

Single-attribute Distributed Metrized Small World Data Structure

V. Krylov, D. Ponomarev, A. Logvinov, A. Ponomarenko

MERA Labs

Nizhny Novgorod, Russia

vkrylov@meralabs.com, dmp@meranetworks.com, alogvinov@meralabs.com, aponom@meralabs.com

Modern applications set the requirement to store huge (on the order of petabytes) and constantly growing amounts of data. We propose the distributed scalable Metrized Small World data structure which is capable of satisfying this requirement. In this paper we discuss the basic variant of the structure which can store data units with a single searchable attribute in the form of a unique string and any amount of non-searchable data. One important application of this form of the structure are content-addressable systems (CAS) which map a content-derived identifier to the content element. Simulation results demonstrate that the proposed structure has logarithmical search complexity, avoids access bottlenecks and can be dynamically scaled to large numbers of data units.

Database; distributed computing; internet technology and applications

I. INTRODUCTION

The contribution of this paper is the distributed decentralized scalable data structure which provides $O(\log N)$ search time and completely separates the concept of network location of the data from the search functionality. This separation is important, for an 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. The functionality of the proposed variant of the structure is similar to those of a distributed hash table (DHT) [7] – searching for a data unit by a unique string attribute. The difference from the consistent hashing approach [9] used by DHTs is that addition of a new network node does not require relocation of data units on the existing data nodes.

The proposed structure has two key properties. First, the data units form a small world overlay network which ensures that any data unit is reachable from any other data unit in a small number of steps. This creates the possibility to quickly find any data unit starting from any other data unit. Second, all data units are ordered and we define the metric (distance) between two data units. This allows us to build the structure in a way that a simple greedy search algorithm can be used to find any data unit.

The proposed data addition algorithm is incremental, i.e. the addition of a new data unit affects only a small number of existing data units.

The rest of the paper is structured as follows. Section II describes the Metrized Small World overlay data structure and its mapping a the network node collection. Section III

describes the data search algorithm and demonstrates that it has $O(\log N)$ search complexity. Section IV specifies the data addition algorithm. Section V provides experimental data and gives a brief analysis that shows that the proposed structure is scalable and avoids data unit access bottlenecks by routing search paths through different data units. Finally we summarize our contributions in Section VI.

II. METRIZED SMALL WORLD DATA STRUCTURE

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 any other node), each node containing 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, which allows for instant identification of a network node containing the data unit. This identifier dependence makes it possible to create an overlay structure on data unit level which is inherently mapped to the physical network nodes. Each data unit stores a mutable set of identifiers of other data units, which are the links that constitute the overlay data structure. 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 Fig 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 from the data units, the nodes do not have direct links to each other.

The algorithms of data unit addition and search operate solely on the overlay structure formed by data units and links between them and 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 addition algorithm which is executed after the new data unit is allocated. The addition of a new node does not affect the existing overlay structure or the distribution of data units

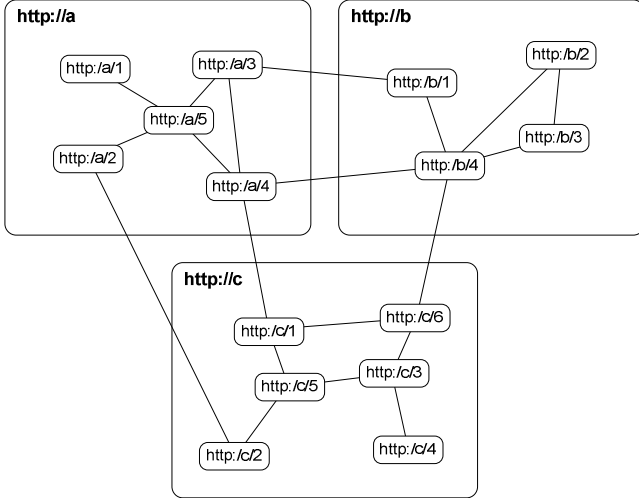


Figure 1. Distributed Metrized Small World data structure.

between nodes; it merely provides new allocation space and processing power for addition and search algorithms.

