An overlay network for distributed exact and range search in one-dimensional space

Alexander A. Ponomarenko
Researcher, Laboratory of Algorithms and Technologies for Network Analysis
National Research University Higher School of Economics
Address: 136, Rodionova Street, Nizhny Novgorod, 603093, Russian Federation
E-mail: aponomarenko@hse.ru

Yury A. Malkov
Junior Research Fellow
The Institute of Applied Physics of the Russian Academy of Sciences
Address: 46, Ulyanova Street, Nizhny Novgorod, 603950, Russian Federation
E-mail: yurymalkov@mail.ru

Andrey A. Logvinov
Project Leader, MERA Labs LLC
Address: 192, Rodionova Street, Nizhny Novgorod, 603093, Russian Federation
E-mail: alogvinov@meralabs.com

Vladimir V. Krylov
Professor, Head of Big Data Laboratory
Nizhny Novgorod State Technical University
Address: 24 Minina Street, Nizhny Novgorod, 603950, Russian Federation
E-mail: vkrylov@heterarchica.com

Abstract

The ability to scale is a desirable business requirement for computer systems. Distributed systems clearly demonstrate this ability and might to process very large volumes of data. Many systems with distributed architecture are based on the distributed hash table (DHT), which manages a set of distributed network nodes connected not only by a physical channel, but also by an additional overlay network. This overlay network is used for searching nodes and for distributing tasks among them. The main feature of this approach is that there is no central element or node which knows the global topology of the network. Nodes in the network are searched by passing a query message from one node to another. Despite that, every node has knowledge only about a small number of other nodes, and the network is organized in such a way that search involves a logarithmical number of nodes.

There are several DHT implementations which specify how to construct and how to support the structure of the network. In this paper, we demonstrate the way in which such a network can be constructed much simpler by applying the sight modification of the recently published Metrized Small World algorithm to the case of one dimension. We provide a theoretical analysis for the case of uniform distribution and empirical analysis for other distributions. The main advantage of the proposed algorithm is that it is immutable to data distribution and does not need to support any particular distribution of the length.

In addition, we show how to separate completely the concept of network location of data from the search functionality. This separation is important, for instance, for building global storages where data is owned by multiple parties and each party is interested in keeping control over the aspects of physical storage and access to the data it owns. So in contrast to DHT, insertion of new data does not require relocation to an existing node.

Key words: nearest neighbor search, metric space, distributed computing, Internet technology and applications, data structure, algorithm.

Citation: Ponomarenko A.A., Malkov Yu.A., Logvinov A.A., Krylov V.V. (2016) An overlay network for distributed exact and range search in one-dimensional space. *Business Informatics*, no. 1 (35), pp. 26–36.
DOI: 10.17323/1998-0663.2016.1.26.36.

Introduction

The scalability of a computer system is its ability to handle a growing amount of work [4]. It can also be referred to as the capability of a system to increase its performance under an increasing load. An analogous meaning is implied when the word "scalability" is used in an economic context, where scalability of a company implies that the underlying business model offers the potential for economic growth within the company. The concept of scalability is desirable in technology as well as in the business setting.

Data storage is a central part of any computer system. The scalability of any computer system strongly depends on the ability to scale its data storage. Traditionally, in most cases developers use relational database management systems (RDBMS) as a data storage. Unfortunately, scalability of RDBMS has inherent limitations [7]. The reason is that to support transactions and consistency, RDBMS need a central coordination point such as a transaction manager, which becomes a bottleneck with a growing number of servers on which RDBMS are deployed. Moreover, most RDBMS have an architecture in which servers have to share data with each other and therefore have to be synchronized, and this is a difficult task for a large number of servers.

To overcome the scalability problems, it was decided to sacrifice some part of functionality for a better performance [5]. Thus NoSQL databases appeared. The most important class of "NoSQL" databases is the key-value stores. Such systems mainly support two operations: retrieving and storing data by a given key. Many popular internet services are based on key-value storages. For example, the messaging system of Facebook and store of dashboard messages of SoundCloud both use Apache Cassandra; many products of Google such as Gmail, YouTube, Google Maps are based on BigTable, which is also a key-value store.

One of the possible ways to implement scalable key-value storage is to use a distributed hash-table (DHT). The distributed hash table is a class of systems which allows you to store and to search data by key in a set of network nodes [17]. A particular implementation of DHT is a protocol which specifies how nodes communicate and how they form an overlay network. It is important that every node doesn't know a full topology of the network. Instead, every node stores only a small amount of information about the network, which in turn is stored in a routing table. Every node has its own key – ID, which typically is a hash value, calculated from its IP address. The data with some key *k* is placed on the node which ID is closer to k than ids of other nodes in terms of some distance function *d*. The search is performed using a greedy algorithm by passing query message from one node to another, based on the list of nodes stored at every node in the routing table. Different DHT implementations use different distance functions. For instance, it can be a simple difference between two numbers x and y $d(x, y) = (x - y) \mathrm{mod}\ n$ [16] or it can be an XOR-metric defined as $d(x, y) = x \oplus y)$ [14].

The main advantage of all DHT implementations is that the expected number of nodes that a query message should pass before it reaches the destination node is $log(n)$. Here n is the total number of nodes in the network. Moreover the maximum size of a routing table at any node is also proportional to $log(n)$. These two properties together make DHT scalable. So the insertion of new nodes adds a very small overhead to the whole performance of the system.

The secret of these two properties lies in the structure of the routing table that is supported at every node. The main idea of all implementations of DHT is to maintain the routing tables in such a way that the i-th record in the routing table (link) of a particular node with id A points to the node with id B such that the distance from A to B belongs to the interval $2^i < d(A,B) \leq 2^{i+1}$. This gives what in turns corresponds to the power law probability distribution of links length $P \sim d^{-1}$. When a new node is being inserted, DHT explicitly forms the routing table of the new node and updates the routing tables of the existing nodes according to this distribution. Together, all nodes with their routing tables form an overlay network which can be represented by graph $G(V, E)$, where the set of vertices V is a set of network nodes and E is a set of edges corresponding to the records in the routing tables. So any particular implementation specifies how to build graph G and how to support it.

