IJAMEC
e-ISSN: 2147-8228

INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS ELECTRONICS AND COMPUTERS

www.dergipark.org.tr/ijamec

International Open Access

Volume 08
Issue 04

December, 2020

Research Article

Distributed Environment Modeling using Path Compression Algorithm

Çiğdem Bakır [a, *], ***Veli Hakkoymaz*** [a]

[a]*Yildiz Technical University, Department of Computer Engineering, Istanbul, 34220, Turkey*

ARTICLE INFO

Article history:
Received 18 September 2020
Accepted 23 November 2020

Keywords:
Distributed environment
Object access
Object move
Path compression

ABSTRACT

In distributed environment, some operations related to objects are performed. For example, objects can be accessed or they can be moved. In our study, events related to objects (object-access, object-move) were created as independent events. In this study, the distributed environment simulation was performed and the effectiveness and success of the path compression algorithm, which we proposed as a result of the experimental study, was demonstrated. The purpose of this study is to show the effectiveness and benefits of the path compression algorithm. Path compression algorithm is an efficient algorithm whose runtime is linear. With the path compression, the long node chain that is formed while data objects are passing between the source node and the destination is broken, so that the objects are retrieved fast and the cost of access is reduced. This result is shown with experimental study by modeling the distributed environment. It is shown comparative the results of the distributed environment simulation according to the various Access/Move (%) rates using binomial distribution. When we use the path compression, the maximum length and mean length of the chain decreases. Thus, with the path compression algorithm, the long node chain created by the objects is broken, the cost of accessing the objects is reduced, and fast access to the objects is ensured. In short, with our study, fast access to data is ensured in a distributed environment.

1. Introduction

A distributed computing environment is performed with three types of nodes: storage, worker, and dissemination [1]. In a distributed environment, objects are subjected to a number of processes. For example, objects in the given data table can be queried, listed, deleted, updated, or new objects can be saved. In a distributed computing environment, objects can be kept in tables to perform these processes, and the processes related to the objects can be performed through these tables. In this study, in order for the processes related to the objects, the object-based distributed environment was modeled and the distributed environment simulation was carried out.

In this study, access and movement of objects, which are among these performed processes related to objects, are discussed. However, some problems arise during access to the object. It is because when we want to access an object, it is needed to look at all the nodes that the object passes until we reach the node where the object resides. Data and function transmission is performed from node to node. In this case, the cost of access increases because long chains are formed between the nodes during access to the object. In our study, this problem was addressed and by carrying out system modeling in the object-based distributed environment, the path compression algorithm providing effectiveness in access to the object was proposed. The aim of our study is to increase the performance through breaking the emerging long chains by the path compression algorithm. With the path compression, by breaking the long chain between the objects passing from the source node to the target node, quick access to the objects is enabled and the cost of access

* Corresponding author. E-mail address: *cigdem.bakr@gmail.com*
DOI: 10.18100/ijamec.797074

is reduced. In addition, through performing a distributed environment simulation, the effectiveness of the proposed algorithm was shown. With the conducted experimental study, the success of the path compression algorithm was also expressed.

The remaining parts of this study are arranged as follows: the related and similar studies are given in the 2nd part, our solution is discussed in the 3rd part, the experimental study is presented in the 4th part and the results are given in the last part.

2. Related Works

A distributed computing environment can be modeled with a graph data structure. A graph is a data structure consisting of a set of nodes and edges connecting these nodes to one another [2]. If G signifies a graph, its definition is as follows:

$$G = (V, E) \tag{1}$$

$$V(G) = \{v1, v2, \ldots, vN\} \tag{2}$$

$$E(G) = \{e1, e2, \ldots, eM\} \tag{3}$$

Hence,

$$E \subseteq VxV \tag{4}$$

The distributed environment is performed with three types of nodes: storage, worker, and dissemination [3]. The edge indicates the transition of a data object from one node to another. The storage node stores objects permanently. The dissemination node allows an object to be copied when the object is requested from itself. It checks whether the worker and the dissemination node have the authorization to import objects (with the privacy policy). The worker node runs programs. The dissemination node stores the frequently used objects in groups [4].

The distributed environment is performed with three types of nodes: storage, worker, and dissemination [3]. The edge indicates the transition of a data object from one node to another. The storage node stores objects permanently. The dissemination node allows an object to be copied when the object is requested from itself. It checks whether the worker and the dissemination node have the authorization to import objects (with the privacy policy). The worker node runs programs. The dissemination node stores the frequently used objects in groups [4].

