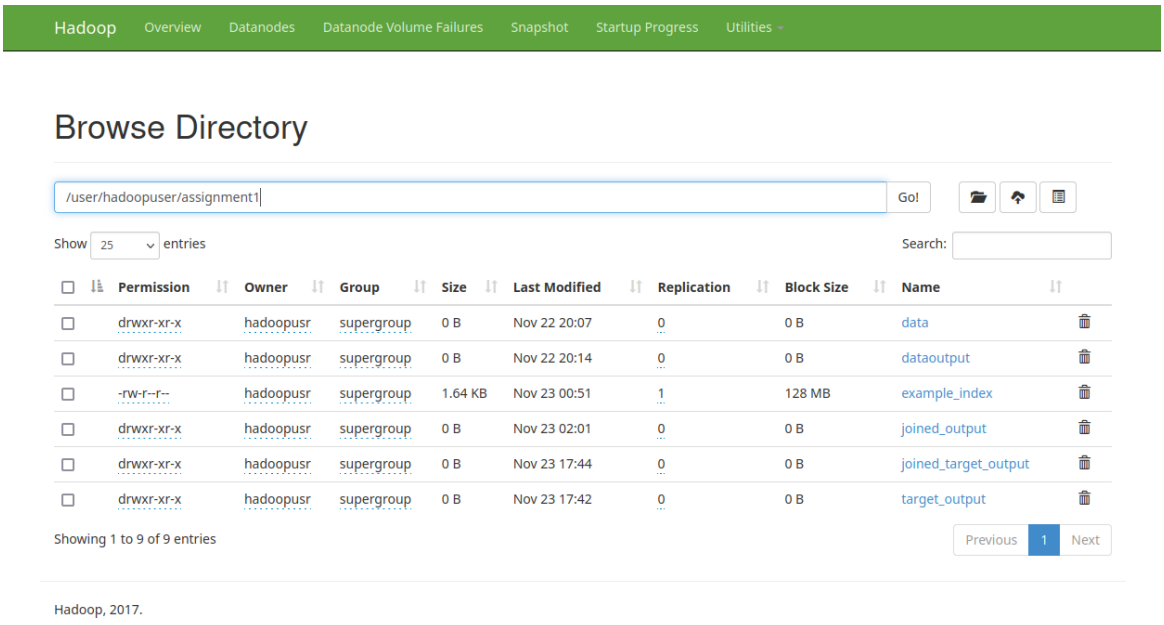

Figure 3: Hadoop Explorer

Now based on the tables we extracted, lets answer those questions:

- How many outlinks does every domain have? : **In the table.**
- How many inlinks does every domain have? : **In the table.**
- Which domain has the most outlinks? : **amazon.com**
- Which domain has the most inlinks? : **blogspot.com**

