

# ACTIVE DATABASE ARCHITECTURE FOR XML DOCUMENTS

V. Krylov  
MERA Labs  
Nizhny Novgorod,  
603163, Russia  
+78312722020

vkrylov@meralabs.com

D. Ponomarev  
MERA Group  
Nizhny Novgorod,  
603163, Russia  
+78312788800

dmp@meranetworks.com

A. Logvinov  
MERA Labs  
Nizhny Novgorod,  
603163, Russia  
+78312788865+3125

alogvinov@meralabs.com

A. Ponomarenko  
MERA Labs  
Nizhny Novgorod,  
603163, Russia  
+78312788865+3126

aponom@meralabs.com

## Abstract

We introduce the XML database architecture to build very large, scalable, loosely structured distributed data storage.

Traditionally data is regarded as passive records operated by DBMS software. Our idea is that every data unit should be active, capable of communication with other data units and database clients. Combined with the special overlay structure incrementally formed by data units (Metriized Small World Graph) this provides for effective distribution of data units among database servers and unbounded scalability of the resulting storage ensuring logarithmic search and append complexity.

Each active data unit is represented as an XML document addressable by a unique URL having a locally stored extendable set of XLink links to other data units, and a software module driving the communication with other data units and clients. Search in this structure is performed by sequential and/or parallel crawling following the links in the list obtained on each step.

The active data units communicate by sending XML messages over a transport protocol such as HTTP. The communication includes the retrieval of XML content and link lists, addition of new links, calculating query relevance and work delegation (so that every unit can actively propagate the process initiated or mediated by another unit).

Since there are no central controlling nodes in the structure, multiple processes of adding new data units and searching for existing data can be performed independently and simultaneously, and begin with any existing data unit. Moreover, because the data units are active, these processes may propagate on their own without being fully dictated by originator. This allows distribution of the data processing along with distribution of the data itself.

We have built a prototype implementation of the architecture. The analysis of the small world overlay structure properties confirmed the possibility of building efficient XML data storages which contain hundreds of petabytes of data.

## 1 INTRODUCTION

The traditional approach to building databases can be characterized by the following statements:

1. Data items are considered passive entities operated by the DB management software.
2. Data items are stored in centralized structures with a single entry point.

But modern applications set the requirements which make this model not optimal, namely the requirement to store and process large amount of data generated by steadily increasing number of uncoordinated sources having no common control center and queried by numerous and uncoordinated data consumers. To fulfill this requirement, new data storage architectures are required with the following properties:

1. Decentralization of both data and data access, since the sources of data and queries are uncoordinated and geographically distributed.
2. Ability to retrieve data in relatively short time compared to the large volume of data in the storage.
3. Scalability, because the amount of data and query sources grows constantly.
4. Ability to work with weakly structured data, since imposing restrictions on the structure of data elements generated by numerous uncoordinated sources is impractical.

The Active Data Units section describes the proposed method of data item representation which provides for decentralization and distribution of both data and processes pertaining to the data. The Overlay Metriized Small World Structure section describes the way to connect data items which allows for finding the relevant data in relatively short time even when number of data items is huge and constantly growing, permitting a multitude of search and addition processes occurring at the same moment of time. In the Processes section we describe how the aforementioned processes are performed. Finally, the Simulation section provides analysis of the properties of Metriized Small World Overlay structure.

## 2 ACTIVE DATA UNITS

Each active data unit (ADU) consists of the following components:

1. XML content.
2. List of XLink links to other ADUs.
3. Message processing module (the active component).

Each of these components has a unique URI which is a sub-URI of the base URI of the active data unit, so that if the URI of any of components is known, the URIs of other components can be trivially calculated. Links in the list (2) point to the base URIs of other ADUs. The links to other ADUs are not semantically related to or embedded in the XML content and serve no purpose other than creating the overlay network to facilitate the search process. The message processing module is represented by an in-memory object (class instance) containing the module state which is linked to the executable code shared by message processing modules of all ADUs on the same server. Thus the state of the message processing module is the component of the ADU to which it belongs while the executable code of the module is a shared server component.

