CS 6240 Parallel Data Processing in MapReduce – 02
Homework 3
Dheeraj Joshi

Q1. Discuss if your pre-processing approach followed the steps outlined above and if your PageRank and Top-k programs implement the pseudo-code from the course modules. Be precise in referring to the exact pseudo-code in the module—there are different approaches and versions for both PageRank and Top-k. Briefly explain why you chose one version over the others. If you took a different approach, briefly explain it and show the pseudo-code. (15 points)

Preprocessing Stage Job:

Parsing Approach:

To parse the html pages in the given compressed Wikipedia.bz2 corpus, I have manipulated the given parser program to handle the following things:

1. Replace "&" with "&" to pass the XML parsing of records with the ampersand character.
2. Initialized a set of text values to store the unique out links for a given page.
3. Created a check for self linking pages

This parser is executed on each record of the corpus containing the page and its corresponding html. The records containing "~" and invalid XML records were pre-handled in the given parser for both the pages as well as links.

Mapper:

In order to handle nodes that are present in outlinks, but are not present in the set of pages available in the corpus, I have tweaked my mapper to emit the following:

1. The page encountered as the key, with its adjacencyListNode containing the page rank and the set of outlinks as the value.
2. Also, every node in the outlinks as a key and with a default adjacencyListNode object as value respectively.

Reducer:

Each reducer task then gets a set of adjacencyListNodes for one specific page. The reducer just accumulates the set of outlinks for each page and emits the page name as key and the updated adjacencyListNode object as value.

CS 6240 Parallel Data Processing in MapReduce – 02
Homework 3
Dheeraj Joshi

PageRank Calculation Stage Job (Solution 2 Technique):

The solution 2 technique from the modules ensures that minimum jobs are required for accumulating delta in the map phase for the next iteration and performing the new PageRank calculation in the reduce phase of current iteration.

To calculate the PageRank values, from the set of pages and their adjacencyListNodes emitted by the parser job, we do the following steps in the every iteration:

Mapper:

For every page in the 1st iteration, emit the page, along with the adjacencyListNode updating it's default page rank value as 1/|total number of pages|.

For the following consecutive iterations, emit the page, along with the adjacencyListNode consisting of it's new page rank value as calculated in the reducer of the previous iteration.

Along with this, we increment the counter of the current job for delta value to be used in the next iteration by the current page rank of the page in context.

If the page has a list of outlinks, emit each page in that set as key and with a page rank value as the adjacencyListNode. PageRank/|total number of pages in the outlinks set|. Note that, this isn't a node, it's a double value emitted.

Note: The "iterationCount" and "total number of pages" (pageCount) are set in the configuration

Reducer:

Each reducer task gets a page and it's corresponding adjacencyListNode object as well as inlink contributions. Based on the type of value, we either set the adjacencyListNode object or accumulate the inlink values for the new PageRank calculation.

After iterating over all values, we use a hopping rate of alpha (initialized as 0.15) for distributing the probability of arriving at the page directly or through a random hop. We then use the accumulated inlinks PageRank values in the formula:

S = accumulated sum of inlinks

D = accumulated delta value (0 in the 1st Iteration)

V = total number of pages

A = alpha rate

newPageRank = A/V + (1-A)(S + D/V)*

After the calculation, we set the page rank in the adjacencyListNode to this new page rank value and emit the page as key along with this updated adjacencyListNode object as value.

Top K Calculation Stage Job (K=100):

After receiving the list of pages with their converged page ranks and set of outlinks, we use the following approach with null keys and single reducer to get the top 100 pages based on their page rank values:

Mapper:

We parse every record to get the page name and its corresponding page rank value and keep a local top k copy in a TreeMap. (Note: TreeMap is a sorted HashMap based on key)

At any point in time while iterating over the pages, if we exceed the size of the local copy by k, we remove the first value (smallest value) from the TreeMap.

Finally, we emit all the k values from the TreeMap to send it to the reducers with a null key.

Note: In case of similar page ranks, we add the page to the text of pages with a "~" separator.

Reducer:

Due to a single reducer set and null as the key, all the values end up at the same reduce task. We iterate over all of the local top k values from every map task and keep a local top k copy. We do the same as the mapper, by removing the first (smallest) value from the TreeMap as soon as the size exceeds by k.

Finally, in the end, we emit all the values from our local top k copy in the descending order to the output.

Q2. Report the amount of data transferred from Mappers to Reducers, and from Reducers to HDFS, in each iteration of the PageRank computation.

