

Metrized Small World Based Data Structure

ABSTRACT

We introduce the search oriented data structure to build very large, scalable, poorly structured and unstructured distributed data storage. The main idea is to link all data units hereby every data unit has to have the local only link list to provide the associative search mechanism by roaming in compliance the links to find all data units which are relevant to the input search mask. We propose to build the link lists dynamically during the data storage growth according our Metrized Small World (MSW) properties algorithm. The known Small World stochastic graphs space equipped by semi-metric model is used to create the links for every new data unit. The XML data unit semi-metrics are developed and investigated to use this approach to create the distributed index. The ordering relationship metric generated is developed and described. As an example we describe the developed storage prototype that uses the MSW algorithm to provide the storage evolution and data search. The prototype size is 100000 units and estimated average path length during the search provided to be about 6-7 links. It means that any mask will generate in average around 12 hops search to find the relevant data unit. Testing of the search features has shown that the theoretical assumptions are correct and we can use the MSW approach to build the petabyte and exabyte storages.

1. INTRODUCTION

Modern applications set the requirement to store and process large amounts of data generated by constantly growing number of uncoordinated sources which have no common control center. The examples of such data is global bank of media item descriptions or Electronic Product Code (EPC) data . Data consumers (query sources) are also numerous and uncoordinated. To fulfill this requirement new data structures are necessary which possess the following properties:

- Decentralization, because the sources of data and queries are uncoordinated and distributed around the globe.
- Search of the relevant information in relatively short time compared the large volume of data contained in the structure.
- Scalability, since the number of data and query sources grows constantly.
- The possibility to deal with weakly structured data, because there is no practical way to impose restric-

tions on the structure of data elements generated by uncoordinated sources.

In the Background section we give an overview of the existing data structures, their advantages and drawbacks. Further in the Metrized Small World section we describe ways to construct the new data structure (MSW) which allows for decentralized storage of unbounded number of data items and ensures reasonable search times. In the Simulation section we describe our prototype implementation of the structure which uses XML documents as data items and XLink to express links between them. Then we expose the experimental characteristics obtained using that implementation.

2. BACKGROUND

There is a multitude of well known state-of-the-art data structures and ways to represent information. Each type of data structure has its own set of properties which determine its efficiency for any given task.

2.1 Arrays

Arrays allow for retrieving an element by its index with the complexity of $O(1)$ by calculating its memory address using the starting address of an array and the index. The complexity of searching an element by its value is $O(n)$ in an unsorted array and $O(\log(n))$ in a sorted array. But in the latter case the operation of appending an element is expensive because array elements must be shifted in memory.

2.2 Linked lists

In linked lists the elements are accessed by traversing elements following the links contained in each element. Linked lists are differentiated by the way the links are established. One simple variant is unidirectional list where each element stores a link to the next element. Pointer to the head element is required to access elements of a unidirectional list. In a bidirectional list each element has links to the next and the previous element. Therefore pointer to an arbitrary element of the list is sufficient to access all other elements [5].

2.3 Binary trees

Binary trees are a special case of linked lists. The way the links are established in a binary tree allows for appending, removing and searching of elements (including minimal and maximal) with the complexity of $O(\log(n))$. The process of searching and appending of elements always starts from the root element [3].

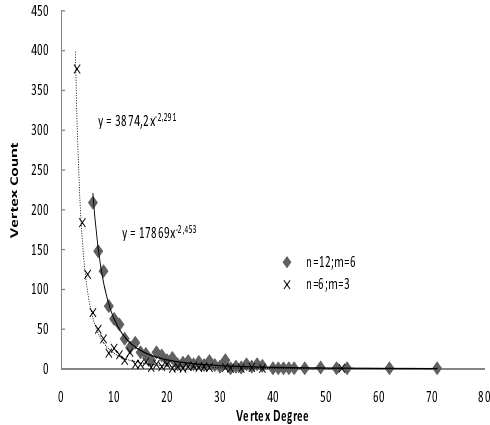


Figure 1: Vertexes degree distribution and shortest path length distribution. Number of vertexes in structure is equal 1000

2.4 Hash tables

Hash tables are associative arrays which map keys to the stored data elements (values). In a common variant of hash table the hash function value of a key is used as address of the list which contains all elements whose keys have the same hash value. The complexity of search in such a hash table is $O(1)$ in the best case and $O(n)$ in the worst case.

