Sepehr Bakhshi

Fall 2022

Introduction:

Our goal is to simulate the concepts of term-based and document-based partitioning. The implementation should be able to assign nodes for processing the queries that it receives from the user using a broker unit. I use python programming language for this purpose.

Initially, I explain the preprocessing of the terms and queries. Next, the implementation details of the simulation are explained. Then the results which are the costs for both the document-based and term-based partitioning is provided. Finally, a conclusion is made based on the results.

Preprocessing:

Preprocessing is applied on two sets. First, the terms are preprocessed and only the PLL and the term itself are kept. Then for queries, the additional numbers are removed and the queries are added to a list. Each query now has a list of terms and these terms are sent to broker to search them in the nodes.

Implementation Details:

For simulation, I used the principles of object-oriented programming as follows:
I have two classes named Node and Broker. The node class keeps the term and PLL as its attributes and the broker class keeps these nodes in an array. In this simulation, we only have one Broker object and ***K*** nodes (K = 4, and K = 32 in our case).
For keeping the items in the Node class, I use a dictionary. Dictionary is more efficient for retrieving items and searching. My search function is placed inside the broker. The broker keeps the node objects and retrieves their data and searches in them for the possible occurrence of any term in the query.
The procedure of the approach is shown in Figure 1.

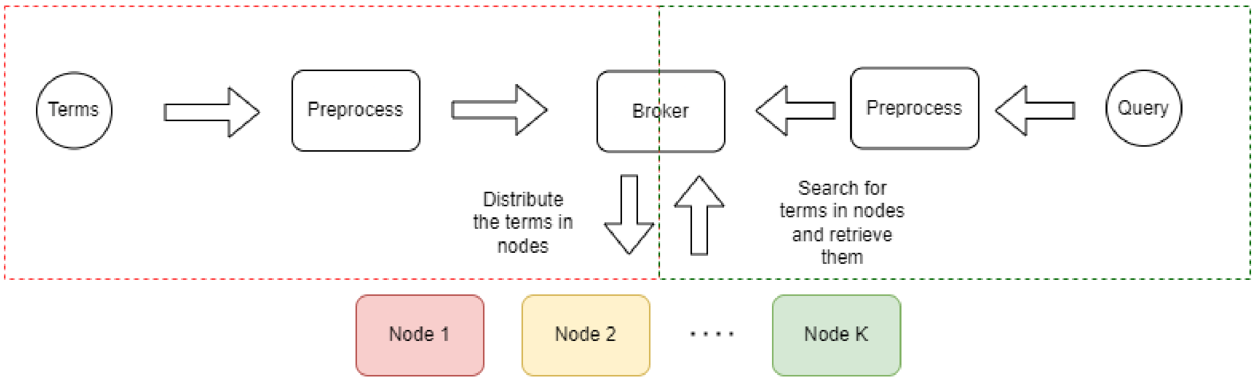
Terms
Preprocess
Broker
Preprocess
Query
Distribute the terms in nodes
Search for terms in nodes and retrieve them
Node 1
Node 2
. . . .
Node K

Figure 1 Term-based partitioning procedure. The left red-dashed box is responsible for distributing the terms. The green-dashed box in the right is responsible for searching the terms in the nodes and retrieving it.

A similar procedure is done for document-based partitioning. In this mode, we can only use one node and broker doesn't have a cost. Based on this reason we neglect it in both the calculation and implementation.

Finally, the cost of each node and the broker is calculated based on the instructions in the assignment file.

Results and discussion:

Term-based Partitioning Results:

For term-based partitioning, I test it on 4,32, and 128 nodes. Although the 128-node size is not included in the assignment file, out of curiosity I wanted to check if adding more brokers would affect the distribution of the node costs after a threshold or not. Below are the results of the experiments:
Query count: 9797
Overall cost (k=4): 1,387,090,520.0
The overall cost per query (average overall cost k=4): 141,583.19
For calculating the total cost, the average cost should be multiplied by the query count which is 9,797. Some queries are not considered as their terms are not included in the wordlist dataset.

