CS39002 - OPERATING SYSTEMS LABORATORY

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Laboratory Assignment 3

Authors:
Abhay Kumar Keshari (20CS10001)
Aniket Kumar (20CS10083)
Jay Kumar Thakur (20CS30024)
Tanmay Mohanty (20CS10089)

Date: February 27, 2023

Design Strategy for the Optimisation

Observations

1. Repeated use of Dijkstra Algorithm will recompute distances of a lot of points which has been redundantly computed in the earlier distance matrix.

2. There are multiple nodes being removed or added to the mapped set of nodes of a particular consumer, considering the division by $1/10^{th}$ of the total each time.

3. Majority of the updates lie in the regions around the new nodes.

4. Occasionally a consumer is invoked before the producer updates the previous graph.

5. The maximum distance is observed to be about 5-6.

Solution

1. For the first iteration, run the Multi-source Dijkstra algorithm to compute the initial distances and paths from the source to the destination nodes.

2. For the upcoming iterations, there are two cases:

   - The consumer is called, before any updation of the graph by the producer.
   - The consumer is called, after the updation of the graph by the producer.

3. For the first case, we do not need to update the distance matrix.

4. For the second case, we evenly distribute the new nodes among the consumers, based on $New\_Node\_ID \ mod \ 10 == Consumer \ ID$.
   Use constrained BFS only to update those nodes which are affected by the new source nodes. Thus iterating through the entire graph is not required.
   Affinity of a new node is defined to be the number of nodes in the previous graph, to which it is connected.
   Heuristic: Sort all the non-source nodes based on their computed affinity.
   Iterate through all these nodes and perform the following:

   (a) The distance of the new non-source node is one hop more than the distance of its nearest neighbor.

   (b) Perform constrained BFS on neighboring nodes to propagate the effect of the above step.

Pseudo-code

```
require: graph
require: distance[]
require: dijkstra(graph)
require: constrainedBFS(nodes)

if distance[] is not computed:
  dijkstra(graph)

else:
  if new nodes are inserted:
    source_nodes <- new nodes which are in the mapped set
    non_source_nodes <- new nodes which are not in the mapped set

    constrainedBFS(source_nodes)
    sort non_source_nodes in decreasing order of affinity
    for node in non_source_nodes:
      constrainedBFS(node)
```

Results of the optimisation

The following results are obtained after 10 minutes of execution:

Process No.	**average CPU-time (μs)**	**Optimised average CPU-time (μs)**
1	9119.55	516.864
2	7443.36	614.364
3	6510.09	714.682
4	9844.55	758.455
5	7476.50	644.136
6	9796.14	734.409
7	8186.05	934.682
8	9330.82	956.545
9	8841.09	843.682
10	9564.68	1520.000
Average	**8611.2830**	**823.8459**

Optimization factor: 10.45