2.5 Distributed hash table

similar to a conventional hash table. The difference is that in DHT the responsibility for storing and searching of data is distributed between a number of network nodes. Each node is responsible for a certain subset of key-value mappings and has links to other nodes. The process of mapping of a key to its value is performed by first finding the node responsible for a given key by following the links and then retrieving the value from that node. There are different kinds of DHTs distinguished by the type of hash function, topology and the algorithm of finding of the node responsible for a given key hash value. Examples of DHT are CAN (Content Addressable Network), Chord [2], Kademlia, Pastry, P-Grid, Tapestry. One drawback of DHTs, as well as of conventional hash tables, is that they cannot search elements by value.

3. METRIZED SMALL WORLD

In the proposed data structure elements are connected by links in a way similar to a linked list where elements are traversed by following the links contained in each element. The difference from similar existing structures is the way in which links are established. To ensure that a data structure has the properties listed in the Introduction, the following requirements must be met:

1. To allow for decentralization any element of the structure must be able to serve as an entry point of access to other elements
2. To ensure search in a small number of operations com-

pared to the number of elements, a relatively short link path must exist between two arbitrary elements of the structure

3. An evident restriction is that any element must store a relatively small number of links

Tree-like structures such as binary or B+ trees satisfy the requirements 2 and 3 but fail to meet the first requirement of decentralization. In tree-like structures search must always start from the root element which prevents the construction of a decentralized structure. Star-like structures can satisfy the first and the second requirements but they necessarily have the central element which stores the links to all other elements, thus failing the requirement 3. Nowadays the structures are known which satisfy all three requirements above. These are the Small World graphs [1]. It was noted that the Small World graphs have the following properties:

- The vertex degrees are distributed according to the power law.
- The average geodesic path length between two vertices is proportional to a logarithm of the number of vertices.
- The clustering coefficient does not depend on the number of vertices.

There are existing methods of construction of such graphs described in [6] but those methods are not suitable for the peer-to-peer systems because they require iterating over the entire set of vertices to add a new vertex. A way to reorganize the data in a peer-to-peer small world system using only local data of each vertex is discussed in [4]. We propose the method of dynamic construction of such structures from the given set of elements. Let V be the set of elements numbered in the order of their appearance. The elements must be connected in such a way that the resulting graph is a Small World graph. The idea is to choose m elements

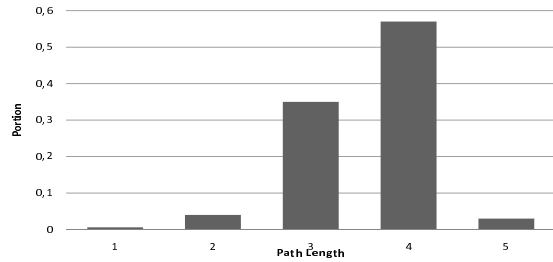
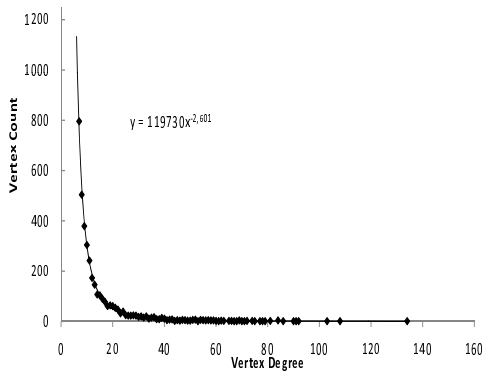


Figure 2: Vertexes degree distribution and shortest path length distribution. Number of vertexes in structure is equal 5000

from the structure with the probability proportional to their vertex degrees. If we follow the links from one element to another in a random manner, the elements with higher vertex degree will be visited with higher probability. We suggest the following algorithm with n and m parameters where n is the number of algorithm steps (determines the join accuracy) and m is the number of links established by the newly added element, $n \geq m$. Let's assume that the structure already contains $i - 1$ elements and the i -th element is being currently added. Then the algorithm is as follows:

Algorithm 1 Add V_i element

Require: n, m, V_i, V_k where $1 \leq k < i$

Let *visitedList* be the set of visited elements

Let *candidateList* be the set of candidates for link establishment

Assume that *candidateList* initially contains only V_k

for $i = 1$ to n **do**

 Arbitrarily choose an element p from *candidateList* not contained in *visitedList*.

if no such element exists **then**

 break

end if

 Remove p from *candidateList*

 Add p to *visitedList*

 Append all neighbors of p not contained in *visitedList* to *candidateList*

end for

Mutually connect V_i to m arbitrary elements from *visitedList*

If a pseudometric $M(V_i, V_j) \rightarrow R$ is defined on a set of elements V it is possible to form the links in such a way that the elements that are close according to the pseudometric are separated by only a small number of links. Such pseudometrics for the sets of XML documents are discussed for example in [7] and [6]. We modified the above algorithm so

that when a new element is appended to the structure the links with m closest elements are established. The modified algorithm is as follows:

Algorithm 2 Add V_i element

Require: n, m, V_i, V_k where $1 \leq k < i$,

Let *visitedList* be the set of visited elements

Let *candidateList* be the set of candidates for link establishment sorted by value of pseudometric to V_i in ascending order

Assume that *candidateList* initially contains only V_k

for $i = 1$ to n **do**

 Sort *candidateList* by value of pseudometric to V_i in ascending order

 Select the first element p from *candidateList* not contained in *visitedList*

if no such element exists **then**

 break

end if

 Remove p from *candidateList*

 Add p to *visitedList*

 Append all neighbors of p not contained in *visitedList* to *candidateList*

end for

Mutually connect V_i to m closest elements from *visitedList*

The use of pseudometric in the construction and search algorithms allows for successful search when the structure of data is different from the query structure. This can be achieved by constructing a pseudometric which will yield minimal values for the data elements which correspond to the query despite the differences in the internal structure.

The formation of the structure according to the proposed method is strongly dependent on the properties of the pseudometric. It is impossible to guarantee that every time a new element is appended to the structure it is connected

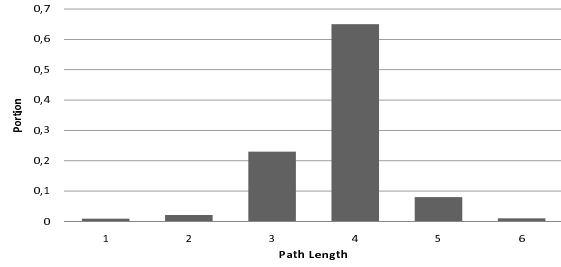
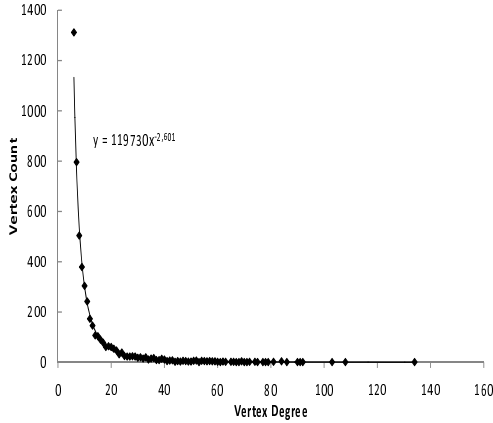


Figure 3: Vertexes degree distribution and shortest path length distribution. Number of vertexes in structure is equal 10000

to the closest element by pseudometric without inspecting all the elements in the structure (it is not possible to find the place for an element in the structure precisely). This problem is eliminated when in addition to the pseudometric we impose the constraint that the elements are ordered by ordering relationship the metric induced. In this case each element must be connected to its previous and next element. For an instance, when the elements are real numbers, the pseudometric can be defined as absolute difference of two elements and the order of elements will correspond with their natural order. In this case the construction algorithm differs from the ones above and consists of three stages. On the first stage the V_{min} element is found whose neighbors have equal or greater values of pseudometric to the element being added (V_i) than the V_{min} element itself. The V_{min} element is a local minimum of the pseudometric between V_i and the elements of the structure. V_{min} is either the previous or the next element for V_i . On the second stage V_i is connected to its previous and next elements. One of these elements is V_{min} and the other is one of the neighbors of V_{min} . On the third stage the previous algorithm is executed to find m closest elements and to establish connections with them assuming $V_k = V_{min}$ and $n = m$.

The method described above is applicable for constructing of a distributed index or for implementing a distributed hash table. A limitation of the method is that the new elements must be added in random order relative to the order prescribed by the pseudometric.

4. SIMULATION

In the implementation prototype the elements of the structure are represented by XML documents. Each document is accessible via HTTP by its unique URL and has a set of XLink links to other document which is also accessible by a unique URL trivially calculated using the URL of the document. This set of links contains all links emanating from the given document. For every link the reverse link exists

Algorithm 3 Add V_i element

Require: m, V_i, V_k where $1 \leq k < i$,

First stage

Let *visitedList* be the set of visited elements

Let *candidateList* be the set of candidates for link establishment sorted by value of pseudometric to V_i in ascending order

Assume that *candidateList* initially contains only V_k

$lastDistance \leftarrow MaxDouble$

$p \leftarrow V_k$

while $M(V_i, p) < lastDistance$ **do**

$lastDistance \leftarrow M(V_i, p)$

$V_{min} \leftarrow p$

 Add p to *visitedList*

 Append all neighbors of p not contained in *visitedList* to *candidateList*

 Sort *candidateList* by value of pseudometric to V_i in ascending order

 Select the first element p from *candidateList* not contained in *visitedList*

if no such element exists **then**

 break

end if

 Remove p from *candidateList*

end while

Second stage

if $V_{min} \leq V_i$ **then**

 Choose among the neighbors of V_{min} the element V_r which follows V_{min}

else

 Choose among the neighbors of V_{min} the element V_r which precedes V_{min}

 Mutually connect V_i with V_r and V_{min}

end if

Third stage

Execute **Algorithm 1** (m, m, V_i, V_{min})