As previously mentioned, all DHT implementations form i-th record in the routing table for node A explicitly by finding node B such that $2^i < d(A,B) \leq 2^{i+1}$ in order to build a graph where the greedy walk algorithm can find a closest node to a given key with $log(n)$ steps.

In this paper we describe an algorithm which constructs a graph with similar navigation properties more naturally and more simply without an explicit support of a particular distribution of the links length. We apply our approximate algorithm suitable for a general metric space [12, 13] with a slight modification of the insertion algorithm to the exact search in the one-dimensional case. Unfortunately, the theoretical analysis of the search algorithm for a general case of Metrized Small

World is still absent. In this paper in section 4 we provide such analysis for the one-dimensional case. In the section 5, we give an empirical study of degree distribution, independence from data distribution, stability analysis and dependency of the search algorithm complexity on its parameters.

Note that previously the algorithm for the case of one dimension was mentioned and used as a basic algorithm for building a multi-attribute data structure in the paper [11]. This paper wrongly refers to the paper with the title "Single-attribute distributed Metrized Small World data structure" where this algorithm should be described in detail; however this paper by mistake was not published. In the presented paper we want to fill this gap. For consistency purposes. We provide this algorithm in section 2. We describe the insertion and search algorithms in detail with the complex empirical study and theoretical analysis.

We also suggest how to completely separate the concept of network location of the data from the search functionality. This separation is important, for instance, for building global distributed storages where data is owned by multiple parties and each party is interested in keeping control over the aspects of physical storage and access to the data it owns. So in contrast to DHT, insertion of new data does not require relocation of the existing node. In this way, the proposed overlay network with the search and insertion algorithm can be considered as a data structure built onto a data set.

The rest of the paper is structured as follows. Section 1 describes the Metrized Small World overlay data structure and its mapping to the network node collection. Section 2 specifies the data insertion algorithm. Section 3 describes the data search algorithms. In section 4 we prove that the search algorithm has $O(\log n)$ complexity. Section 5 provides the experimental data and gives an analysis of the properties of the proposed structure. Finally, we summarize our contributions in the Conclusion.

1. One-dimensional Metrized Small World network

The distributed Metrized Small World (MSW) structure consists of a number of fully interconnected network nodes (so that each network node can communicate directly with another node). Each node contains a number of data units. Both nodes and data units have identifiers unique in the context of the whole structure. Node identifiers are resolved to network addresses (e.g. DN to IP) by an external system (such as DNS). A data unit identifier contains the identifier of a node on which the unit is allocated, thereby allowing instant identification of a network node containing the data unit. This makes it possible to create an overlay structure on data unit level, which is inherently mapped to the physical network nodes. Each data unit A stores a mutable set $N(A)$ of identifiers of other data units, which are the links that constitute the overlay network. The overlay network can be considered as a graph $G(V, E)$, where the set V corresponds to the set of all data units and the set E contains an edge (A, B) if data unit A has a link to data unit B.

All links are bidirectional, i.e. each data unit is linked with data units linked with it. Nodes are not directly aware of each other; identifiers of other nodes can only be obtained from the links to data units allocated on those nodes.

An illustration of the structure is given on *Figure 1*. The structure is allocated on three network nodes identified by unique URL addresses. Data units are allocated on these nodes and also have unique URL addresses which are sub-URLs of the nodes. The overlay structure is built on the data units; the nodes do not have direct links to each other. The algorithms of data unit insertion and search operate solely on the overlay structure formed by data units, and links between them are completely agnostic to the physical node collection supporting the structure (data unit identifiers are treated as atomic by the algorithms). The choice of a node to allocate a new data unit or the decision to add a new node are completely independent of the data unit insertion algorithm which is executed after the new data unit is allocated. The introduction of a new node does not affect the existing overlay structure or the distribution of data units between nodes; it merely provides new allocation space and processing power for insertion and search algorithms.

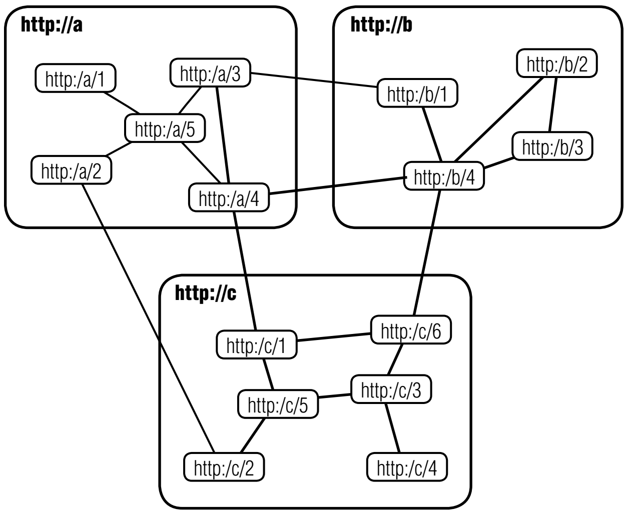

Fig. 1. An overlay network structure constructed on a set of data units

The overlay structure is accessed in a locally iterative manner, starting from an arbitrary known data unit (entry point) and following the links contained in the visited data units. To find a relevant data unit, one must choose an entry point and navigate through the structure, choosing a link at each step.

To make this process efficient, two requirements must be met:

1. There should be a relatively short path between the entry point and the target data unit. Since neither the entry point nor the target data unit are known beforehand, a short path must exist between any pair of data units in the structure;
2. A criterion must exist which would allow us to decide which link at each step will bring the search process closer to the result. It is also desirable that the gradient of the proximity criterion corresponds to the shortest path to the result in the structure, avoiding false local minimums when possible.

To address the first requirement, we have developed a data unit insertion algorithm that incrementally produces a structure with small-world properties. The small-world graphs [1, 2, 19] have the essential property that they have a short path between any two vertices while most vertices are not neighbors of each other.

