UDC 004.82:004.62=111

S.S. Zobnin, V.B. Polyakov

INTELLIGENT TIME SERIES STORING AND INFERENCE ENGINE IMPLEMENTATION WITH FOCUS ON PERFORMANCE AND HIGH LEVELS OF ABSTRACTION

This paper covers several aspects of intelligent time series database implementation. It also includes the description and analysis of a symbolic time series representation scheme. The paper focuses on various indexing and parallelization approaches in conjunction with actual backend storage engines. Special emphasis is made on identifying the problem of combining simple queries with time series pattern search and retrieval requests and finding a solution to this problem. The paper also considers query definition and provides the general architecture of a time series database with data mining capabilities.

TIME SERIES; DATABASE; DISTRIBUTED SYSTEM; INDEXING.

С.С. Зобнин, В.Б. Поляков

РЕАЛИЗАЦИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ СИСТЕМ ХРАНЕНИЯ И ОБРАБОТКИ ВРЕМЕННЫХ РЯДОВ, ПРЕДСТАВЛЕННЫХ В ВИДЕ ВЫСОКОУРОВНЕВЫХ АБСТРАКЦИЙ

Рассмотрены аспекты реализации высокопроизводительных систем хранения и обработки временных рядов, представленных в виде высокоуровневых абстракций. Особое внимание уделено производительности конечного решения и способам увеличения производительности. Проведено сравнение различных высокоуровневых представлений временных рядов по разным критериям. Изучены различные распределенные низкоуровневые хранилища данных. Установлена и рассмотрена проблема эффективного выполнения комбинированных запросов к хранилищу временных рядов. Рассмотрена задача создания запроса для рассматриваемой базы данных временных рядов. Приведена конечная архитектура рассматриваемой системы.

БАЗА ДАННЫХ ВРЕМЕННЫХ РЯДОВ; РАСПРЕДЕЛЕННАЯ СИСТЕМА; ИНДЕКСИРОВАНИЕ.

Nowadays big volumes of time series data are generated in various fields of science and engineering. Most of the time such data is just stored and forgotten (or analyzed using standard visualization / retrieval tools «by hand»), but it is very tempting to have an opportunity to extract certain amounts of information that might be useful for immediate and strategical problem solving and control tasks, such as events, patterns, motifs, anomalies etc, in an automated or even automatic manner. The implementation of such system (which may also be called as intelligent database) is always accompanied by various problems. The main part of this paper contains a set of such problems and ways of solving them, referenced by related work and several contributions of this paper's author.

The main contribution of this paper is the integration of various techniques (which originated in signal processing, data mining, big data and distributed systems, engineering fields) connected with efficient and production-ready intelligent time series database implementation into a single reusable framework. Several secondary contributions are related to existing algorithm combinations and optimization. A novel time series approximation scheme is proposed. The purpose of this paper is to provide a general method (framework) of solving a range of tasks related to intelligent time series management and mining, as well as a general overview of the problem with some concrete ideas and practices.

Time Series Data, Representation Scheme

First of all, data itself, or, to be specific, the amount of time series and data quality is a problem. Hundreds of gigabytes of time series data to be processed in such system require high performance of storage backends. The quality or (and) specifics in data streams is also an issue: intelligent similarity search algorithms tend to depend on internal parameters of time series and have problems with generalization. Usually one should consider at least problems of noise (SNR) and data misses and provide ways of intelligent algorithms parametrization.

