Selection of optimal path finding algorithm for data transmission in Distributed Systems

Zakhar Mozil, Yevhenii Vavruk

1. Computer Engineering, Lviv Polytechnic National University, UKRAINE, Lviv, Stepana Bandery 28A, wormik358@gmail.com
2. Computer Engineering, Lviv Polytechnic National University, UKRAINE, Lviv, Stepana Bandery 28A, evv1@ukr.net

Abstract - Considered typical structure of the distributed system. Also considered problem of data transmission in a multilevel distributed system. The criteria for optimal pathfinding algorithm are selected and the optimal algorithm is chosen. There is described Bellman-Ford algorithm for finding the path in the graph.

Keywords – shortest pathfinding problem, optimal pathfinding, distributed system, graphs, Bellman-Ford algorithm.

Introduction

One of the main tasks that are solved when designing multi-level distributed data processing systems is to provide the speed of exchange between the nodes of the system. The speed depends on the structure of the system, data transfer protocols and selected algorithms for information transmission. Moreover, system performance depends on chosen pathfinding algorithm.

Therefore, the problem of choosing an algorithm which provide the best way to transmit data is relevant. For this purpose in the article are considered basic pathfinding algorithms in graphs, the basic criteria are determined and the optimal algorithm is chosen.

Distributed System structure designing

So, as the problem of finding an optimal path does not exist in one-level distributed systems - the structure of the system should be multi-level. Let's assume that the weight of the graph is the delay of data transfer between the nodes (one byte data transmission time (ms)), this will allow to determine the optimal path, even consider on the nodes that are overloaded.

Designed structure is multi-level, it has 3 levels. So, any node pair has path between no more than 2 intermediate nodes. In general, the system has 8 nodes. Designed structure is shown in Fig. 1.

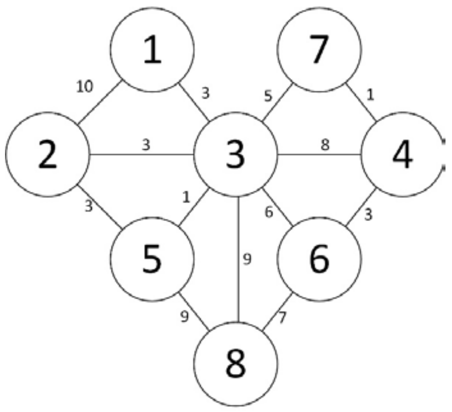

Fig. 1. Designed structure

Pathfinding algorithms

There is a generalized algorithm for finding the optimal path in the graph, the essence of which is as follows: in the weighted graph all the possible combinations of the passage from the initial to the final node are selected, after which for each combination the total weight of all the edges is determined and between all combinations is chosen the one which has the least weight. .

Using a generalized algorithm is not an optimal solution, because of its complexity of calculations. To find the optimal path in the graph, there are other algorithms. A list of algorithms for finding the optimal path and the tasks they solve are presented in Table. 1
All algorithms for finding the optimal path are divided into three categories:

- determining the optimal path between a pair of nodes;
- definition of the optimal path between all pairs of nodes;
- determining the optimal path between a given node and all other nodes.

Table 1

List of algorithms for finding the optimal path in the graph

Name	**Problem**
Dijkstra	Single source
Bellman-Ford	Single source. Weights can be negative
A*	Node pair
Floyd-Warshall	All node pairs
Johnson	All node pairs. Weights can be negative
Terry	Node pair
Lee algorithm	Node pair
Dantzig algorithm	All node pairs

Criteria for optimal pathfinding algorithm selection

The time complexity of algorithms for finding the optimal path depends on the number of nodes in the graph; accordingly, algorithms for finding the optimal path between all pairs of nodes are not suitable because of the use of a large amount of memory for storing data about paths between nodes and longer time calculations than in other algorithms.