ADUs communicate with each other and with non-ADU actors (such as database clients) by means of XML messages. Each message contains the following information:

1. Message identifier which determines the semantics of processing of the message.
2. List of string parameters which further specify the details of processing.
3. List of attached XML documents involved in the processing of the message.

Messages are exchanged in request-response manner, which means that after a message is processed, the response message is returned to the sender of the original message. If HTTP is used as transport protocol, the original message would be sent in an HTTP POST request and the response message would be received in the HTTP response. The latter means that non-ADU agents are not required to have URIs in order to send messages to and obtain responses from ADUs, but cannot receive messages other than responses to their own messages.

Active data units use resources of the server which hosts them and their base URIs are sub-URIs of the server base URI. Therefore at the network level the storage can be viewed as a set of servers processing ADUs and exchanging messages pertaining to those ADUs, which is different from the approach described in [1] in that all servers play the same role (there are no “superpeers”). But ultimately the ADUs, not the servers, are the endpoints of communication; hence the

distribution of ADUs between servers is irrelevant to the semantics of the communication, influencing only the time which is necessary for the messages to reach the destination. The communication of ADUs processed by the same server is logically indistinguishable from the communication of ADUs processed by different servers but can be significantly faster because the network is not involved. Thereby the proposed architecture can be considered as an unstructured p2p system with storage and execution autonomy according to the classification provided in [2].

The following operations are possible on ADUs:

1. Retrieve XML content of the ADU.
2. Retrieve the list of links of the ADU.
3. Add a link to the list of links of the ADU.
4. Calculate the metric value between the XML content of the ADU and a given XML document (the metric values are described in the section 3).

Each operation is initiated by sending a message with the corresponding identifier to the target ADU. The result of the operation is retrieved in the response message.

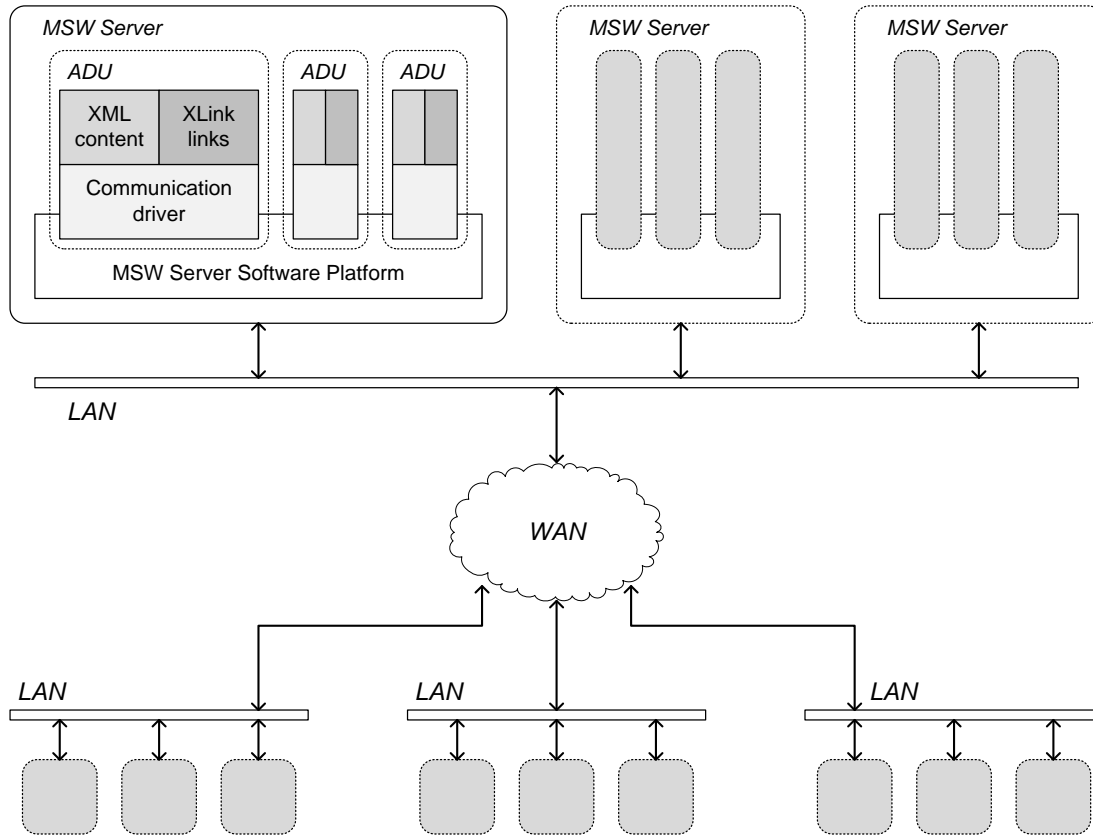
## 3 OVERLAY METRIZED SMALL WORLD STRUCTURE