To address the second requirement, a data unit contains searchable information and non-searchable information. As a model of searchable information in this paper, we consider the set of objects U that can be bijectionally mapped to the unit interval of real numbers. For example, U can be a set of all possible strings $S = (s_1,...,s_n)$, where s_i is the index of a symbol in the alphabet B. If we define a mapping,

$$C(S) = \sum_{i=1}^{n} \frac{s_i}{|B|^i}.$$

For the purposes of the description of algorithms, we will denote as $C(D_i)$ a mapped value of the searchable information of the data unit D_i. We used a simple metric M between data units w hich is calculated as $M(D_i, D_j) = |C(D_i) - C(D_j)|$. This metric is used both by the insertion and search algorithms. Such a simple model of the searchable information allows us to make the search exact, since there is a natural linear order on the set of real numbers. This order is related to the metric M. Based on the linear order, the insertion algorithm has an ability to find the exact Voronoi's neighbors for every data unit. That is the main advantage of this model in contrast with an approximate search in a general metric space [10, 12, 13], where it is hard to find the Voronoi's neighbors exactly.

However, this approach is useful for the key-value storages and systems like CAS (Content Addressable Storage) [18] where a content-derived location-agnostic identifier of the data object, e.g. hash-code, is transformed into a content-agnostic location identifier, e.g. network or disk address, by the use of which the data object is retrieved. In terms of the MSW structure, the content-derived identifier would constitute the searchable information of the data unit and the content would be the non-searchable information. As a result of the search process, the content-derived identifier will be transformed into the identifier of the data unit which stores the content.

Other comparable solutions are the distributed hash table (DHT) [3] systems such as Chord [16]. These systems use a consistent hashing approach [8] to decide which network node will store the data unit. Insertion of a new network node requires relocation of K/n number of data units on average where K is the total number of data units and n is the number of nodes. In the MSW system, by contrast, insertion of a new node does not affect existing nodes because allocation of data units is completely irrelevant to the search and insertion algorithms. The new node becomes involved in the structure when a data unit allocated on the new node is added to the MSW structure using an entry point on one of the existing nodes. Another difference from the DHT systems is that MSW creates an overlay network on the data unit level while the overlay network of DHT systems like Chord is built on the node level.

A detailed description of the technological aspects of a scalable database system based on the MSW structure is presented in [9].

2. Data insertion algorithm

The purpose of the data unit insertion algorithm is to connect a newly allocated data unit to the structure by populating its list of links and correspondingly extending link lists of the data units chosen to be connected with the new data unit so that the greedy walk algorithm can find any data unit starting from any data unit. When there is an ordering relationship R related to the distance metric (for example the natural order of rational numbers), we can ensure the greedy search is guaranteed by connecting each data unit with its direct predecessor D_{pre} and direct successor D_{succ} — in contrast to an approximate search for a general metric space. In order to make the search algorithm faster, we additionally connect a new data unit with other m closest data units and do not remove these links during evolution of the structure.

The pseudo code of the algorithm is presented below.

```
Insertion_with_Order
  Input: $D_{new}$ – new data unit; $D_{ep}$ – entry point data unit; $m$ – parameter (small natural number).
1. Let $D_{cur} = D_{ep}$.
2. For each $D \in N(D_{cur})$ calculate $M_i = M(D_i, D_{new})$.
3. If $min(M_i < M(D_{cur}, D_{new})$ let $D_{cur} = D_i$ for which $M_i = min(M_i)$ and go to step 2.
4. If $C(D_{cur}) < C(D_{new})$ let $D_{pre} = D_{cur}$ and let $D_{succ}$ be the direct successor of $D_{new}$ chosen from the neighbors of $D_{cur}$.
5. If $C(D_{cur}) > C(D_{new})$ let $D_{succ} = D_{cur}$ and let $D_{pre}$ be the direct predecessor of $D_{new}$ chosen from the neighbors of $D_{cur}$.
6. Connect $D_{new}$ with $D_{pre}$ and $D_{succ}$ if they exist.
7. Repeat $m$ times:
    7.1. If $D_{pre}$ exists, let $D'_{pre}$ be the direct predecessor of $D_{pre}$ chosen from its neighbors.
    7.2. If $D_{succ}$ exists, let $D'_{succ}$ be the direct successor of $D_{succ}$ chosen from its neighbors.
    7.3. If none of $D'_{pre}$ and $D'_{succ}$ exist then break.
    7.4. If only $D'_{pre}$ exists or $M(D'_{pre}, D_{new}) < M(D'_{succ}, D_{new})$ connect $D_{new}$ and $D'_{pre}$, and let $D_{pre} = D'_{pre}$.
    7.5. If only $D'_{succ}$ exists or $M(D'_{succ}, D_{new}) < M(D'_{pre}, D_{new})$ connect $D_{new}$ and $D'_{succ}$, and let $D_{succ} = D'_{succ}$.
```

At each iteration, the algorithm obtains the neighbors of the current data unit D_{cur} closest to D_{new} and calculates the distance M_i to each of them. After that, the algorithm updates D_{cur} and makes the next iteration. So, we use a kind of greedy search algorithm to find the data unit closest to the new data unit (after all iterations the closest to D_{new} data unit will be in D_{cur}). We find out whether D_{cur} precedes or follows D_{new} and connect the new data unit with its direct predecessor D_{pre} and direct successor D_{succ}. Finally, we connect the new data unit with m closest existing data units from the right (succeeding data unit) and left (preceding data unit) side of it.

3. Search algorithm

The greedy search algorithm finds a data unit with a string closest to a given string S_q starting from the entry point data unit D_{ep}. If there is a data unit with a string identical to S_q present in the structure, the greedy algorithm will return this data unit. For brevity, we will treat each data unit as if it were the same entity as the searchable string which it contains.

```
Greedy_Search
  Input: $S_q$ – key; $D_{ep}$ – entry point data unit.
  Return: data unit which is closest to $S_q$.
1. Let $D_{cur} = D_{ep}$.
2. For each $D \in N(D_{cur})$ calculate $M_i = M(D_i, S_q)$.
3. If $min(M_i > M(D_{cur}, S_q)$, then return $D_{cur}$.
4. Let $D_{cur} = D_i$ for which $M_i = min(M_i)$ and go to step 2.
```

The greedy search algorithm described above relies on the fact that each data unit is always connected by the insertion algorithm with its direct predecessor and successor in the order defined by natural order of values $C(D_i)$. This ensures absence of false local minimums in the structure. In section 4, we will show that the complexity of this search algorithm grows logarithmically with the number of data units in the structure.