The general intelligent algorithm that works with time series data usually uses some intermediate time series representation. Many time series representations have been introduced (e. g. FFT, Wavelets, Piecewise-Constant Approximation, curve-based approximation etc). Recently it has been shown that symbolic representations of time series outperform most of other representations in terms of approximation performance and execution performance. Moreover, they allow using some frameworks which require symbolic input (e. g. Markov models, Kolmogorov complexity based methods, suffix trees), give good compression rate and give a lower bound on the real value (useful in data mining applications), reduce sensitivity to noise and greatly improve computational performance [1]. There are two types of time series symbolization: the first one is based on quantization and the second is based on temporal segmentation. It was found that quantization-based symbolization performs generally better on signals without temporal structure (the criteria are: information loss, accuracy, alphabet size) and vice-versa [2]. Several extensions for basic representation scheme have been developed, such as: symbolization based on kNN segmentation [3] where a relative frequency is used to determine the best approximation parameters, symbolization based on *k*-Means [4] which highly outperforms the standard scheme on highly Gaussian distributed streams. One may also employ a more intelligent approach: in order to optimize the algorithm for a particular dataset, a neural network may be tested and used either for an efficient rank-based parameter pick as in [3], or for an efficient segmentation scheme as in [4]. There is no doubt that symbolic representation of time series is most appropriate for similarity search in intelligent database (most arising) and it is the best solution for fast retrieval due to availability of various data structures from bioinformatics and computer science communities.

During the investigation of SAX time series representation the following specifics were found. SAX is very dependent on parameters (window size, word size, alphabet size). The window size parameter is responsible for capturing the dynamics of time series (data is normalized in each window and as a consequence). The word size and alphabet size parameters determine performance/space requirements of the transformation and operations over transformed time series. The word size and alphabet size correspond to discretization and quantization. It comes true that determining optimal parameters of SAX is an important and actual problem. Moreover, domain knowledge should be integrated in order to organize the most efficient storage and retrieval scheme. An example of such domain knowledge integration is manually defined stream groupings provided by an expert, the other example is unsupervised classification of time series, which determines parameters. It should be noted that it is also not optimal to generate an infinite number of parameter triples, because this requires a lot of additional computations. The optimization task should be defined to perform an on-line unsupervised classification that also minimizes the number of class reallocation in time.

Indexing Symbolic Time Series. Simple Queries. Approximations and Storage. Parallel Execution

Indexing is the core of a database, because there always are response and performance requirements of a database in production systems. One always faces a space-time dilemma when some indexing scheme is implemented, and uses an underlying representation (symbolic in our case) for the best performance. The chosen indexing scheme directly depends on queries to be executed against a database. In case of intelligent time series database, the main query is a «pattern query» or a similarity

query. The other possible queries are event queries like «*X* more than 10 and *Y* less than 20» and more simple ones.

Here two combinations of indexing schemes are proposed. Each of these combinations allows fulfilling all query requirements.

Let's consider the first combination. In order to allow fast «direct» pattern match against symbols, a suffix tree is built over symbolic representations of time series. This can be called the first level of index. The second level of index is an index for similarity-based queries. A KD-tree is used for this purpose. The same KD-tree is used to execute event queries. Performance is improved via using the following methods. At first, one probabilistic suffix tree is used to perform faster first level index operations. Secondly, KD-tree performs partial queries and uses heuristics. Last, but not the least, a separate «reduces space» index could be built in the reduced feature space of time series, where the reduction is done via domain knowledge or general PCA or SOM scheme. These approaches were evaluated by the author of this paper. Alternative approaches to KD-tree index are other structures optimized for similarity search: general B-tree based structures [5], R-trees [6]. It should be noted, that the second index level techniques listed here are basic ones: trees need special measures to handle balancing issues, more parallelization and distribution considerations should be made. Even though a tree index can be equally distributed, processing time for each node needs prediction, which itself represents a forecasting problem that is not easily solved; nodes distribution mechanisms with random effects also have some difficulties.

The second combination of methods and techniques that allow fulfilling advanced querying requirements as well as performance requirements gives more focus on distributional and fault-tolerance capabilities of the system. Instead of using a suffix tree as a primary indexing structure, one can use a distributed hashing structure. There has been large research on distributed hashing structures recently, especially Distributed Hash Tables (DHT): CAN [7], Chord [8] and others. Generally, the problem of indexing time series was thoroughly studied in [9].

The advantage of the suffix tree as a first level indexing structure is its simplicity and general ability to perform suffix (substring) queries. DHT-like structures require much more additional space to allow such expressivity. On the other hand, DHT-based structures are purely distributed, thus naturally allowing distributed computations and storing.