The overlay Metrized Small World graph structure (MSW, described in greater detail in [3]) is formed by the ADUs which represent the vertices of the graph and the links locally stored in each ADU which represent the edges of the graph. For every link (A, B) stored in ADU A, there is a reverse link (B, A) stored in the ADU B, so the MSW graph can be considered undirected. No ADU can contain a link to itself.

To ensure that the overlay structure formed by ADUs has the properties listed in the Introduction section, the following requirements must be met:

1. Any ADU in the structure must be able to serve as an entry point of access to other elements. This is necessary for decentralization since having fixed entry points would create a bottleneck for data access.
2. A relatively short path must exist between any two ADUs to ensure search in a small number of operations compared to the number of ADUs.
3. Any ADU must store a relatively small number of links so that the access to individual links will not present storage and search complexity problems in itself.

There are also requirements pertaining to the method of construction of the structure rather than to the structure itself:



**Figure 1. Active distributed MSW XML database architecture.**

1. The structure must be formed incrementally with the addition of new ADUs, i.e. the addition of a new ADU must not require reconfiguration of the links between previously added ADUs.
2. The addition of a new ADU must not require iteration over the whole set or a large subset of ADUs.

Nowadays the structures are known which satisfy the three requirements above: the Small World graphs [4, 5, 8]. It was noted that the Small World graphs have the following properties:

1. Power law vertex degree distribution.
2. The average shortest path length between two vertices proportional to the logarithm of the number of vertices.
3. The clustering coefficient independent of the number of vertices.

The existing methods of construction of Small World graphs described in [9] are not suitable because they fail to satisfy the second structure formation requirement above by requiring the iteration over the whole set of vertices each time a new vertex is added. There is also a way to reorganize the data in a peer to peer system to establish the small world properties described in [10], but it fails to satisfy both requirements on the construction of structure by requiring both iteration over

the whole set of links and rewiring of the existing links. The dynamic incremental Small World graph construction method in application to MSW structure which satisfies both requirements is described in the Processes section.

The Small World graph structure is a necessary prerequisite for effective search since it ensures the existence of a short sequence of links between any two ADUs. But there also must be a way to find this sequence without trying any possible direction on each step, i.e. there must be some useful measure of proximity of an ADU to the search pattern which will govern the choice of the next link on each step. Furthermore, a proximity measure between ADUs is necessary for construction of the structure in order to make the relative position of the elements consistent with the search algorithm, i.e. to create useful pattern-to-ADU proximity gradients and to clusterize similar ADUs. The latter means that ADUs which are measured as close to each other are separated by a smaller than average number of links.

We propose the unified proximity measure between ADUs or between the search pattern and an ADU which we call the semi-metric. The semi metric is calculated between two arbitrary XML documents, i.e. either between the contents of two ADUs or between the content of an ADU and the search pattern. The search pattern is considered to have the same structure as ADUs

which are expected to match the pattern having relevant XML elements or attributes set to specified values and irrelevant XML elements omitted from the pattern.

Figure 1 gives an overview of the proposed architecture on the component level. Multiple ADUs are hosted by each MSW server and share the common server software platform. The servers are connected by a LAN allowing multiple servers per a single local area network. Ultimately each server can connect to any other server via WAN.