Since all data units are ordered, it is possible to define an operation that retrieves all data units from the given interval. We will denote the direct predecessor of element x as $D_{pre}(x)$ and the direct successor of x as $D_{suc}(x)$. The pseudo code of range search is presented below.

```
Range_ Search
  Input: $L_q$ – left bound; $R_q$ – right bound; $D_{ep}$ – entry point data unit.
  Return: the set of data units from interval $[L_q, R_q]$.
1. Let $D_{cur} = Greedy\ Search(\frac{L_q + R_q}{2}, D_{ep})$
2. $T = \varnothing$; $D'_{cur} = D_{cur}$
3. While $D_{cur} \geq L_q$ do $T = T \cup D_{cur}$ and let $D_{cur} = D_{pre}(D_{cur})$;
4. Let $D_{cur} = D_{suc}(D'_{cur})$.
3. While $D_{cur} \leq R_q$ do $T = T \cup D_{cur}$ and let $D_{cur} = D_{suc}(D_{cur})$;
4. Return $T$.
```

The idea of the range search algorithm is very simple. We start from the center of the interval, go left until we reach the left bound and collect all data units to the set T. After that, we do the same in the opposite direction until we reach the right bound and return the set T as a result.

4. Theoretical analysis of the search algorithm

Theorem. The algorithm Greedy_Search has $O(\log n)$ complexity in the structure formed by the construction algorithm Insertion_with_Order.

Proof. Let us assume that the images $C(D_i)$ are uniformly distributed in the segment [0, 1]. The goal is to demonstrate that each time the size of the structure doubles, the average number of steps required to reach any data unit from any other data unit increases by an amount bounded by a constant, thereby producing logarithmic search complexity growth. First, we consider the situation when we have the set of N first data units in the structure. We call this set the first generation. Let us assume that we can reach any data unit from any other data unit in at most P steps. The probability density of the metric distance L between adjacent (in terms of metric) data units falls exponentially, so if we ignore the intervals between adjacent data units R times larger than average (i.e. larger than R/N) we omit only exponentially small number of cases. We thereby ignore the rare cases of very large gaps between adjacent data units in the first generation.

Next, we add N of the second generation data units, i.e. double the number of data units. Consider the largest interval between adjacent data units in first generation. The number of second generation data units k falling into the interval conforms to the Poisson distribution

$$p(k)=\frac{\lambda^k}{k!}e^{-\lambda} \text{ for large } N.$$

Since the length of the interval is R/N we have the expected value of the number n_{II} of second generation data units falling into the interval $\lambda = R$ and hence

$$p(k)=\frac{R^k}{k!}e^{-R}.$$

If we assume that $n_{II} < M$ we will be wrong only in a small number of cases if M is large enough. Now we will determine the upper bound U on the number of steps required to reach any data unit from any other data unit after the insertion of the second generation data units. Consider the worst case when both initial and target data units belong to the second generation. We are taking into account only the links between adjacent first generation data units, adjacent second generation data units and adjacent first and second generation data units, which are a subset of all links in the structure, which further increases U. The links between adjacent first generation data units are longer in terms of metric than the links between adjacent data units in the second generation.

Hence, the former links are preferable to the latter in reaching the target data unit in the smallest number of steps. At most, M steps are required to reach the closest first generation data unit from the initial data unit, at most, P steps are required to reach the first generation data unit closest to the target data unit, and again, at most, M steps are required to reach the target data unit from the closest first generation data unit.

Therefore, at most $P + 2M$ steps are required to reach any data unit from any other data unit after the insertion of second generation data units. Thereby, we have shown that when the structure size is doubled, the number of search steps is only increased by the constant $2M$, which means that we have $O(log\, n)$ search complexity where is the number of data units in the structure.

5. Simulations

We have implemented a single-attribute MSW structure. The simulations have been performed for different numbers of data units in the structure and for different values of parameter m of the insertion algorithm.

Figure 2 shows the average path length (number of steps) which the search algorithm needs to reach the data unit closest to the query. Networks with sizes in range from 1000 to 1000000 were generated by the Insertion_with_Order algorithm on the set of real numbers which has five different random distributions: uniform on the segment [0, 1], power law with power 0.5; normal distribution with mean 0.0 and standard deviation 1.0; log-normal with mean 4.0 and standard deviation 0.5; and points distributed normally with deviation 1.0 around 10 centers {-10.0, -7.0, -5.0, 0.0, 1.0, 2.0, 10.0}.

