Assignment № 3

Arun K Rajagopalan, Texas A&M University 11/09/2015

Problem

Apply MapReduce to graph relabeling and shows the benefits of doing so.

Analysis

The problem can be split into 3 stages -

1. Graph Inversion 1
2. Graph Inversion 2
3. Compute TopN

Each of these stages have been designed to operate independently of the previous stage. By treating each stage as a black-box, I was able to efficiently debug and ensure correctness.

Graph Inversion

Algorithm

I present an overview of the Algorithm at a high level.

- The graph inversion step 1 and step 2 are basically the same, with minor modifications. Thus, I templatized the graph inversion. It takes as template an argument that specifies the size of the vertex node (either uint64 or uint32). Specializations are provided by means of a boolean flag to differentiate the two stage.
- The inversion proceeds under the assumption that the nodes of the graph arrive in a sorted order.
- We read the nodes and relabel either from the previous run or sequentially depending on whether it is the first run or not.
- For each node we relabel, we map it into a buffer inverted.
- The buffer performs sort-compact.
- We then write the buffer to disk.
- We then need to merge the split files. This is similar to a merge of two sorted arrays.
- Finally, we end up with an output file that is inverted.

Optimizations

There are multiple avenues of optimization that I looked into.

- While mapping the inverted nodes, I send it into 8 buckets, so that I can do a parallel sort. In the first phase, I use the top 3 bits for this. In the second inversion, I cannot do this since the nodes are no longer properly distributed. Instead, I partition on the range of the input.
- The merge is performed as a 4-way merge executed in 2 threads (effectively, 8 files are being merged concurrently).
- To maintain the file handle count, I use a fileHandle map inside each thread that keeps the total count to 2 per thread plus one shared write handle between the two threads.
- I used un-buffered reads to get maximum read performance.

Areas of future improvement

While I was able to achieve good speedup with the above mentioned techniques, these we some of the areas I would have also liked to improve.

- Asynchronous reads will give me much better performance.
- The adjacency list sort in the final step can be distributed to a thread pool. I got a prototype working, but it was consuming too much memory.
- I am missing a few nodes from the final count. This could possibly be due to some data getting cut off at a buffer overrun.

TopN

Algorithm

I present an overview of the Algorithm at a high level.

- Since the relabelled graph now fits in memory, we can directly increment the counter for each node as we see them.
- Once this is done, we create a priority queue of size 10. This queue holds the top 10 elements in the array as we scan it.

Non-trivial and interesting points

- memmove should be used instead of memcpy if the dst buffer overlaps with source.

Statistics

Following is the output when run on my personal Macbook Pro (Intel Core i5-4278@2.6GHz, 8GB RAM, SSD) running Windows 10.

```
Starting 1st Inversion
Total IO: read - 53.40 GB; write - 47.13 GB
Took 252 seconds
Ending 1st Inversion

Starting 2nd Inversion
Total IO: read - 39.64 GB; write - 38.60 GB
Took 272 seconds
Ending 2nd Inversion

Overall stats - RunTime: 525 seconds;
Total read 93.03 GB; Total write 85.73 GB
Peak working set : 499.70 MB
DONE!

-----------------


Starting Top 10

TOP 10 Node count

1. 24247151 2947626
2. 21153020 2209833
3. 17991418 1996682
4. 13620962 1287415
5. 14499776 1203819
6. 3229951 1031992
7. 173482 932753
8. 14715032 932356
9. 9904332 803930
10. 18092264 744919

Total sum Y_i(Y_i-1)/2 = 3144604790303
Total IO: read - 7.41 GB; write - 0.00 GB
Took 53 seconds
Ending Top 10
```

```
Overall stats - RunTime: 53 seconds;
Total read 7.41 GB; Total write 0.00 GB
Peak working set : 459.32 MB
DONE!
```

Top 10 links

```
1 microsoft.com 2947630
2 google.com 2209834
3 yahoo.com 1996682
4 adobe.com 1287417
5 blogspot.com 1203819
6 wikipedia.org 1031992
7 geocities.com 932753
8 w3.org 932359
9 msn.com 803930
10 amazon.com 744919
```

ATTRIBUTION

I took help from the following sites to help with my assignment

1. stackoverflow.com
2. MSDN
3. sparetimelabs.com/printfrevisited