A storage and cluster computation mechanism is required in order to fulfill strict requirements of productions environment. Although there is some research into standard backends usage for time series mining tasks [10], such DBMSs lack replication and fault-tolerance due to highly structured data predisposition. The integration of intelligent capabilities requires modifying the source code like in [11], which is error prone and not appropriate in many cases. A modern distributed computation framework is much more preferable, given the fact that parallelism can always be exploited and is implemented naturally in such frameworks. One more requirement is the ability of manual RAM caching of computations in such framework, the indexes are more preferable in RAM rather than on hard drive. As a result, considerable systems are [12–14].

It should be noted, that such distributed systems are executed on commodity hardware (i. e. standard hardware with no special requirements), comparing to specialized massively parallel solutions, for example [15] meaning that they are more universal and have bigger latencies comparing to specialized hardware/software massively parallel solutions at the same time. One implication is that the time series database system implemented on top of commodity hardware is not capable of handling hard real time tasks arising, for example, in military domains.

The General Architecture of an Intelligent System

The general architecture of an intelligent time series database can be represented in Fig. 1.

The central component is the intelligent database itself, where indexes are stored in memory, while raw data is stored in symbolic form on the disk. This may require a cluster of machines due to large size of indexes.

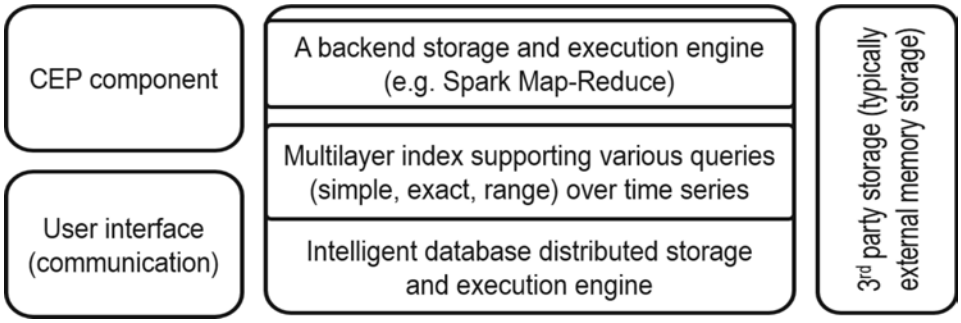

Fig. 1. The overview of architecture of an intelligent time series database

The system is highly coupled with some 3rd-party storage, because raw data is always stored somewhere. A good architectural solution is to create an «Adapter» for each 3rd party, the database is working with, so that the intelligent database itself and adapters are highly reusable.

The CEP component of the system is responsible for preprocessing data for detection of some event types. This component executes event processing queries against data, a good solution is to reuse some CEP engine while building indexes and use some declarative language to find events rather than to encode requests by hand. CEP component also provides additional stream processing capabilities for on-line analysis, i. e. stream manipulation, aggregation, joining etc.

It is very important to have a subsystem responsible for user communication so that an operator would have a way to present patterns for queries in convenient forms. An example of such assistance is a visual pattern constructor, where it is easy to define the shape of a curve (pattern) to be found in the database.

The concept of multilevel indexing is represented in Fig. 2.

As discussed in the previous session, the most optimal storing and indexing scheme for time series is the scheme containing both computer science techniques and bioinformatics techniques. This allows fulfilling performance requirements as well as provides capabilities to perform simple queries, exact time series match queries, range queries, N-nearest time series queries and others. Not only does such scheme allow such querying capabilities, but it also allows using distributed systems and networks concepts and experience: standard DHT or CAN implementations may be used as the second level of index.