## 4 PROCESSES

There are three possible processes in the MSW structure:

1. Modifying existing ADUs.
2. Searching for existing ADUs.
3. Adding new ADUs.

The first process is performed by marking the ADU being modified as obsolete and creating a new version of that ADU with different URI and modified data. While this may be considered inefficient from the storage space usage point of view, this approach preserves the existing links of the ADU being marked as obsolete thereby retaining the small world properties of the structure without link rewiring. MSW-preserving obsolete ADU removal is a subject of further research. The ADUs marked as obsolete will not appear in search results but still can serve as intermediate points in a search process. There is also the possibility to use these obsolete ADUs to query the data pertaining to the particular moment in the past by returning obsolete versions of ADUs instead of the actual ones if the actual ones were added after the specified moment of time.

Assuming that the search process can be reduced to set operations on the sets of results of searching ADUs which are relevant to a set of patterns, we will further consider the search process as process of finding ADUs relevant to a single search pattern.

The search process in the MSW storage starts from an arbitrarily selected ADU (the entry point) and continues by following the links obtained on each step until the relevant ADU is reached or the time or step number quota is exhausted. If more than one relevant ADU is expected, the process can be branched after acquiring the first result ADU to obtain the entire set or a subset of relevant results. The relevance of an ADU to the query can be expressed as a function of the semi-metric value between the ADU and the query, which is the key difference from the hash-like approaches described in [6] and [7] where only complete key matches are found. The branched process is limited by the boundary of the relevant ADU cluster. Additionally, a maximum depth constraint can be applied to the branches to further limit the number of obtained results and search time.

We propose the following algorithm of adding new ADUs to the existing MSW structure, which is also the algorithm by which the MSW structure is incrementally constructed. In this algorithm  $n$  is the number of algorithm steps per single addition (determines MSW characteristics preservation),  $m$  is the number of links established by the ADU being added,  $n \geq m$ .

Let's assume that the structure already contains  $i - 1$  ADUs and we want to add the  $i$ -th ADU. Then the algorithm is as follows:

1. Arbitrarily select an ADU  $v_k$  where  $1 \leq k \leq i - 1$ .
2. Let VisitedList be the set of visited ADUs.
3. Let CandidateList be the set of candidate ADUs for link establishment sorted by value of semi-metric to  $v_i$  in ascending order.
4. Assume that CandidateLists initially contains only  $v_k$ .
5. For  $j < -1$  to  $n$  do
  - a. Sort CandidateList by value of semi-metric to  $v_i$  in ascending order.
  - b. Select the first ADU  $p$  from CandidateList not contained in VisitedList. If no such ADU exists then break.
  - c. Add  $p$  to VisitedList.
  - d. Add the set of  $p$  neighbor ADUs to CandidateList.
6. Mutually connect the  $v_i$  ADU with  $m$  arbitrary ADUs from VisitedList.

One important consequence of the decentralization of the MSW structure is that processes in it can be performed entirely independently and simultaneously since they are unlikely to involve the same ADUs. If the process  $P1$  involves the set of ADUs  $A1$  and the process  $P2$  involves the set of ADUs  $A2$ , they can occur totally asynchronously (i.e. not causing each other to wait for a common resource) if  $A1 \cap A2 = \emptyset$ . If  $A1 \cap A2 = A_c \neq \emptyset$ , a separate synchronization lock will be required on each ADU from  $A_c$ , assuming the worst case of ADUs processing messages sequentially. This will only cause either  $P1$  or  $P2$  to wait for a lock if they happen to access the same element from  $A_c$  simultaneously.

The processes in the MSW database can be driven either externally or internally. In the former case a database client communicates individually with each ADU involved in the process, making the decision on each step regarding how the process will continue. The ADUs only respond to the messages described in the Active Data Units section without initiating any messages themselves. In this approach the client has full control over the addition and search processes, which has the disadvantage of the fact that consistence of the MSW

structure is dependent on the compliance of each client with the addition algorithm described above.

The internal approach to the processes control consists in creation of a special ADU containing the search pattern which does not participate in the MSW structure, but communicates with the ADUs of the MSW structure to obtain search results in the same way as a client would communicate with ADUs in the external approach. The communication of the search pattern ADU and the client can be arranged as follows:

1. The client creates the search pattern ADU  $As$ .
2. The client sends the search initiation message  $Msi$  to  $As$  and waits until  $As$  responds.
3.  $As$  performs the search process and sends the response message  $Msr$  back to the client containing the set  $Sr$  of URIs of the result ADUs in the form of an attached XML document containing XLink links to each result ADU.
4. The client reads the result set  $Sr$  from the  $Msr$  message and retrieves the contents of the result ADUs.