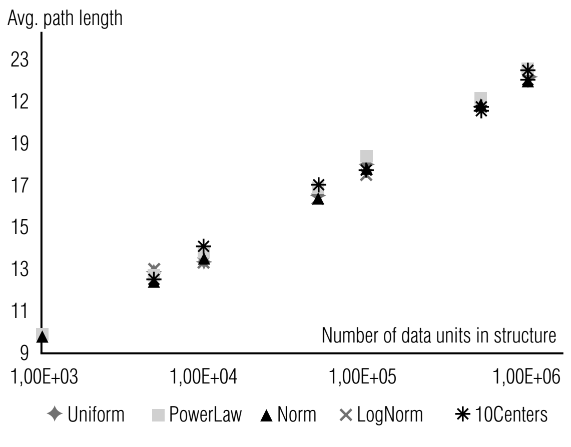

Fig. 2. Average path length for different distributions

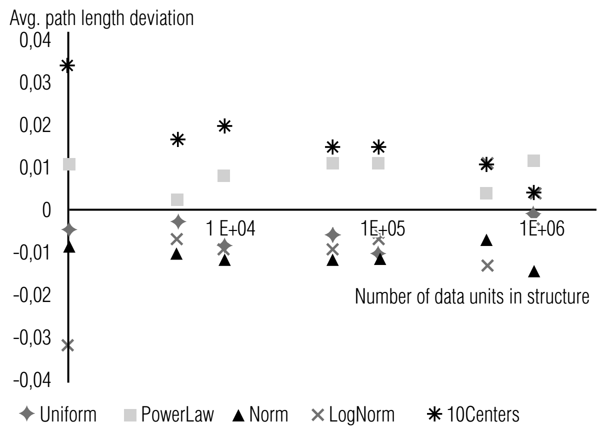

Fig. 3. Deviation of average path for different distributions

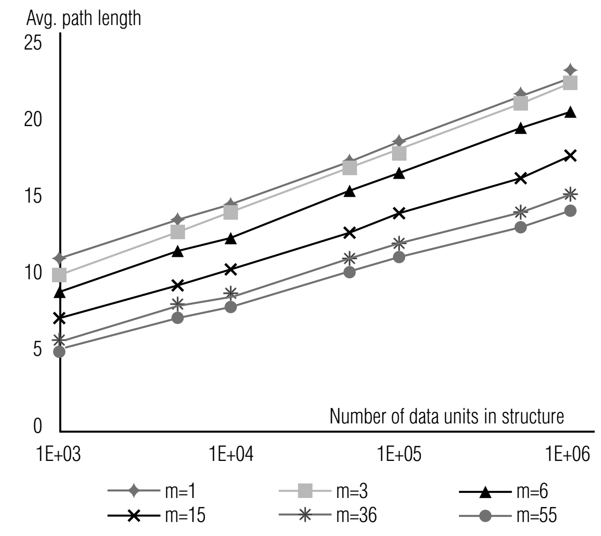

Fig. 4. Average path length of search algorithm

The deviations of the path length from the average value for the different distributions are shown in *Figure 3.*

As can been seen from *Figure 2* and *3*, the structure demonstrates clear logarithmic dependency from the structure size with very small deviation for all distribution types. Independence of different distributions was expected, since all algorithms of the MSW structure do not directly use the value of distance between data units and are based only on the relative order. Therefore, all remaining simulations have been performed on the sets of data units whose images have been uniformly distributed on the interval (0, 1).

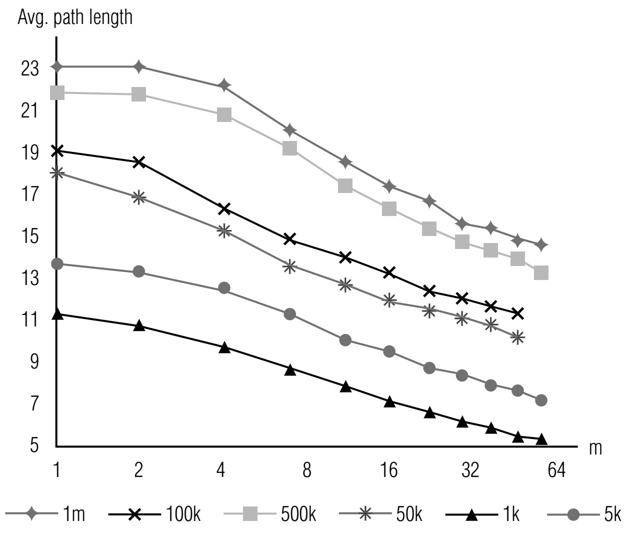

Fig. 5. Dependency of path length for parameter m
for fixed structure sizes 1K-1M

Figure 4 and *5* show dependency of the path length on the parameter *m*. F*igure 4* presents results for various structure sizes and *Figure 5* for six different fixed sizes: 1 million, 500 thousand, 100 thousand, 50 thousand, 5 thousand and 1 thousand data units. As can be seen from *Figure 5*, the dependence of the average path length on the parameter *m* is close to inverse logarithmic law.

In order to study how the network nodes, which are presented as data units, may be loaded during the search, we have measured how often the search algorithm visits the particular data unit. The value of loading was calculated as a fraction of the number of searches in which the particular data unit has been involved to the total number of searches. As can be seen from *Figure 10*

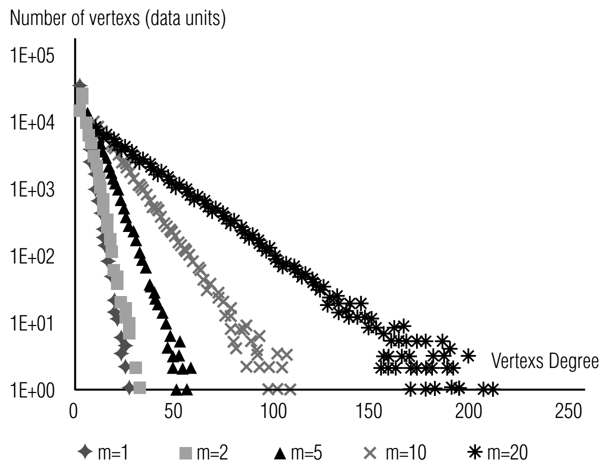

Fig. 6. Vertex degree distribution

and *11*, the maximum value of the data unit load is independent of the structure size and the fraction of load for a particular data unit has exponential dependence on the vertex degree (number of links to other data units). This indicates that additional links increase the diversity of search paths, thereby lowering the load on the data units with larger number of links, all of which helps to avoid bottleneck problems.

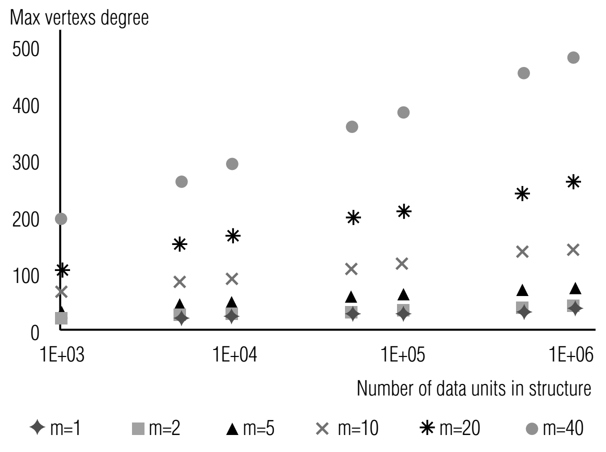

Fig. 7. Maximum vertex degree

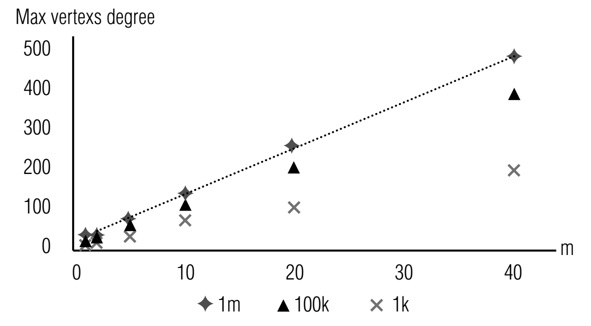

Fig. 8. Maximum vertex degree from parameter m for 1 million, 100K and 1K data units in structure