Table 1 Average cost per query for nodes and the broker (K=4)

	Broker	1	2	3	4
Cost	46421.76	21616.33	17082.18	45231.39	44158.84

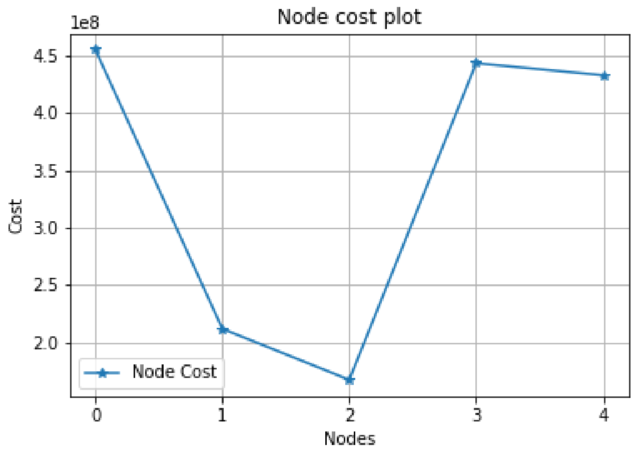

Figure 2 Total cost for nodes and the broker. Broker cost is in the node number 0 (k=4).

Table 2 Average cost per query for nodes and the broker (K=32)

	Broker	1	2	3	4
Cost	109,791.62	423.45	334.33	527.39	666.01

	5	6	7	8	9
Cost	12,484.22	1,032.52	1,099.85	1,112.49	2,821.41

	10	11	12	13	14

Cost	993.92	1,115.04	17,407.69	493.14	908.06

	15	16	17	18	19
Cost	13,709.92	18,057.86	2,101.13	413.02	25,966.93

	20	21	22	23	24
Cost	4,260.96	1,617.19	6,931.85	1,211.63	692.02

	25	26	27	28	29
Cost	606.94	2,137.21	893.51	1,205.77	1,068.86

	30	31	32
Cost	4,331.28	707.13	756.04

Query count: 9797

Overall cost (k = 32): 1,914,310,186.00

Overall cost per query (average overall cost k=32):195,397.59

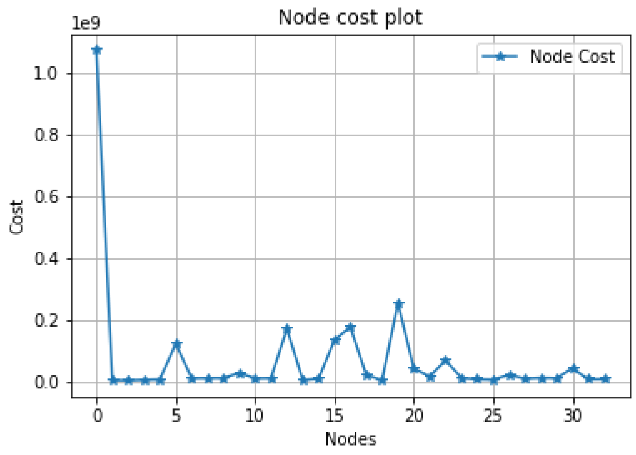

Figure 3 Total cost for nodes and the broker. Broker cost is in the node number 0 (k =32)

For the k =128, I do not report the numbers, however, we can see the cost at each node in the following Figure:

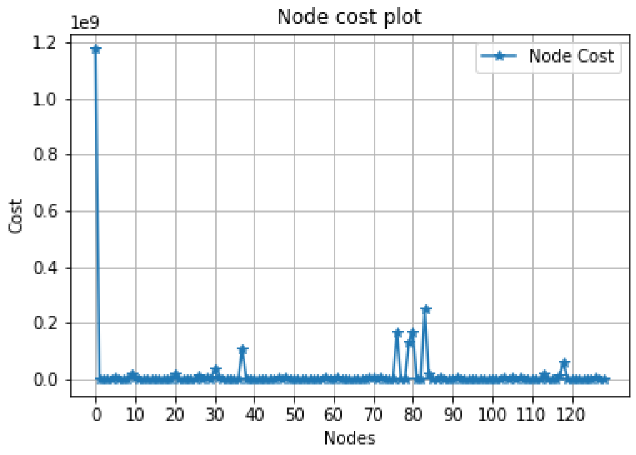

Figure 4 Total cost for nodes and the broker. Broker cost is in the node number 0 (k =128)