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 on each step which would presumably decrease the number of steps from the current data unit to the result.

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 two data units in the structure.
2. A criterion must exist which would allow to decide which link on 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 local minimums when possible.

To address the first requirement we have developed the data unit addition algorithm that incrementally produces a structure with small world properties. The small world graphs [3, 4, 5] 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 we have created a metric (proximity measure) between data units which is consistently used by the addition and search algorithms. The choice of the metric calculation algorithm is greatly dependent of the structure of data units and whether the metric proximity should reflect the proximity properties of the application domain. If the metric does reflect application domain proximity, one can do not only full match searches but also search for inexact but useful matches. For the purposes of this paper we only consider the properties of the metric per se, without connection to the application domain.

A detailed description of the MSW structure is given in our paper [1]; technological aspects of a scalable database system based on the MSW structure are covered in [2].

A data unit contains searchable and non-searchable information. For the purposes of this paper the searchable information consists of only a single unique character string. However simple, this approach is useful for systems like CAS (Content Addressable Storage) [6] 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, using which the data object is retrieved. In terms of 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 the result of search process, the content-derived identifier will be transformed into the identifier of the data unit which stores the content.

Another comparable solutions are the distributed hash table (DHT) [7] systems such as Chord [8]. These systems use consistent hashing approach [9] to decide which network node will store the data unit. Addition 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, addition of a new node does not affect existing nodes at all because allocation of data units is completely irrelevant to the search and addition 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 existing nodes. Another difference from the DHT systems is that MSW creates an overlay network on data unit level while the overlay network of DHT systems like Chord is built on node level.

For the purposes of the description of algorithms, we define the bijectional mapping between string values and real numbers in range $(0,1)$. Assuming that the string value is in the form of character array $S = (s_1, \dots, s_n)$, we define the mapping $C(S) = \sum_{i=1}^n \frac{s_i}{B^i}$ where B is the maximum character value plus one. This mapping has the useful property that the order of string images corresponds to the lexicographical order of the strings. The original string can be obtained from the image by iteratively multiplying it by B and taking the integral part as the next character value until nothing remains to be multiplied. The mapping requires infinite floating point precision of the image in order to be able to reconstruct the original string but such reconstruction is not used by the search and addition algorithms. The metric between two strings is defined as $M(S_1, S_2) = |C(S_2) - C(S_1)|$.

III. DATA SEARCH ALGORITHM

The goal of the search algorithm is to find a string identical to a given string S_q starting from the entry point data unit D_{ep} , or to determine that the search string is not present in the structure. For brevity, we will treat the each data unit as if it were the same entity as the searchable string which it contains.

1. Let $D_{cur} = D_{ep}$.
2. If $M(D_{cur}, S_q) = 0$ then D_{cur} is the result.

3. For each neighbor D_i of D_{cur} calculate $M_i = M(D_i, S_q)$.
4. If $\min(M_i) > M(D_{cur}, S_q)$ then there is no matching data unit in the structure.
5. 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 with its direct predecessor and successor in the order defined by natural order of images $C(D_i)$ by the addition algorithm. This ensures absence of local minimums in the structure.

Next we will show that the complexity of this search algorithm grows logarithmically with the number of data units in the structure. Assume that the images $C(D_i)$ are uniformly distributed on the interval $(0,1)$. 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 N first generation data units in the structure. 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. Thereby we ignore the unlikely cases of very large gaps between adjacent first generation data units.

Next we add N second generation data units, i.e. double the number of data units. Consider the largest interval between adjacent first generation data units. The number of second generation data units k falling into the interval conforms to the Poisson distribution $p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$ 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 of number of steps required to reach any data unit from any other data unit after the addition 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 is 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 second generation data units hence the former links are preferable over 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 addition 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 N is the number of data units in the structure.