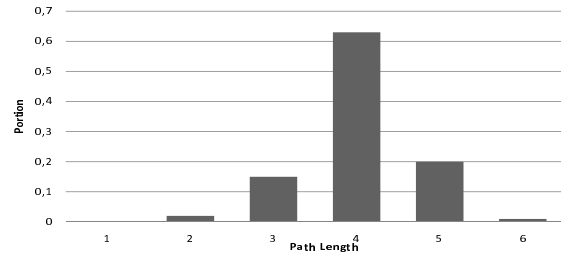
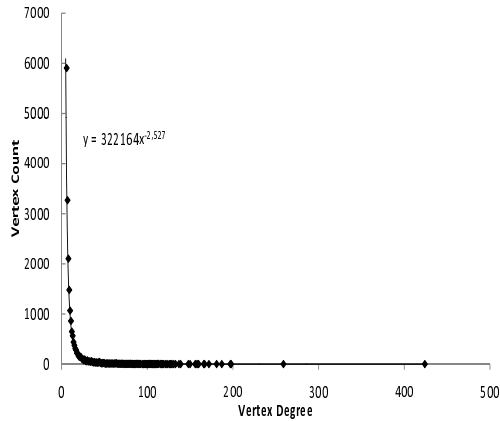


Figure 4: Vertexes degree distribution and shortest path length distribution. Number of vertexes in structure is equal 20000

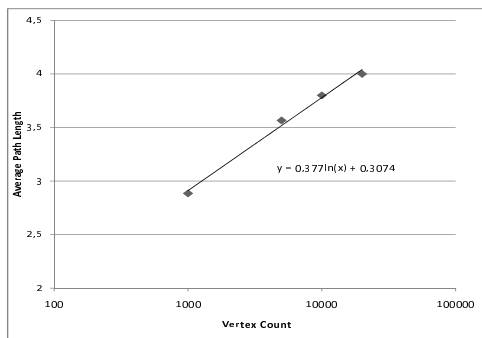


Figure 5: Shortest path length distribution averaged over all structures

in the list of links for the document which is the target of the link. When a new document is added to the structure a link to this document is dynamically added to the list of links of every existing document to which the new document must be linked. We provide the results obtained from execution of the add-metric algorithm using XML media item descriptions as test data. Structures containing 1000, 5000, 10000 and 20000 elements were assembled using this algorithm. The graphs of vertex degree and shortest path length distributions are shown in figures 1-3. In Figure 3 you can see how the average shortest path length between vertices changes with increasing number of vertices in the structure.

5. CONCLUSIONS

We proposed the three requirements which the new data structure must satisfy. In our implementation prototype the elements are accessed by sending a HTTP request to a particular URL which imposes no constraints on the physical location of the elements. Any element with a known URL can serve as an entry point for access to other elements, thus satisfying the first requirement. The path length distribution graphs demonstrate that a short path exists between any two nodes and it shows little growth when the number of elements is increased. Thereby the structure satisfies the second requirement. The vertex degree distribution graphs show that vertex degrees are distributed according to the power law, whereby the number of vertices having high degrees is far less than the number of low degree vertices. Hence the structure satisfies the third requirement.

We are developing the new effective metrics and search algorithms with minimal deviations from the shortest path. When the search criterion is a complete match with the mask the algorithms we described can become ineffective. So we explore new ways of constructing the structure, in particular based on the partial ordering of the data elements.

6. REFERENCES

- [1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97, January 2002.
- [2] D. K. M. F. K. Ion Stoica, Robert Morris and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, October 2001.
- [3] D. E. Knuth. *The art of computer programming, volume 1: fundamental algorithms*. John Wiley and Sons, Addison-Wesley, 1968.
- [4] C. Schmitz. Self-organization of a small world by topic. In *Proc. of the 1st Intl. Workshop on Peer-to-Peer*

- Knowledge Management*, pages 61–66, PUBLISHER=.
- [5] C. E. L. Thomas H. Cormen and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
 - [6] F. C. W. Aiello and L. Lu. Random evolution in massive graphs. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 510. FOCS, IEEE Computer Society, 2001.
 - [7] C. S. Zhang Z., Li R. and Z. Y. Similarity metric in xml documents. In *Knowledge Management and Experience Management Workshop*, pages 84–89, 2003.