The dependence of the maximum fraction load on the parameter m is plotted in *Figure 9* . The study of vertex degrees is presented in *Figure 6*, 7 and *8*. It is easily seen that the maximum vertex degree has a logarithmic dependence on the structure size (*Figure 8*) and has a linear dependence on the parameter m. *Figure 6* shows that the distribution of vertex degrees follows exponential law.

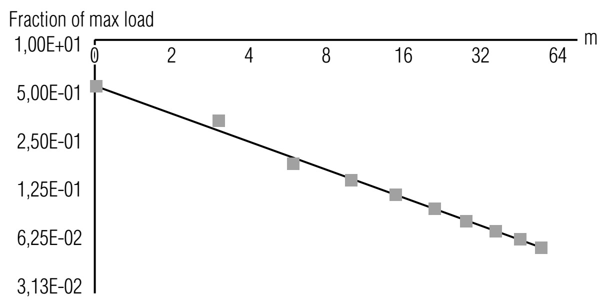

Fig. 9. Fraction of Max Load from parameter m for 100K data units in structure

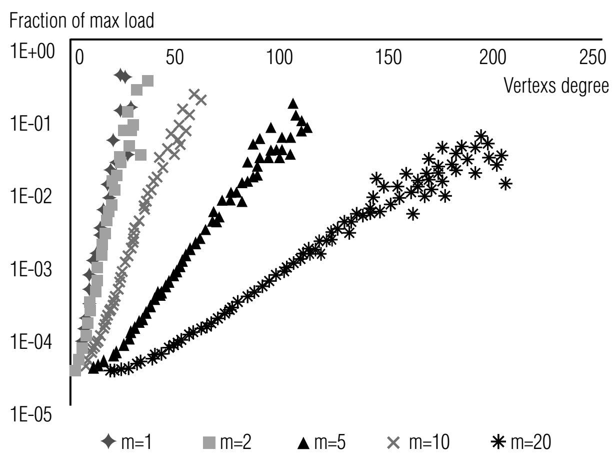

Fig. 10. Data unit load vs the number of links for 100K data units in the structure

We have also investigated the stability property of the structure, i.e. its ability to work in case of node removal. *Figure 12* shows the average percentage of the number of nodes which should be removed (randomly with uniform distribution) to split the network into two components of connectivity. The average percent of removed data units follows the power law from the total number of data units in the structure. The value of the exponent in this power law also obeys the power law with the exponent equal to -1.24 empirically (*see Figure 13*). Therefore, we observe a relation $\frac{r}{n} \sim n^{c\cdot m^{-1.24}}$, where c is some constant ~ 1.7 and r is the expected number of removed data units for which the network is split.

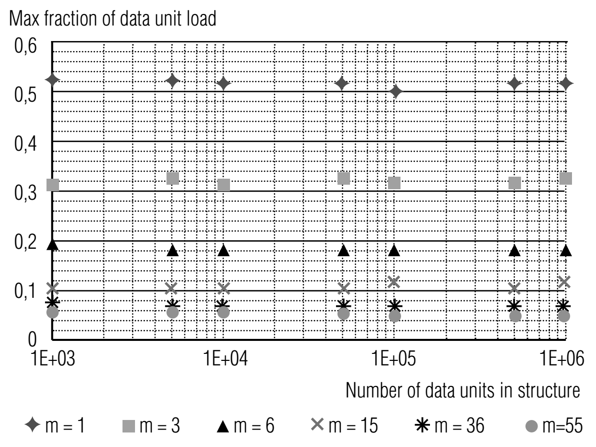

Fig. 11. Maximum data unit load

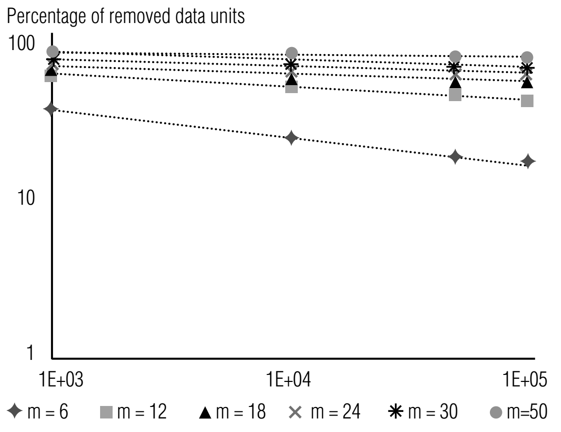

Fig. 12. The average percentage of how many nodes should be removed randomly with uniform distribution to split a network into two components of connectivity

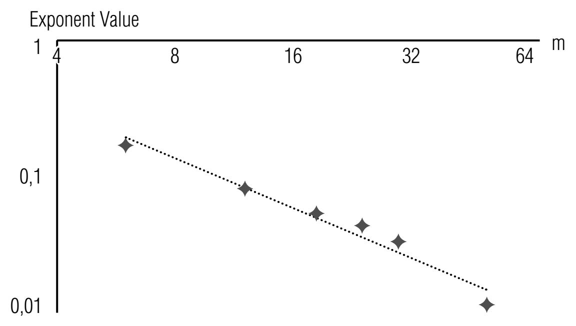

Fig. 13. The value of the exponent in the law of dependency on the number of removed data units from the structure size

Conclusion

In this paper, we have introduced a variant of Metrized Small World data structure which allows exact searching for data units by a single unique string attribute. We have described data insertion and data search algorithms and have demonstrated that the structure has $O\left(\frac{\log n}{m^{\alpha}}\right)$ search complexity, where empirically $\alpha \sim 0.11$.