IV. DATA ADDITION ALGORITHM

The purpose of the data unit addition algorithm is to connect the newly allocated data unit to the structure by populating its list of links and respectively appending link lists of the data units chosen to be connected with the new data unit. The process of new data unit addition does not alter the links between the existing data units. The idea behind the algorithm is to connect the new data D_{new} unit with its direct predecessor and direct successor in the structure to prevent the search algorithm from experiencing local minimums and additionally connect it with other m closest data units. D_{ep} is the entry point data unit for the algorithm, which is an arbitrary chosen existing data unit. The algorithm is defined as follows:

1. Let $D_{cur} = D_{ep}$.
2. For each neighbor D_i of 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. Mutually 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 $M(D'_{pre}, D_{new}) < M(D'_{succ}, D_{new})$ mutually connect D_{new} and D'_{pre} and let $D_{pre} = D'_{pre}$.
 - 7.5. If only D'_{succ} exists or $M(D'_{succ}, D_{new}) < M(D'_{pre}, D_{new})$ mutually connect D_{new} and D'_{succ} and let $D_{succ} = D'_{succ}$.

V. SIMULATION AND ANALYSIS

We have built a prototype of MSW structure and performed a simulation with different numbers of data units in the structure and different m parameter values of the addition algorithm. The simulation was performed directly on data unit images uniformly distributed on the interval $(0,1)$. We have simulated different structure sizes in range

from 50 to 500000 data units measuring the following parameters: average shortest path length between two data units, maximum vertex degree, vertex degree distribution and maximum data unit load (percentage of search processes involving a single data unit). The results are shown on Fig. 2-5.

As can be seen on Fig. 2 and 3, both average shortest path length and maximum vertex degree scale logarithmically with the number of data units. Therefore the structure is suitable for storing very large amounts of data. Fig. 4 shows that vertex degree distribution is exponential independently of number of data units or the value of m , which means that the number of high-degree vertices is small compared to the total number of vertices (data units). Fig. 5 shows that the maximum load on a data unit does not grow with the number of data units and decreases with the number m of additional links established by a newly added data unit. This indicates that additional links increase the diversity of search paths thereby lowering the load on the data units with large number of links (vertex degree) which helps to avoid bottleneck problems.

VI. CONCLUSIONS

In this paper we have introduced a variant of scalable distributed Metrized Small World data structure which allows searching for data units by a single unique string attribute. We described the data addition and data search algorithms and demonstrated that the structure has $O(\log N)$ search complexity. We have provided experimental data that confirms scalability and decentralization of the structure.

Our contributions can be summarized as follows:

- Overlay structure on data unit level with any number of entry points providing $O(\log N)$ search complexity using a simple greedy search algorithm.

- Incremental data unit addition algorithm which alters only a small amount of data units on every data unit addition.
- Physical allocation of data units independent of data content and search functionality.
- Dynamic addition of new network nodes without need of relocation of data units on existing nodes.
- Search paths distributed in a way that avoids overloading of single data units.

REFERENCES

- [1] V. Krylov, A. Logvinov, A. Ponomarenko, D.Ponomarev "Metrized Small World Properties Data Structure", Proc. Software Engineering and Data Engineering (SEDE 2008).
- [2] V. Krylov, A. Logvinov, A. Ponomarenko, D.Ponomarev "Active Database Architecture for XML Documents", Proc. Computer applications in Industry and Engineering (CAINE 2008).
- [3] D.J. Watts "Small Worlds", Princeton, New Jersey: Princeton University Press, 1999.
- [4] L.A.N. Amaral, A. Scala, M. Barthelemy and H.E. Stanley, "Classes of small-world networks", Proc. Nat. Acad. Sci. U.S.A., 2000.
- [5] R. Albert and A.-L. Barabasi "Statistical mechanics of complex networks." Rev. Mod. Phys., 74(1): pp. 47-97, January 2002.
- [6] N. Tolia, M. Kozuch, M. Satyanarayanan, B. Karp, T. Bressoud and A. Perrig "Opportunistic Use of Content Addressable Storage for Distributed File Systems" Proc. USENIX Annual Technical Conference, June 2003.
- [7] H. Balakrishnan, M.F. Kaashoek, D. Karger, R. Morris and I. Stoica "Looking Up Data in P2P Systems", In Communications of the ACM, February 2003.
- [8] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", Proc. ACM SIGCOMM, August 2001.
- [9] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine and D. Lewin "Consistent hashing and random trees", Proc. 29th annual ACM symposium on Theory of computing (ACM Press New York, NY, USA): 654-663, 1997.