In earlier studies in the literature [3,4,5], a distributed label model was developed. a separate label was used for each transaction (read, write) performed on the object, and only read and write transactions were carried out. We used the path compression algorithm to increase data access speed and reduca data cost. We showed the advantages and success of our proposed algorithm with experimental study. Since we used statistical approaches in distributed environment simulation, our study has been more successful than other studies [6, 7]. Moreover, unlike previous studies [4, 8], data confidentiality and data consistency were realized simultaneously.

The scientific contribution of this model is necessary to reduce the time and calculation cost of accessing the data. The purpose of this study was to develop a method that allows different users to access the data in a distributed environment. Our study is aim, with the path compression method, the data are accessed faster and the cost of access is reduced.

To be able to perform reading and writing and ensure sending data and function, worker nodes request a copy of the objects from the dissemination node.

A distributed environment includes one or more nodes from each node [4]. In this way, it performs data shipping (function/calculation) from node to node. With different nodes, the data and function shipping is achieved more rapidly. In addition, in multiple nodes, reading and writing operations are performed on objects [9].

Definition: Data shipping is the sending of the data from the location (node) of it to the location (node) where the calculation was made. The data is passed for also the calculation of the values of the copied objects. All the processes are performed on the client [10].

Definition: Function shipping is the sending of the calculations from their node to the node where the data presents. All processes and queries are performed on the server [3].

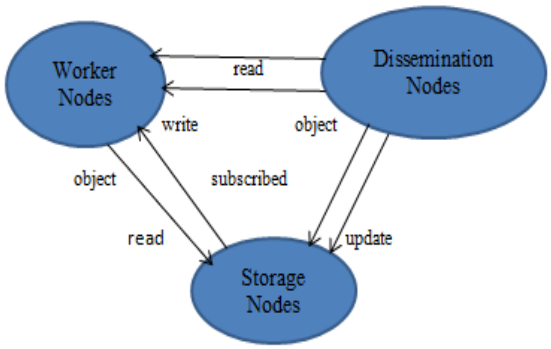

Figure 1. Providing the distributed environment with three types of nodes

Figure 1 shows the updating of object groups in the distributed environment. To ensure the object update, the worker and dissemination nodes take the object from the storage node. In this way, reading and writing processes, data shipping and function shipping can be performed on the object. When any object is updated, the storage node also updates that object [11]. The reading and writing

processes on the updated object are performed by the worker node. The dissemination node controls the information flow according to security and privacy policies. If there is authorization, the worker node takes the object from the dissemination node and performs the required actions [12].

3. Method

The data are expressed as objects, and various names are given to these objects. An object is represented by oid, which indicates the identity of this object. The oid consists of the host that indicates where the object is stored, and parts showing which object is on the host (that is, the object number).

3.1. Path Compression

The *path* is a graph structure that shows the transmission of the data objects by the actors (nodes). The nodes are connected to each other in the form of chains. As an object moves between the actors, the object identifier (oid) that shows the identity of the object is updated to include the next address of the object. For example, in Figure 2, an object (oid_1) is transferred to the storage nodes A_1, A_2, A_3, A_4, and A_5, respectively. When the object is transferred, only the address information remains on the previous node and the object is transported to the new node. When the object is moved from the A_4 storage node, it is transferred to node A_5 and the new address is saved as $oid_1(4)$ to node A_4. In other nodes where the object is passed, the address (reference) of the object remains ($oid_1(1)$–$oid_1(2)$–$oid_1(3)$–$oid_1(4)$). The object shown with $oid_1(5)$ is actually located in node A_5. $Oid_1(5)$ in A_5 shows the new oid value of the object. Because object movements create long chains between nodes in this format, the cost of accessing the object increases. In order to prevent long chains, the *path compression method* is used, which gives the result shown in Figure 3. *Path compression* is the process of updating the reference in each node on the path, which starts from the root node to the node where the object is currently located, with the current location address (See Algorithm 1).

3.2. Object Access Process Example

If we desire to locate an object without using path compression, it needs to look at all the nodes that the object passes until the desired node is reached. In Figure 2, when the position of an object in Node A_5 is asked to Node A_1, nodes A_1, A_2, A_3, A_4, and A_5, where the object address is stored, must be accessed, respectively. However, with the path compression method shown in Figure 3, it goes directly to node A_5 (A_1 gives the object address in A_5 directly). Thus, fast access to the object is possible, and the cost of access is reduced.The path compression algorithm (Algorithm 1: Algorithm Pathcompress) is an efficient algorithm with linear runtime.