We have provided experimental data that demonstrates the scalability and decentralization properties of the structure. Our contributions can be summarized as follows:

✦ overlay structure on the data unit level with any number of entry points providing $O\left(\frac{\log n}{m^{\alpha}}\right)$ search complexity using a simple greedy search algorithm;

✦ the structure is independent of data distribution;

✦ an incremental data unit insertion algorithm modifies only a small amount of data units;

✦ the short paths for the greedy search algorithm are formed without explicitly maintaining any particular distribution of the length of links;

✦ physical allocation of data units is independent of data content and search functionality;

✦ dynamic insertion of new network nodes does not need a relocation of data units on existing nodes;

✦ search paths are distributed in a way that avoids overloading single data units.

Acknowledgments

This work was conducted in National Research University Higher School of Economics and supported by RSF grant 14-41-00039. ■

References

1. Albert R., Barabasi A.L. (2002) Statistical mechanics of complex networks. *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97.
2. Amaral L.A.N., Scala A., Barthelemy M., Stanley H.E. (2000) Classes of small-world networks. *Proceedings of the National Academy of Sciences (USA)*, vol. 97 (21), pp. 11149–11152.
3. Balakrishnan H., Kaashoek M.F., Karger D., Morris R., Stoica I. (2003) Looking up data in P2P systems. *Communications of the ACM*, vol. 46, no. 2, pp. 43–48.
4. Bondi A.B. (2000) Characteristics of scalability and their impact on performance. Proceedings of the *2nd International Workshop on Software and Performance (WOSP 2000), 17-20 September 2000, Ottawa, Canada*, pp. 195–203.
5. Cattell R. (2011) Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, vol. 39 (4), pp. 12–27.
6. Dean J., Ghemawat S. (2008) Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, vol. 51, no. 1, pp. 107–113.
7. Harizopoulos S., Abadi D.J., Madden S., Stonebraker M. (2008) OLTP through the looking glass, and what we found there. Proceedings of the *2008 ACM SIGMOD International Conference on Management of Data (SIGMOD 2008), 9-12 June 2008, Vancouver, Canada*, pp. 981–992.
8. Karger D., Lehman E., Leighton T., Panigrahy R., Levine M., Lewin D. (1997) Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. Proceedings of the *29th Annual ACM Symposium on Theory of Computing, ACM, 4-6 May 1997, El Paso, Texas, USA*, pp. 654–663.
9. Krylov V., Logvinov A., Ponomarenko A., Ponomarev D. (2008) Active database architecture for XML documents. Proceedings of the *21st International Conference on Computer Applications in Industry and Engineering (CAINE 2008), 12-14 November 2008, Honolulu, Hawaii, USA,* pp. 244–249.
10. Krylov V., Logvinov A., Ponomarenko A., Ponomarev D. (2008) Metrized small world properties data structure. Proceedings of the *17th International Conference on Software Engineering and Data Engineering (SEDE 2008), 30 June - 2 July 2008, Los Angeles, California, USA*, pp. 203–208.
11. Logvinov A., Ponomarenko A., Krylov V., Malkov Y. (2010) Metrized small world approach for nearest neighbor search. Proceedings of the *2010 Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2010), 1-2 June 2010, Nizhny Novgorod, Russia*, vol. 4, pp. 151–156.
12. Malkov Y., Ponomarenko A., Logvinov A., Krylov V. (2014) Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, no. 45, pp. 61–68.
13. Malkov Y., Ponomarenko A., Logvinov A., Krylov V. (2012) Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. *Lecture Notes in Computer Sciene – Similarity Search and Applications*, vol. 7404, pp. 132–147.
14. Maymounkov P., Mazieres D. (2002) Kademlia: A peer-to-peer information system based on the XOR metric. *Peer-to-Peer Systems*, vol. 2429, pp. 53–65.
15. Rabl T., Gomez-Villamor S., Sadoghi M., Muntes-Mulero V., Jacobsen H.A., Mankovskii S. (2012) Solving big data challenges for enterprise application performance management. *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1724–1735.
16. Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H. (2001) Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160.
17. Tarkoma S. (2010) *Overlay networks: Toward information networking.* Boca Raton, FL: CRC Press.
18. Tolia N., Kozuch M., Satyanarayanan M., Karp B., Bressoud T.C., Perrig A. (2003) Opportunistic use of content addressable storage for distributed file systems. Proceedings of the *2003 USENIX Annual Technical Conference (USENIX 2003), General Track, 9-14 June 2003, San Antonio, Texas, USA*, vol. 3, pp. 127–140.
19. Watts D.J. (1999) *Small worlds: The dynamic of networks between ordered and randomness.* New Jersey: Princeton University Press.

Оверлейная сеть для распределенного точного и интервального поиска в одномерном пространстве

А.А. Пономаренко
научный сотрудник, Лаборатория алгоритмов и технологий анализа сетевых структур
Национальный исследовательский университет «Высшая школа экономики»
Адрес: Российская Федерация, 603093, г. Нижний Новгород, ул. Родионова, 136
E-mail: aponomarenko@hse.ru

Ю.А. Мальков
младший научный сотрудник
Институт прикладной физики, Российская академия наук
Адрес: Российская Федерация, 603950, г. Нижний Новгород, ул. Ульянова, 46
E-mail: yurymalkov@mail.ru

А.А. Логвинов
руководитель проектов, ООО «МераЛабс»
Адрес: Российская Федерация, 603093, г. Нижний Новгород, ул. Родионова, 192
E-mail: alogvinov@meralabs.com

В.В. Крылов
доктор технических наук, профессор, заведующий лабораторией больших данных
Нижегородский государственный технический университет им. Р.Е.Алексеева
Адрес: Российская Федерация, 603950, г. Нижний Новгород, ул. Минина, 24
E-mail: vkrylov@heterarchica.com

Аннотация

Для современной компьютерной системы возможность масштабироваться является необходимым бизнес-требованием. Распределенные системы явно демонстрируют это свойство, как и способность легко обрабатывать сверхбольшие объемы данных. Многие системы с распределенной архитектурой имеют в основе распределенную хэш-таблицу (distributed hash table, DHT), которая используется в качестве дополнительной логической сети, объединяющей множество распределенных серверов. Эта логическая сеть используется для поиска узлов и распределения задач между ними. Главное преимущество такого подхода – отсутствие какого-либо центрального элемента или узла, который обладал бы информацией о структуре всей сети. Поиск узлов в сети происходит путем передачи поискового сообщения от одного узла к другому. Несмотря на то, что каждый узел «знает» только небольшое число других узлов, сеть организованна таким образом, что процедура поиска затрагивает логарифмическое число узлов.