It is convenient to have a distributed optimization component in common space to allow its reusability: the need of solving optimization tasks arises on multiple levels of the system. Currently a version of distributed

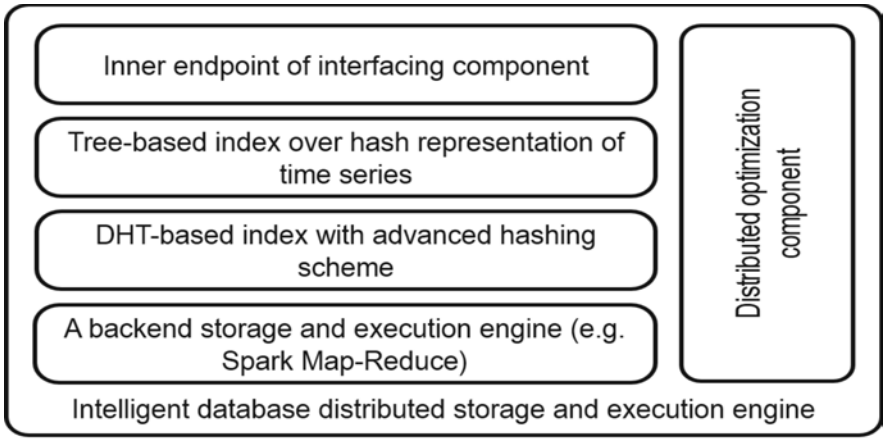

Fig. 2. The overview of architecture of an intelligent time series database

simulated annealing is used as an optimization framework. A special protocol is implemented for optimization of agents' communication. This protocol allows asynchronous optimization, incremental updates of «current optimum» and dynamic task reallocation.

The following figure (Fig. 3) shows in detail the process of parsing incoming and generating a distributed index for later querying as well as the process of query generation, processing and execution, one can also consider this figure as a general algorithm, i. e. how the system works.

A system constantly accepts incoming data from various origins. CEP/Stream processing (as well as any simple processing like stream multiplication) queries are constantly executed on incoming data in order to receive alternated streams.

These data streams arrive to the module responsible for generating symbolic representation. This very module contains additional domain knowledge, possibly automatically inferred from data, that allows more efficient storage and representation. For example, there can be additional logic for automatic unsupervised classification of incoming data streams. A simpler approach is to integrate domain knowledge of an expert in various forms. For example, an expert can set up proper groups of similar signals from all. Different parameters based on domain knowledge are then used for symbolic representation generation algorithm.

The output of the module described in the previous paragraph goes to a specially designed hash function that gives hash values to time series dividing them into equivalence groups (dimensionality reduction). Generated hash values are then mapped into DHT / CAN nodes and distributed around the network. This process is executed continuously because amounts of incoming data can be too large. The other continuously running indexing process is the process that performs indexing of hashes into tree structures. These structures are also ideally distributed as it was discussed in the previous chapters.

User defined queries, possibly from a special graphical interface for query construction, as well as constraints on window size and additional parameters are supplied to the other side of the system. This data is then translated into patterns suitable for search, a search query is executed and some post processing of search results are performed.

At least two general ways of the described system integration can be defined. The first one puts time series database in the front line (data acquisition layer). The second way of integration considers time series database as an