As we can see in the figures, the broker cost increases as we increase the number of nodes. On the other hand, by increasing the number of nodes, their cost decreases drastically. After a threshold, however, the node costs become stable and we see fewer changes in the node costs, but the broker cost still increases even with node sizes equal to 100.

Another interesting factor that I realized in the evaluation results was that some nodes have exceptionally higher costs compared to others ones. For instance, when K = 32, node number 12,15,16,19 have high costs compared to other ones. The reason is that they contain some terms that have extremely high PLLs. When such a term appears in the query and the node that contains the high-PLL term, then its cost increases drastically. To alleviate this problem, an idea came to my mind. In the round-robin assignment of the terms, I assign a ceiling limit for each node, and until it is not filled with enough terms, and the ceiling is not achieved, I keep adding more terms to that node. With this technique, we can see a little bit of improvement, but still, as some terms have high PLLs, this solution can not solve the problem completely.

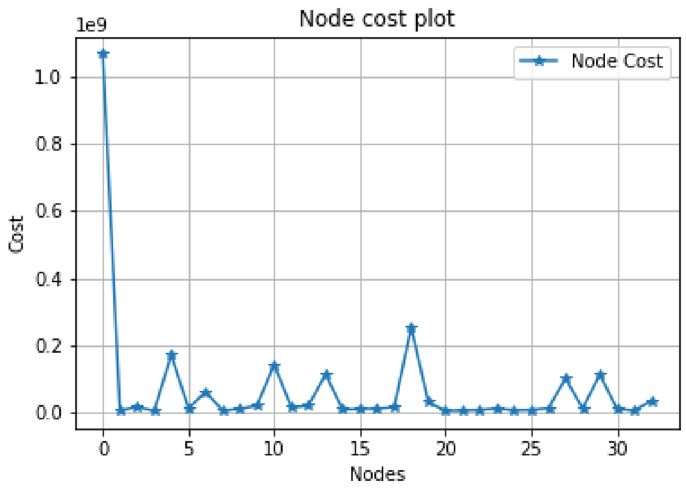

Figure 5 Total cost for nodes and the broker. Broker cost is in the node number 0 (k =32) with ceiling limit equal to 1000.

As we can see by comparing Figure 5 and Figure 3, the node costs are more diverse and distributed.

Document-based Partitioning Results:

Overall cost (k=4): 324,874,859.0

Overall cost per query (average overall cost k=4): 33,160.65

Average Node cost: 32,023.68

Overall cost (k=32): 50,533,398.0

Overall cost per query (average overall cost k=32): 5,158.04

Average Node cost: 4,004.70

Discussion:

Initially, I compare the term-based and query-based partitioning results. By looking at the overall cost, my results show that document-based partitioning has less cost. One reason is that the broker cost is zero in document-based retrieval and this reduces the overall cost. Another reason is that terms with large PLLs are distributed between the nodes and this results in less burden on each node. This is more obvious when we have a larger number of nodes (K = 32). One interesting point that I noticed in my experiments was the differences in the runtime. Although I am using the same code for searching the nodes, the runtime for the document-based mode is more, as it has a greater number of terms in it. This higher runtime is caused by the higher number of terms in each node in the document-based retrieval.

Conclusion:

I implemented the document-based and term-based partitioning in a simulated environment. The results show that the document-based partitioning reduces the overall cost. However, by increasing the number of nodes, the high cost of each node is reduced to a more reasonable cost. Besides, adding more node may result in more runtime specially in document-based partitioning.