Существует несколько реализаций концепции DHT, которые регламентируют, каким образом логическая сеть строится и поддерживается. В этой работе мы демонстрируем, как такая сеть может быть сконструирована очень простым способом, с применением (с небольшими изменениями) недавно опубликованного алгоритма метризованного тесного мира (Metrized Small World) для случая одномерного метрического пространства. В работе мы приводим теоретический анализ для случая равномерного распределения данных и эмпирический анализ для других распределений. Главным преимуществом предложенного алгоритма перед аналогами является его устойчивость к различным распределениям данных и отсутствие необходимости поддержания распределения длин ссылок, определенного явным образом.

Также в работе описывается, каким образом можно полностью отделить физическое размещение данных от поисковой функциональности. Это разделение важно, например, для построения глобальных хранилищ данных, данными которых владеют несколько сторон, причем каждая сторона заинтересована в том, чтобы иметь полный контроль над физическим размещением и доступу к своим данным. В отличие от DHT, добавление новых данных не требует их перемещение на другие узлы.

Ключевые слова: поиск ближайшего соседа, метрическое пространство, распределенные вычисления, Интернет-технология и приложения, структура данных, алгоритм.

Цитирование: Ponomarenko A.A., Malkov Yu.A., Logvinov A.A., Krylov V.V. An overlay network for distributed exact and range search in one-dimensional space // Business Informatics. 2016. No. 1 (35). P. 26–36.
DOI: 10.17323/1998-0663.2016.1.26.36.

Литература

1. Albert R., Barabasi A.L. Statistical mechanics of complex networks // Reviews of Modern Physics. 2002. Vol. 74. No. 1. P. 47–97.
2. Amaral L.A.N., Scala A., Barthelemy M., Stanley H.E. Classes of small-world networks // Proceedings of the National Academy of Sciences (USA). 2000. Vol. 97 (21). P. 11149–11152.
3. Looking up data in P2P systems / H.Balakrishnan [et al.] // Communications of the ACM. 2003. Vol. 46. No. 2. P. 43–48.
4. Bondi A.B. Characteristics of scalability and their impact on performance // Proceedings of the 2nd International Workshop on Software and Performance (WOSP 2000), 17-20 September 2000, Ottawa, Canada. 2000. P. 195–203.
5. Cattell R. Scalable SQL and NoSQL data stores // ACM SIGMOD Record. 2011. Vol. 39 (4). P. 12–27.
6. Dean J., Ghemawat S. Mapreduce: Simplified data processing on large clusters // Communications of the ACM. 2008. Vol. 51. No. 1. P. 107–113.
7. Harizopoulos S., Abadi D.J., Madden S., Stonebraker M. OLTP through the looking glass, and what we found there // Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD 2008), 9-12 June 2008, Vancouver, Canada. 2008. P. 981–992.
8. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web / D.Karger [et al.] // Proceedings of the 29th Annual ACM Symposium on Theory of Computing, ACM, 4-6 May 1997, El Paso, Texas, USA. 1997. P. 654–663.
9. Krylov V., Logvinov A., Ponomarenko A., Ponomarev D. Active database architecture for XML documents // Proceedings of the 21st International Conference on Computer Applications in Industry and Engineering (CAINE 2008), 12-14 November 2008, Honolulu, Hawaii, USA. 2008. P. 244–249.
10. Krylov V., Logvinov A., Ponomarenko A., Ponomarev D. Metrized small world properties data structure // Proceedings of the 17th International Conference on Software Engineering and Data Engineering (SEDE 2008), 30 June - 2 July 2008, Los Angeles, California, USA. 2008. P. 203–208.
11. Logvinov A., Ponomarenko A., Krylov V., Malkov Y. Metrized small world approach for nearest neighbor search // Proceedings of the 2010 Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2010), 1-2 June 2010, Nizhny Novgorod, Russia. 2020. Vol. 4. P. 151–156.
12. Malkov Y., Ponomarenko A., Logvinov A., Krylov V. Approximate nearest neighbor algorithm based on navigable small world graphs // Information Systems. 2014. No. 45. P. 61–68.
13. Malkov Y., Ponomarenko A., Logvinov A., Krylov V. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces // Lecture Notes in Computer Sciene – Similarity Search and Applications. 2012. Vol. 7404. P. 132–147.
14. Maymounkov P., Mazieres D. Kademlia: A peer-to-peer information system based on the XOR metric // Peer-to-Peer Systems. 2002. Vol. 2429. P. 53–65.
15. Solving big data challenges for enterprise application performance management / T.Rabl [et al.] // Proceedings of the VLDB Endowment. 2012. Vol. 5. No. 12. P. 1724–1735.
16. Chord: A scalable peer-to-peer lookup service for internet applications / I.Stoica [et al.] // ACM SIGCOMM Computer Communication Review. 2001. Vol. 31. No. 4. P. 149–160.
17. Tarkoma S. Overlay networks: Toward information networking. Boca Raton, FL: CRC Press, 2010. 237 p.
18. Opportunistic use of content addressable storage for distributed file systems / N.Tolia [et al.] // Proceedings of the 2003 USENIX Annual Technical Conference (USENIX 2003), General Track, 9-14 June 2003, San Antonio, Texas, USA. 2003. Vol. 3. P. 127–140.
19. Watts D.J. Small worlds: The dynamic of networks between ordered and randomness. New Jersey: Princeton University Press, 1999. 168 p.