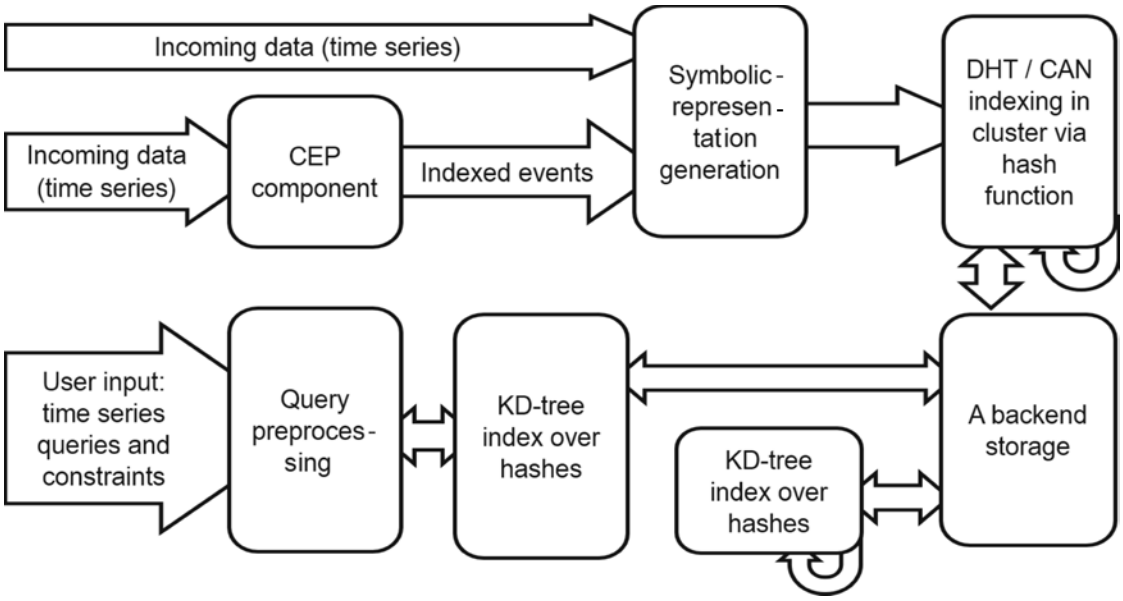

Fig. 3. Internal data flows

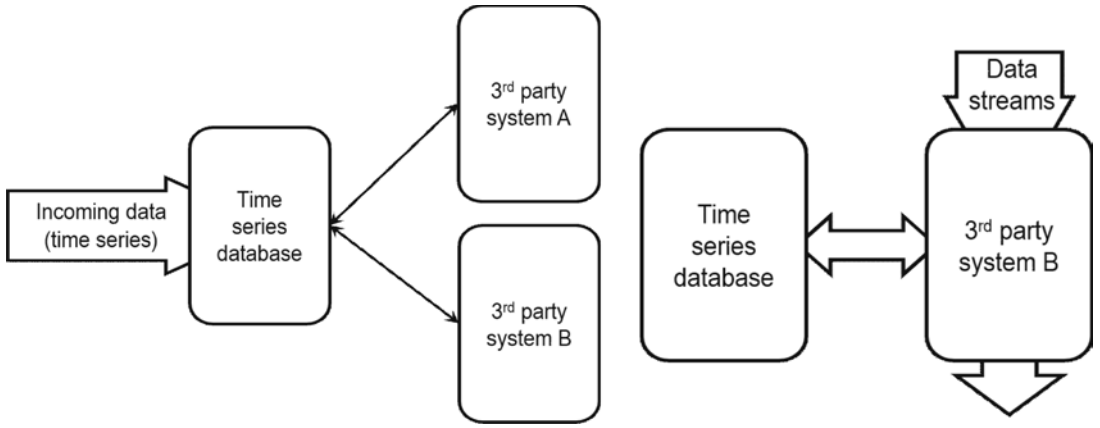

Fig. 4. Ways of integrating time series database with other systems: left, where time series database is considered as a part of data acquisition layer and right, where considered system is used as an auxiliary system for other products

auxiliary part of another bigger system. Both these cases are represented in Fig. 4.

The left case (Fig. 4, left) inserts time series database into the system circuit, thus reducing total system fault-tolerance. On the other side, in such way time series database receives on-line data and has this data indexed allowing other systems to work with this data at once. Thus, such way is the best for 3rd party systems that require intelligent time series manipulations with low latency. It is also possible to stream data in parallel to third party components having all pros at the same moment.

The right case (Fig. 4, right) puts time series database aside from real time series. In this way an intelligent database does not have data in real time, but it is not located in the main data circuit. Such way of integration is more suitable for cases when off-line processing is required for some existent systems as a separate module, and extra low latency is not required.

The approach provided in this paper is applicable to a wide range of tasks related to time series: analysis, efficient storage, approximation, motif discovery, classification, and others. For example, in order to implement a fully functional prototype of the vibration diagnostic system based on models from [16], it is required to have a subsystem able to provide both raw time series functionality and similarity search functionality. The system based on the concept from [17] also requires historical access and advanced time series similarity search capabilities.

It was shown that the chosen methods (symbolic representation + multilevel indexes with approximations at distributed storage, queries are executed in cluster in parallel) greatly outperform standard approaches (standard time series representations, SQL databases) in terms of amount of queries per second and single query execution time.