Algorithm 1. Path Compression Algorithm

```
Algorithm YolKısalt (Start:Düğüm)
// Start node
1: X ←Start;
2: Y ←Start;
3: if (X=null or  next[X]==null) return;
4: // determine previous node to probe (Y)
5: while (next[X]!=nll) do
6:          Y←X;
7:          X←next[X];
8: end while
9: // update display
10: Z←Start;
11: while (Z!=X) do
12:       next[Z]←next[Y];
13:       Z←next[Z];
14: end while
15: return;
```

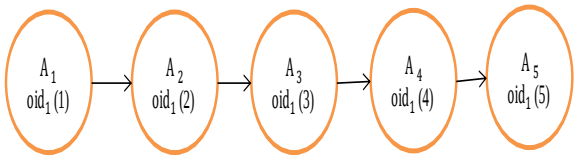

Figure 2. Node chain formation

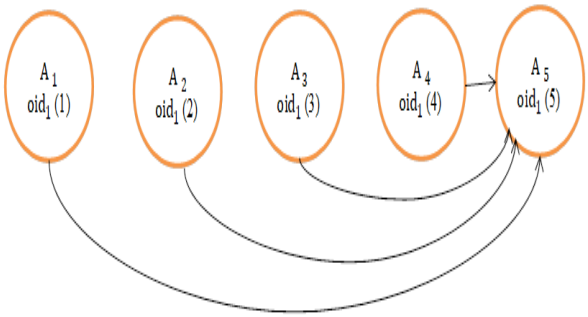

Figure 3. The new condition that occurs as a result of the path compression

3.3. Distributed Environment Simulation

In a distributed environment, some operations related to objects are performed. For example, objects can be accessed or moved. In our study, events related to objects (object-access and object-move) were created as

independent events, and a distributed environment simulation was performed. The purpose of this study was to show the effectiveness and the benefits of the path compression algorithm. In the distributed computing environment, let us assume that there are n nodes and k objects. For example, let these n nodes be denoted as d[1], d[2], …, d[n].

Each node has a local object table. Let k objects be denoted as o_1, o_2, …, o_k. The object table on each node will contain the object information showing that it resides on this node or contains the address of the object if it is in another node. For each node, there is a notation similar to that shown in Figure 4. Initially, objects are randomly assigned to nodes. This is accomplished with the function F: oi → d [j], for $1 \le i \le k$ (random (1, n)). Each o_i object is assigned to any d[j] node. We used five functions related to the objects:

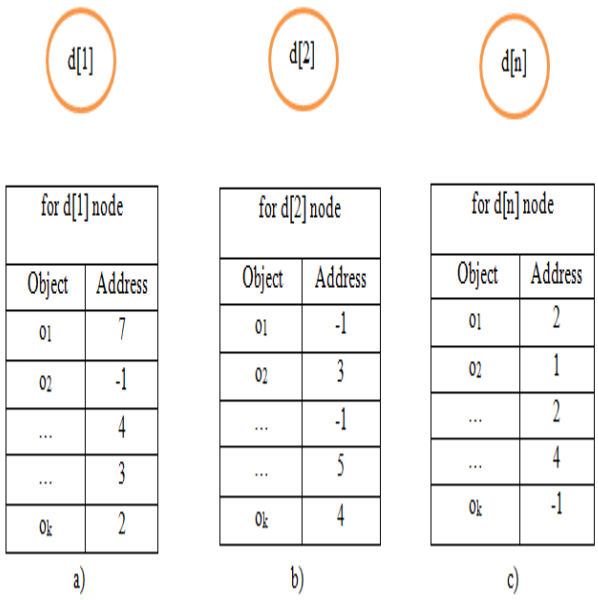

Figure 4. a) d[1] local object table, b) d[2] local object table, c) d[n] local object table)

Function 1: object_access (i:object, j:node): This function looks at the local object table of node **j** for object **i** to be accessed. If the address shown by this node is −1, it means that the object is located on this node. If not, the node address is retrieved from the **j** node object table and is assigned to **j**. This process continues until the object is found. This function returns the chain length.

Function 2: object_move (i:object, j:source node, x:destinion node): With this function, object **i** is moved from the **j** node, in which it is currently located, to the **x** node. The node where all objects reside is stored in the general object table.