Domain Name	Count	Domain Name	Count	Domain Name	Count
amazon.com	25	blogspot.com	23	amazon.de	16
youtube.com	16	amazon.co.jp	10	wordpress.com	10
tumblr.com	7	yahoo.com	3	flickr.com	3
typepad.com	3	blogalaxia.com	2	wikipedia.org	2
zoomblog.com	1	animationplayhouse.com	1	azspot.net	1
blogia.com	1	classicalvalues.com	1	downthisvideo.com	1
eltangoysusinvitados.com	1	google.com	1	img-dpreview.com	1
kenyaunlimited.com	1	memeorandum.com	1	mesvilaweb.cat	1
mu.nu	1	over-blog.com	1	pjmedia.com	1
qwika.com	1	realestate.com.au	1	sapo.pt	1
vodpod.com	1	wikidict.de	1	wiktionary.org	1
1000notes.com	0	100500.tv	0	yahoo.com.au	0
yoyo.com	0	zappos.com	0	amazon.es	0
amazon.fr	0	amazon.it	0	angrybirds.com	0
apple.com	0	audible.com	0	audible.de	0
automattic.com	0	abebooks.com	0	beautybar.com	0
blogg.se	0	bookdepository.co.uk	0	bookdepository.com	0
buyvip.com	0	canalblog.com	0	abebooks.de	0
carambatv.ru	0	creativecommons.org	0	cubicle17.com	0
cyprien.fr	0	diapers.com	0	disqus.com	0
dpreview.com	0	drdrum.com	0	amazon-presse.de	0
endless.com	0	fabric.com	0	facebook.com	0
flickr.net	0	gmpg.org	0	goodreads.com	0
histats.com	0	amazon.ca	0	hockeyfights.com	0
icecastles.com	0	imdb.com	0	imdb.de	0
instagr.am	0	javari.de	0	javari.jp	0
joblo.com	0	amazon.cn	0	linkwithin.com	0
livejournal.com	0	lockerz.com	0	lovefilm.de	0
minecraftforum.net	0	minimalist.co	0	moxnews.com	0
myhabit.com	0	ning.com	0	nytimes.com	0
perublogs.com	0	petervidani.com	0	rea-group.com	0
amazon.co.uk	0	rivals.com	0	shopbop.com	0
smallparts.com	0	soap.com	0	spreadshirt.com	0
technorati.com	0	tinyurl.com	0	twitter.com	0
wag.com	0	wikimedia.org	0	woot.com	0
wordpress.org	0				

Table 7: Outlinks Counts

Domain Name	Count	Domain Name	Count	Domain Name	Count
blogspot.com	14	wikipedia.org	4	youtube.com	3
amazon.fr	3	creativecommons.org	3	flickr.com	3
google.com	3	amazon.ca	3	amazon.cn	3
myhabit.com	3	amazon.co.uk	3	shopbop.com	3
tumblr.com	3	amazon.de	2	yahoo.com	2
apple.com	2	bookdepository.co.uk	2	canalblog.com	2
dpreview.com	2	facebook.com	2	imdb.com	2
amazon.co.jp	2	twitter.com	2	amazon.com	2
typepad.com	2	wordpress.com	2	1000notes.com	1
100500.tv	1	yahoo.com.au	1	yoyo.com	1
zappos.com	1	amazon.es	1	amazon.it	1
angrybirds.com	1	audible.com	1	audible.de	1
automattic.com	1	abebooks.com	1	beautybar.com	1
blogalaxia.com	1	blogg.se	1	bookdepository.com	1
buyvip.com	1	abebooks.de	1	carambatv.ru	1
cubicle17.com	1	cyprien.fr	1	diapers.com	1
disqus.com	1	drdrum.com	1	amazon-presse.de	1
endless.com	1	fabric.com	1	flickr.net	1
gmpg.org	1	goodreads.com	1	histats.com	1
hockeyfights.com	1	icecastles.com	1	imdb.de	1
instagr.am	1	javari.de	1	javari.jp	1
joblo.com	1	linkwithin.com	1	livejournal.com	1
lockerz.com	1	lovefilm.de	1	minecraftforum.net	1
minimalist.co	1	moxnews.com	1	ning.com	1
nytimes.com	1	over-blog.com	1	perublogs.com	1
petervidani.com	1	rea-group.com	1	realestate.com.au	1
rivals.com	1	sapo.pt	1	smallparts.com	1
soap.com	1	spreadshirt.com	1	technorati.com	1
tinyurl.com	1	vodpod.com	1	wag.com	1
wikimedia.org	1	woot.com	1	wordpress.org	1
zoomblog.com	0	animationplayhouse.com	0	azspot.net	0
blogia.com	0	classicalvalues.com	0	downthisvideo.com	0
eltangoysusinvitados.com	0	img-dpreview.com	0	kenyaunlimited.com	0
memeorandum.com	0	mesvilaweb.cat	0	mu.nu	0
pjmedia.com	0	qwika.com	0	wikidict.de	0
wiktionary.org	0				

Table 8: Inlinks Counts