The ongoing research is targeted at investigation and comparison of concrete algorithms and methods for each stage of the described systems, integration of chosen algorithms, improving them and investigating the means of domain knowledge integration.

The research and implementation of the system based on the concepts from this paper is performed as a part of the project related to time series analysis and visualization in Siemens LLC.

REFERENCES

1. **Lin J., Wei Li et al.** Experiencing SAX: a Novel Symbolic Representation of Time Series, 2007.

2. **Sant'Anna A., Wickstrom N.** Symbolization of time-series: An evaluation of SAX, Persist, and ACA. *Image and Signal Processing, 4th Internat. Congress on*, 2011, Vol. 4, pp. 2223–2228.

3. **Ahmed A.M., Bakar A.A., Hamdan A.R.** Improved SAX time series data representation based

on Relative Frequency and K-Nearest Neighbor Algorithm. *Intelligent Systems Design and Applications, 10th Internat. Conference on*, 2010, pp. 1320–1325.

4. **Pham N.D., Quang Loc Le, Tran Khanh Dang.** Two Novel Adaptive Symbolic Representations for Similarity Search in Time Series Databases. *12th Internat. Asia-Pacific Web Conference APWEB-2010.*

5. **Cui Yu, Bin Cui, Shuguang Wang, Jianwen Su.** Efficient index-based KNN join processing for high-dimensional data. *Inf. Softw. Technol.,* 2007, Vol. 49, No. 4, pp. 332–344.

6. **Kuan J., Lewis P.** Fast k nearest neighbour search for R-tree family. *Information, Communications and Signal Processing, Proceedings of Internat. Conference ICICS-1997*, Vol. 2, pp. 924–928.

7. **Ratnasamy S., Francis P., Handley M., Karp R., Shenker S**. A scalable content-addressable network. *SIGCOMM Comput. Commun.,* 2001, Rev. 31, No. 4, pp. 161–172.

8. **Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H.** Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun,* 2001, Rev. 31, No. 4, pp. 149–160.

9. **Mar Yi Yi, Maw Aung Htein.** Tree-based Indexing for DHT-based P2P Systems. *Internat. Journal of Computer Applications*, 2013, Vol. 62, pp. 27–33.

10. **Sorokin A., Selkov G., Goryanin I.** A user-defined data type for the storage of time series data allowing efficient similarity screening. *European Journal of Pharmaceutical Sciences,* 2012, Vol. 46, Iss. 4, pp. 272–274.

11. **Bartunov O., Sigaev T.** Efficient K-nearest neighbour search in PostgreSQL. *Conference talk, PGDay-2010*, Roma.

12. **Dean J., Ghemawat S.** MapReduce: simplified data processing on large clusters. *Commun. ACM,* 2008, Vol. 51, No. 1, pp. 107–113.

13. **Zaharia M., Chowdhury M., Franklin M.J., Shenker S., Stoica I.** Spark: cluster computing with working sets. *In Proceedings of the 2nd USENIX Conference on Hot topics in cloud computing*, USENIX Association, Berkeley, CA, USA, p. 10.

14. GridGain systems GridGain. Available: http://www.gridgain.com

15. **Polyakov V.B.** Organizacija vychislenij v arhitekturah, podderzhivajushhih massovyj parallelism. *Voprosy radioelektroniki.* Moscow: Elektronica Publ., 2008, pp. 33–41. (rus)

16. **Zinkovsky A.V., Potekhin V.V., Ilyin I.Y.** Modelling of vibration activity on a man locomotorium. *Proceedings of the 10th Internat. Congress on Sound and Vibration*, 2003, pp. 3851–3856.

17. **Arsen'ev D.G., Shkodyrev V.P.** Strategija gruppovogo upravlenija robotami na osnove situacionno-celevogo planirovanija. *Izvestija Juzhnogo federalnogo universiteta. Tehnicheskie nauki.* Taganrog, 2010, pp.40–43 (rus)

СПИСОК ЛИТЕРАТУРЫ

1. **Lin Jessica, Wei Li, et al.** Experiencing SAX: a Novel Symbolic Representation of Time Series. 2007.