The global object table stored in distributed systems is shown in Table 1. This global object table shows on which node each existing object is. It is always up to date. Furthermore, the object_access(i, j) and object_move(i, j, x) functions operate independently of each other.

Table 1. General Object Table

All objects	
Object	**Address**
o_1	
o_2	
…	
…	
…	
o_k	

Function 3: break_chain (i:object, j:source node, x:destinion node): The length of the chain of the **i** object accessed by the object_access(i, j) function is checked. The length of the chain is equal to the number of nodes that are used to access object **i**. A threshold value **T** is determined. If the length of the chain is equal to or greater than the threshold value, the **i** object breaks the chain starting from the **j** node to the **x** node where the object resides. If the chain length is less than the threshold value, this function is not executed. The appropriate values are determined by changing the threshold value.

Function 4: Book_keeping (i:object, L:chain length): The length of chain **L** is calculated for each **i** object to be accessed. The keeping of the records by this function is shown in Table 2. In this simulation, by running the DriverForObjAccess( ) and DriverForObjMove( ) functions at certain rates, we calculated the average and the maximum chain lengths. By changing the threshold value, we re-calculated the maximum and the average chain lengths.

Table 2. L Chain for Each Object Accessed with Z=Book_Keeping (I:Object, L:Chain_Length)

Object (i)	3	5	2	8	7	1	….	Zmax
Node Length (L)								

Function 5: compute_ statistics ( ): It records the length of the chain of the object, which is as much as **Zmax**, being accessed. When objects with a length equal to **Zmax** are accessed, the mean chain length is calculated by assuming that sufficient statistics are collected. The mean length of the chain is determined by taking the ratio of the total length of all the objects accessed to the number of objects accessed. The mean chain length is calculated as follows:

$$Mean\ chain\ length = \sum_{i=1}^{Zmax} L[i]\ /Zmax \qquad (8)$$

4. Experimental Study

In the distributed environment, some operations related to objects are performed. For example, objects can be accessed or moved. In our study, events related to objects (object-access and object-move) were created as independent events, and a distributed environment simulation was performed. The purpose of this study was to show the effectiveness and the benefits of the path compression algorithm. In our study, access to the objects was shown with "Access," and the moving of the objects was shown with the "Move" operations.

In the distributed environment, system modeling was performed in two ways: static and dynamic. In the static environment, while access to the objects was performed frequently, the moving of the objects was performed less. In contrast, in the dynamic environment, while access to the objects was performed less, the moving of the objects was performed more frequently. In our study, modeling was carried out for some processes related to the objects in the static and the dynamic environments.

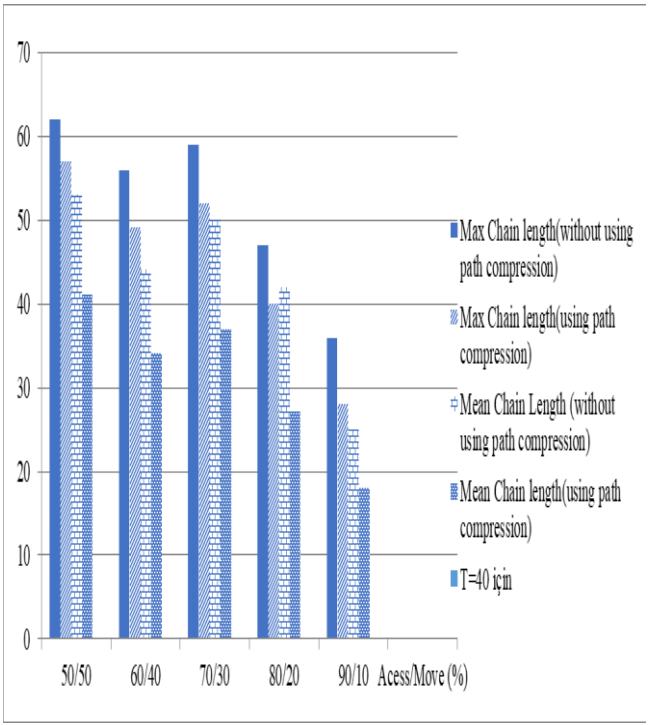

Figure 5. Maximum and mean chain length for T=40 using binomial distribution

In Figures 5, the results of the distributed environment simulation according to the various Access/Move (%) rates, respectively, for T = 40, are shown using the binomial distribution. In these graphics, the graph comparatively shows the maximum chain length and the average chain length for the cases in which we used and did not use the path compression algorithm. In this graph, **T** is the threshold value, and if the chain length is equal to or greater than the threshold value, the chain length is broken. Each column in the graph shows the maximum and the average chain lengths in the cases in which we used and did not use the path compression algorithm, respectively. In the graph, the maximum and average chain lengths are calculated by increasing the amount of access to the objects via the Access/Move (%) rates, such as 50/50, 60/40, 70/30, 80/20, and 90/10, and by decreasing the number of moving objects. When we increased the amount of access to the objects and used the path compression algorithm, the average chain length was broken more.