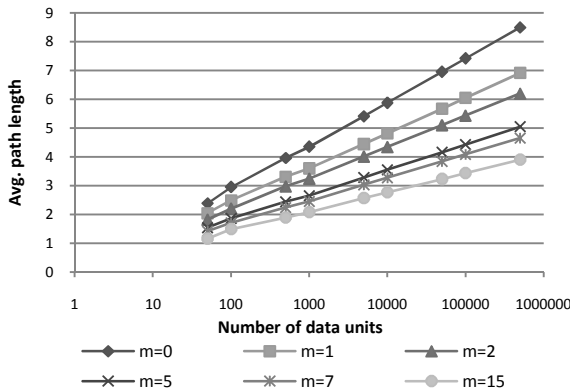


Figure 2. Average shortest path length between two data units.

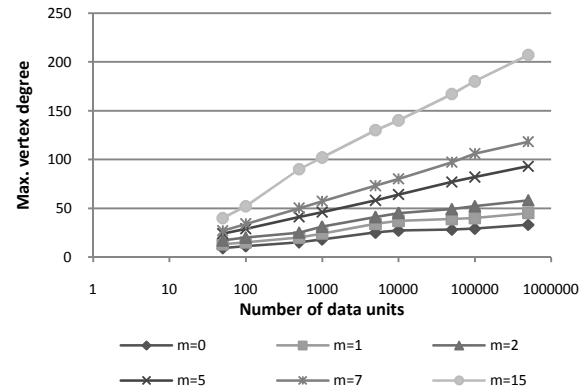


Figure 3. Maximum vertex degree.

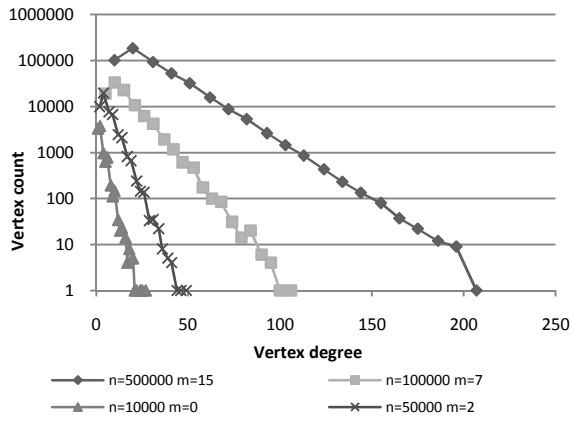


Figure 4. Vertex degree distribution.

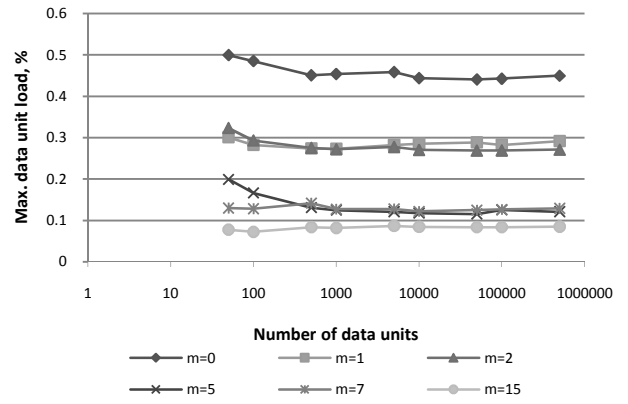


Figure 5. Maximum data unit load.