2. **Sant'Anna A., Wickstrom N.** Symbolization of time-series: An evaluation of SAX, Persist, and ACA // Image and Signal Processing, 4th Internat. Congress on. 2011. Vol. 4. Pp. 2223–2228.

3. **Ahmed A.M., Bakar A.A., Hamdan A.R.** Improved SAX time series data representation based on Relative Frequency and K-Nearest Neighbor Algorithm // Intelligent Systems Design and Applications, 10th Internat. Conf. on. 2010. Pp. 1320–1325.

4. **Pham N.D., Quang Loc Le, Tran Khanh Dang.** Two Novel Adaptive Symbolic Representations for Similarity Search in Time Series Databases // 12th Internat. Asia-Pacific Web Conference. 2010.

5. **Cui Yu, Bin Cui, Shuguang Wang, Jianwen Su.** Efficient index-based KNN join processing for high-dimensional data // Inf. Softw. Technol. 2007. Vol. 49. no. 4. Pp. 332–344.

6. **Kuan J., Lewis P.** Fast k nearest neighbour search for R-tree family // Information, Communications and Signal Processing. Proc. of Internat. Conf. ICICS. 1997. Vol. 2. Pp. 924–928.

7. **Ratnasamy S., Francis P., Handley M., Karp R., Shenker S.** A scalable content-addressable network // SIGCOMM Comput. Commun. 2001. Rev. 31. no. 4. Pp. 161–172.

8. **Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H.** Chord: A scalable peer-to-peer lookup service for internet applications // SIGCOMM Comput. Commun. 2001. Rev. 31. no. 4. Pp. 149–160.

9. **Mar Yi Yi, Maw Aung Htein.** Tree-based Indexing for DHT-based P2P Systems // Internat. J. of Computer Applications. 2013. Vol. 62. Pp. 27–33.

10. **Sorokin A., Selkov G., Goryanin I.** A user-defined data type for the storage of time series data allowing efficient similarity screening // European J. of Pharmaceutical Sciences. 2012. Vol. 46. Iss. 4. Pp. 272–274.

11. **Bartunov O., Sigaev T.** Efficient K-nearest neighbour search in PostgreSQL // Conf. talk, PGDay-2010, Roma, 2010.

12. **Dean J., Ghemawat S.** MapReduce: simplified

data processing on large clusters // Commun. ACM. 2008. Vol. 51. no. 1. Pp. 107–113.

13. **Zaharia M., Chowdhury M., Franklin M.J., Shenker S., Stoica I.** Spark: cluster computing with working sets // In Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing. USENIX Association, Berkeley, CA, USA. P. 10.

14. GridGain systems GridGain [электронный ресурс] / URL: http://www.gridgain.com

15. **Поляков В.Б.** Организация вычислений в архитектурах, поддерживающих массовый параллелизм // Вопросы радиоэлектроники. М.: Электроника, 2008. С. 33–41.

16. **Zinkovsky A.V., Potekhin V.V., Ilyin I.Y.** Modelling of vibration activity on a man locomotorium // Proc. of the 10th Internat. Congress on Sound and Vibration. 2003. Pp. 3851–3856.

17. **Арсеньев Д.Г., Шкодырев В.П.** Стратегия группового управления роботами на основе ситуационно-целевого планирования // Известия Южного федерального университета. Технические науки. Таганрог, 2010. С. 40–43.

ZOBNIN, Sergey S. *St. Petersburg State Polytechnical University. Siemens LLC.*
195251, Politekhnicheskaya Str. 21, St. Petersburg, Russia.
E-mail: sergey.s.zobnin@gmail.com

ЗОБНИН Сергей Сергеевич – *студент кафедры систем и технологий управления Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 21.
E-mail: sergey.s.zobnin@gmail.com

POLYAKOV, Vadim B. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 21, St. Petersburg, Russia.
E-mail: vadim7702@yandex.ru

ПОЛЯКОВ Вадим Борисович – *доцент кафедры систем и технологий управления Санкт-Петербургского государственного политехнического университета, доктор технических наук.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 21.
E-mail: vadim7702@yandex.ru