5. Conclusion

Flow of data objects in a distributed environment has been modeled via graph. While accessing the objects, a long chain may be formed between nodes. With path compression method this chain is broken; cost of accessing objects is lowered; performance gain is attained. Simulation of the distributed environment was carried out through an experimental study, and in addition to the evaluation of the algorithm, its effectiveness was also demonstrated by the experimental study.

Finally, an important point to consider in the path compression algorithm is that the object access frequency is high and that the path compression process is rarely performed. To guarantee this, the path compression algorithm must be run only when the length of the chain exceeds a certain threshold value. This threshold value is a parameter that must be determined as a result of careful evaluations or experimental studies. If the threshold value is selected as big, object access will slow down because of the long chain formation. On the other hand, if it is selected too small, the cost of accessing the object will increase because the path compression algorithm runs frequently.

Author's Note

Abstract version of this paper was presented at 9th International Conference on Advanced Technologies (ICAT'20), 10-12 August 2020, Istanbul, Turkey with the title of "Object Based Distributed Environment Modelling and Simulation".

References

[1] W.Cheng, D.R.Ports at all, "Abstractions for Usable Information Flow Control in Aeolus", *2012 Usenix Annual Technical Conference*, 2012, pp.1-13. DOI: 10.5555/2342821.2342833

[2] H.Esfandiari, M.Hajigohayi at all, "Streaming Algorithms for Estimating the Matching Size in Planar Graphs and Beyond", *ACM Transactions on Algorithms*, 2018, vol 14, no.8. DOI: 10.1145/3230819

[3] J. Liu and M. D. George, " Fabric: A Platform for Secure Distributed Computation and Storage", ACM Symposium on Operating Systems Principles and Implementation (SOSP),

2009, pp.321-334. DOI: 10.1145/1629575.1629606

[4] J.Liu, O.Arden at all, " Fabric:Building Open Distributed Systems Securely by Construction", *Journal of Computer Security*, vol 25, no.4-5, 2017, pp. 367-426. DOI: 10.3233/JCS-15805

[5] M.Alizadeh, S.Abolfazli at all, "Authentication in Mobile cloud computing:A survey", *Journal of Network and Computer Applications*, vol 61, 2016, pp.59-80. DOI: 10.1016/j.jnca.2015.10.005.

[6] R.Barejee, S.Chatterjee at all, "Performance of a Discrete Wavelet Transform Based Path Merging Compression Technique for Wireless Multimedia Sensor Networks", *Wireless Personal Communications*, vol.4, 2019, pp.57-71. DOI: 10.1007/s11277-018-6008-7

[7] J.Janet, S.Balahrishnan at all, "Optimizing Data Movement within Cloud Environment using Efficient Compression Techniques", *International Conference on Information Communication and Embedded Systems*, 2016.

[8] J.Cui, L.Shao at all, "Data aggregation with end-to-end Confidentiliaty and Integrity for large-scale Wireless Sensor Networks", *Peer to Peer Networking and Applications*, vol.25, no.5, 2018, pp. 1022-1037. DOI: 10.1007/s12083-017-0581-5

[9] Y. Vural and Ş.Sağıroğlu, " A review on Enterprise Information Security and Standards", *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol 23, no.2, pp. 507-522, 2008.

[10] O.Arden and A.C.Myers, "A calculus for flow-limited authorization", *IEEE Computer Security Foundations*, 2016, pp. 135-149. DOI: 10.1109/CSF.2016.17

[11] Ç.Bakir and V.Hakkoymaz, "Dağıtık Veritabanında Veri Etiketleme ile Bilgi Akış Denetimi", *5.Ulusal Yüksek Başarımlı Hesaplama Konferansı*, Esenler İstanbul, 2017.

[12] A. C.Myers and B.Liskov, "Protecting Privacy using the Decentralized Label Model", *ACM Transactions on Software Engineering and Methodology*, vol 9, no.4, pp. 410-442, 2000.