Some algorithms for finding the optimal path on one pairs of nodes use a heuristic function to reduce the time complexity of the algorithm, but algorithms that use the heuristic function do not guarantee that the path between nodes will be optimal. Accordingly, the use of such algorithms is impossible, since it is necessary to calculate the optimal path for data transmission in the distributed system.

As previously mentioned: the weight of a node is a delay in data transmission between adjacent nodes, so the algorithm should work with graphs in which the scales between adjacent edges may differ significantly.

In addition to the above criteria, you also need to have the ability to process the nodes independently of each other, to speed up the search speed of the optimal path. That is, the ability to process a node should not depend on the results of previously processed nodes.

Thus, the search algorithm for the optimal path must meet the following criteria:

1. Solve the problem of finding an optimal path with one pair of nodes or between a given node and all other nodes;
2. Do not use heuristic functions;
3. Have the ability to work with graphs in which the weights of adjacent edges can vary significantly;
4. The ability to process a node should not depend on the results of previously processed nodes.

Optimal pathfinding algorithm selection

Taking into account the above criteria for selecting the algorithm, the list of algorithms from Table. 1 can be reduced.

Such algorithms as: Floyd-Worceshl, Johnson and Danzig do not meet the first criterion.

The algorithm A * uses a heuristic function, therefore, does not meet the second criterion.

The wave algorithm can only work with graphs that have a single length, respectively, the algorithm does not meet the third criterion.

The processing of the next node in the Terry algorithm depends on the result of the previous processed node, so the algorithm does not meet the fourth criterion.

Thus, the following criteria are satisfied by the chosen criteria: Deikstri, Bellman-Ford. Unlike the Dietcaster algorithm, the Bellman-Ford algorithm admits negative weights of edges in a graph. In addition, the complexity of the Bellman-Ford algorithm is O (VE), when the complexity of the Deikstry algorithm in the classical realization is O (V^2E). Therefore, the Bellman-Ford algorithm was chosen to implement data transfer in a distributed system.

Bellman-Ford algorithm

The algorithm uses the principle of relaxation, which consists in the fact that each time the node is processed, it records the distance to the nearest node, respectively, at the end of the algorithm, each passed node will contain a distance to the nearest neighbor node.

Let's find the path from point S to point F. The algorithm processes each node in the graph. The node that processes the algorithm is V. The algorithm chooses a node that is adjacent to the current node, denote it U. After that, the relaxation process takes place: if distance (S, V) + distance (V, U) <distance (S, U) , then there is an update of the distance from S to U. distance (S, U) = distance (S, V) + distance (V, U). The Bellman-Ford algorithm guarantees the optimal path to each graph node for V-1 iterations, where V is the number of vertices in the graph.

The pseudo-code for the algorithm is presented below.

```
function BellmanFord(list vertices, list edges, vertex source)::distance[], predecessor[]
  // graph initialization
  for each vertex v in vertices:
     distance[v] := inf
     predecessor[v] := null
  distance[source] := 0

  // relaxation on every vertex
  for i from 1 to size(vertices)-1:
     for each edge (u, v) with weight w in edges:
        if distance[u] + w < distance[v]:
           distance[v] := distance[u] + w
           predecessor[v] := u
    return distance[], predecessor[]
```

Conclusion

As a result of the analysis, the structure of the distributed system was developed. The criteria for selecting the optimal path search algorithm in the graph were selected. The Bellman-Ford algorithm is chosen to find the optimal way of transmitting data in a distributed system.

In further research it is planned to develop a graph generator according to the given criteria (number of nodes, number of levels of the system, etc.). Develop a Bellman-Ford algorithm based on CUDA technology. The constructed graphs are used to find the optimal path to the developed algorithm and to investigate the speed and memory sizes for different types of graphs.

References

[1] Pradesh M. Modified Dijkstra's Algorithm for Dense Graphs / Madhua Pradesh., 2016.

[2] Krianto S. O. Bellman Ford algorithm in Routing Information Protocol / Sulaiman Oris Krianto., 2017.