6 m4.large machines:

Iteration No:	Data transferred from Mappers to Reducers	Data transferred Reducers to HDFS
1	1199056002	1128380802
2	1464328495	1130107219
3	1461366351	1128630095
4	1463075721	1130141523
5	1461748202	1130264374
6	1462486949	1130133981
7	1462066279	1130204141
8	1462194476	1130140160
9	1462225520	1130166598
10	1462224003	1130137630
11	1462205326	1130155291

11 m4.large machines:

Iteration No:	Data transferred from Mappers to Reducers (in Bytes)	Data transferred Reducers to HDFS (in Bytes)
1	1295721312	1073775964
2	1565351072	1128380804
3	1562634191	1128630622
4	1564189513	1130141344
5	1562953031	1130264428
6	1563552555	1130134805
7	1563165441	1130204993
8	1563306984	1130139949
9	1563331130	1130166118
10	1563326184	1130137869
11	1563296325	1130154232

Q3. Does it change over time? If so, briefly discuss why or why not? (5 points)

The amount of data transferred from Mappers to Reducers in the 1st iteration for both versions is considerably lesser than the other iterations. This is because the page rank and inlink contributions passed along for every page is either 1/total number of pages or a fraction of it and therefore, in turn, lesser data is written to the HDFS by the reducers.

However, for the next consecutive iterations the number of bytes transferred remains consistent across all iterations for both versions as only the page rank values are updated for every page. As a result, the number of bytes written to HDFS by reducers also remain consistent after the 1st iteration in both versions.

Q4. Report for both configurations (i) pre-processing time, (ii) time to run ten iterations of PageRank, and (iii) time to find the top-100 pages. There should be 2*3=6 time values. (6 points)

Note: Time format (hh:mm:ss)

6 m4.large machines (1 master and 5 workers)

6 Machines: (1 Master 5 Slaves)	Start	End	Total Time
Pre-processing time:	19:36:52	20:18:32	0:41:40
Time to run ten iterations of PageRank:	20:18:33	20:44:33	0:26:00
Time to find the top-100 pages:	20:44:33	20:45:17	0:00:44

11 m4.large machines (1 master and 10 workers)

11 Machines: (1 Master 10 Slaves)	Start	End	Total Time
Pre-processing time:	0:49:15	1:10:45	0:21:30
Time to run ten iterations of PageRank:	1:10:46	1:27:50	0:17:04
Time to find the top-100 pages:	1:27:51	1:28:32	0:00:41

Speedup comparison:

Jobs	6 Machines	11 Machines	Speed Up
Pre-processing	0:41:40	0:21:30	1.937984496
PageRank Calculation x10	0:26:00	0:17:04	1.5234375
Top - K (K=100)	0:00:44	0:00:41	1.073170732

Q5. Critically evaluate the runtime results by comparing them against what you had expected to see and discuss your findings. Make sure you address the following question: Which of the computation phases showed a good speedup? If a phase seems to show fairly poor speedup, briefly discuss possible reasons—make sure you provide concrete evidence, e.g., numbers from the log file or analytical arguments based on the algorithm's properties.

The pre-processing phase shows a considerable speedup between the two versions. The 11 machine version performs the pre-processing in approx. 21 minutes compared to approx. 41 minutes taken by the 6 machine version. This is due to having 19 map jobs on an average for all iterations processing the entire data on a 11 machine version versus 10 map jobs on an average for the 6 machine version.

The time to run 10 iterations of Page Rank does show some adequate amount of speedup (26 minutes for 6 machine version vs 17 minutes for 11 machine version). This is largely due to more number of map and reduce tasks performed in parallel due to the availability of more machines. But this gain is somewhat adjusted with the greater amount of shuffles required (almost double of 5 machine version) for accumulating similar inlink contributions and node object across reducers.

The speedup for top k program across two versions is really poor (44 seconds for 6 machine version vs 41 seconds for 11 machine version) because of the constant number of input records from the reducer phase of last page rank iteration and a single reducer task configured. This means that, there was no parallel execution at the reduce phase. Also, the map and reduce tasks only keep "k" (100 in this case) records locally across both versions.

Q6. Report the top-100 Wikipedia pages with the highest PageRank's, along with their rank values and sorted from highest to lowest, for both the simple and full datasets. Do they seem reasonable based on your intuition about important information on Wikipedia? (5 points)

From my observations across the top-100 pages with the highest page rank's for both versions of the Wikipedia data, it seems that the majority of the Wikipedia pages contain geographic information (e.g. Countries and cities like United States, New York city, etc.)

This majority is followed by the time of an event (e.g. World War II) generally specified by year of occurrence (e.g. 2000, 1996, 2006).

The other pages observed relate to translation or languages mentioned in the pages for e.g. Latin, French, etc.

Considering all these observations, I do assert that Wikipedia pages hold majority information about these domains and therefore, are higher ranked pages in the corpus.