The addition process using the internal approach is performed in a similar way, but the acting entity is the ADU being added.

The advantage of the internal approach is that all actions of the ADUs are controlled by the server software as opposed to the client software in the external approach. Therefore the responsibility for MSW structure consistency lies on the servers, not on clients, which is preferable from the security and reliability point of view.

## 5 SIMULATION

We provide the results obtained from execution of the proposed ADU addition algorithm using XML media item descriptions as test data. Structures containing 1000, 5000 (not shown on the graph), 10000 and 20000 elements were assembled using this algorithm. The graphs of vertex degree and shortest path length distributions are shown on Figures 2-4. On Figure 5 you can see how the average shortest path length between vertices changes as the number of vertices in the structure increases.

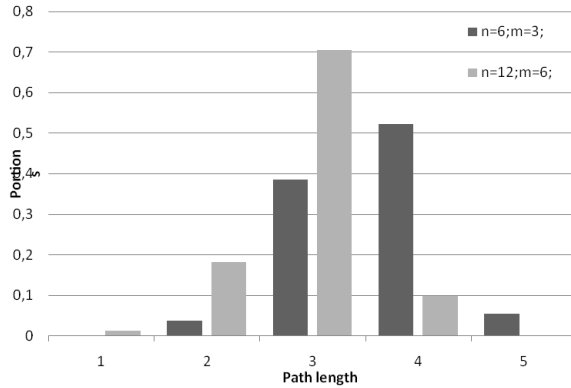
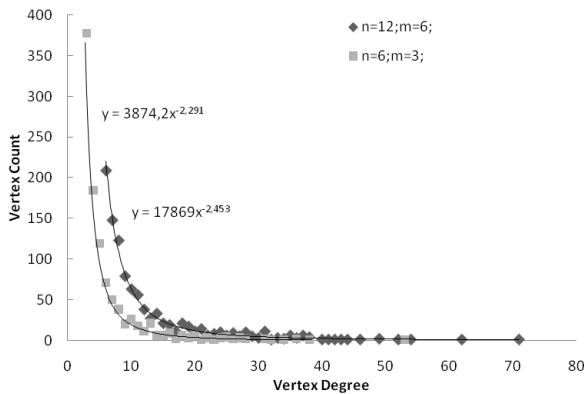


Figure 2. Vertex degree distribution and shortest path length distribution. The number of vertices in the structure is 1000.

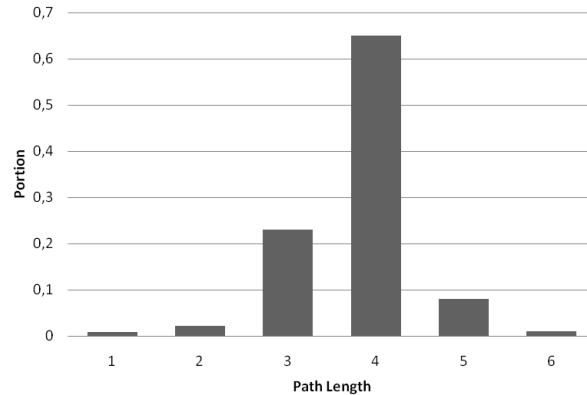
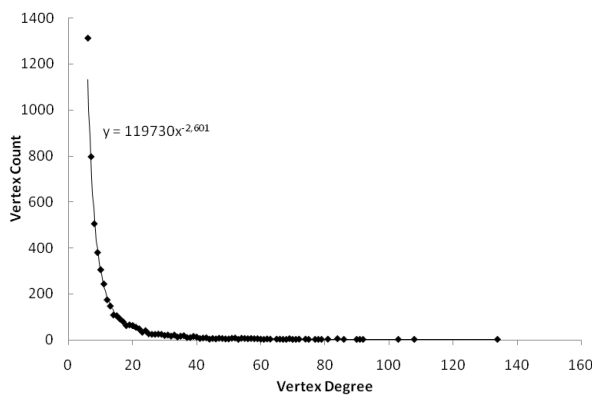


Figure 3: Vertex degree distribution and shortest path length distribution. The number of vertices in the structure is 10000.

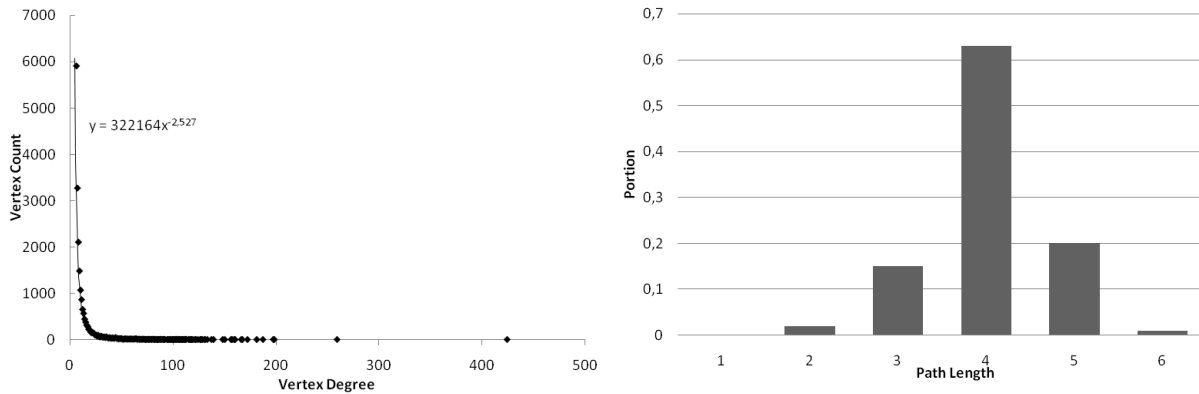


Figure 4. Vertex degree distribution and shortest path length distribution. The number of vertices in the structure is 20000.

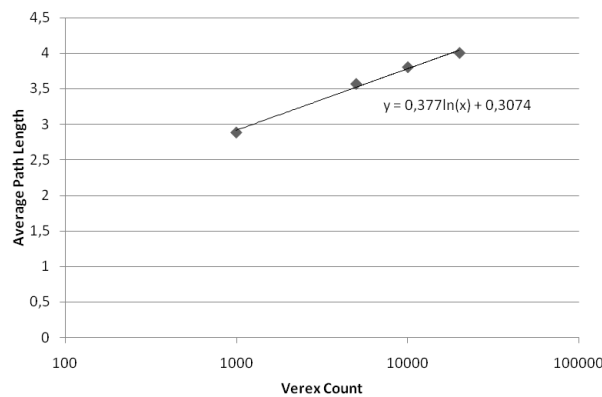


Figure 5. Average shortest path length vs. vertex count.

## 6 REFERENCES

- [1] C. Sartiani, P. Manghi, G. Ghelli and G. Conforti "XPeer: A Self-Organizing XML P2P Database System." Proc. P2P&DB, 2004.
- [2] G. Koloniari, E.Pituora "Peer-to-Peer Management of XML Data: Issues and Research Challenges." SIGMOD Record, Vol. 34, No. 2, June 2005.
- [3] V. Krylov, A. Logvinov, A. Ponomarenko, D.Ponomarev "Metrized Small World Properties Data Structure", SEDE 2008.
- [4] D.J. Watts "Small Worlds", Princeton, New Jersey: Princeton University Press, 1999.
- [5] 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.
- [6] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, S. Shenker, "A scalable content-addressable network." SIGCOMM 2001, pp. 161-172.
- [7] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications." SIGCOMM 2001, pp. 149-160.
- [8] R. Albert and A.-L. Barabasi "Statistical mechanics of complex networks." Rev. Mod. Phys., 74(1): pp. 47-97, January 2002.
- [9] F. C. W. Aiello and L. Lu "Random Evolution in massive graphs." In proc. of the 42<sup>nd</sup> IEEE symposium on Foundations of Computer Science, p 510. FOCS, IEEE Computer Society, 2001.
- [10] C. Schmitz "Self-organisation of a small world by topic." In Proc. Of the 1st Intl. Workshop on Peer-to-Peer Knowledge Management, pp. 61-66, 2001.