UDC 004.7
doi:10.31799/1684-8853-2020-5-43-49

Articles

Storage scaling management model

***Sovetov B. Ya.*[a]**, *Dr. Sc., Tech., Professor, orcid.org/0000-0003-3116-8810,*
***Tatarnikova T. M.*[b]**, *Dr. Sc., Tech., Associate Professor, orcid.org/0000-0002-6419-0072, tm-tatarn@yandex.ru*
***Poymanova E. D.*[b]**, *Senior Lecturer, orcid.org/0000-0002-7903-2480,*
[a]*Saint-Petersburg Electrotechnical University «LETI», 5, Prof. Popov St., 197376, Saint-Petersburg, Russian Federation*
[b]*Saint-Petersburg State University of Aerospace Instrumentation, 67, B. Morskaia St., 190000, Saint-Petersburg, Russian Federation*

Introduction: *The implementation of data storage process requires timely scaling of the infrastructure to accommodate the data received for storage. Given the rapid accumulation of data, new models of storage capacity management are needed, which should take into account the hierarchical structure of the data storage, various requirements for file storage and restrictions on the storage media size.* ***Purpose:*** *To propose a model for timely scaling of the storage infrastructure based on predictive estimates of the moment when the data storage media is fully filled.* ***Results:*** *A model of storage capacity management is presented, based on the analysis of storage system state patterns. A pattern is a matrix each cell of which reflects the filling state of the storage medium at an appropriate level in the hierarchical structure of the storage system. A matrix cell is characterized by the real, limit, and maximum values of its carrier capacity. To solve the scaling problem for a data storage system means to predict the moments when the limit capacity and maximum capacity of the data carrier are reached. The difference between the predictive estimates is the time which the administrator has to connect extra media. It is proposed to calculate the values of the predictive estimates programmatically, using machine learning methods. It is shown that when making a short-term prediction, machine learning methods have lower accuracy than ARIMA, an integrated model of autoregression and moving average. However, when making a long-term forecast, machine learning methods provide results commensurate with those from ARIMA.* ***Practical relevance:*** *The proposed model is necessary for timely allocation of storage capacity for incoming data. The implementation of this model at the storage input allows you to automate the process of connecting media, which helps prevent the loss of data entering the system.*

Keywords *— data storage system, storage media, multi-level data storage, databank, databank scaling, databank control, required storage capacity forecast.*

For citation: Sovetov B. Ya., Tatarnikova T. M., Poymanova E. D. Storage scaling management model. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2020, no. 5, pp. 43–49. doi:10.31799/1684-8853-2020-5-43-49

Introduction

Data storage is a service needed by companies, governmental structures or individuals. Banks and insurance companies are active users of this service, as they keep all the primary documentation in scanned format. National security structures also have a high demand in data storage [1]. Urban infrastructure projects such as “Smart yard”, “Safe city”, etc. assume real-time data storage with subsequent archiving. The storage of unstructured medical data is often regulated by law. Individual users also contribute to the rapid growth of the data amount, creating private “clouds” in order to store their personal content.

The tendency of rapid data accumulation is referred to as “big data”. It is often characterized by its “three Vs”:

Volume, i. e. physical amount of data;

Velocity, i. e. speed of the data growth;

Variety, i. e. the capability to process different data types in the same time.

Big data is a trend in the modern stage of our transition to a digital society. To organize data storage, you need special infrastructure [2]. It can be provided by a data storage system (DSS) which is a software/hardware solution for recording the data to be stored, secure storage of this data, and reading it by user’s demand [3]. The total capacity of a DSS, without going into details about the software/hardware solution, is also called data storage [4, 5].

A data storage is a complex control object which must deal with two types of load: data coming in real time, and data whose secure storage requires timely procedures of archiving, backuping, etc. [6–8]. Thus, the problem of big data is not the lack of the storage space but the lack of adequate models for controlling this space, in particular, its timely extension [9, 10]. Currently, the solution for this problem is not automated, being a part of the DSS operator’s duties. Implementing such a model at the input of a storage would allow us to automate the process of linking up the storage media, making sure the input data is not lost [11].

The results presented in this article are a continuation of the work [12]. The model proposed in [12] allows you to make short-range forecasts about events that require your DSS to be scaled. However, data storage systems need mid- and long-range forecasts, as the implementation of scaling takes time, including the time for purchasing, installing and setting the equipment.

Description of the research object and statement of the problem

A data storage system follows the principle of hierarchical storage. Nowadays, at least three levels are specified in the DSS architecture [13–15]:

– RAID (Redundant Array of Independent Disks);

– automated libraries;

– long-term storage media.

Increasing the number of the levels is the problem of vertical scaling. Let us denote the number of levels in a DSS as m.

Increasing the number of the volumes/carriers at the i^{th} level of a DSS is the problem of timely horizontal scaling. Let us denote the number of volumes/carriers in a DSS as n [16].

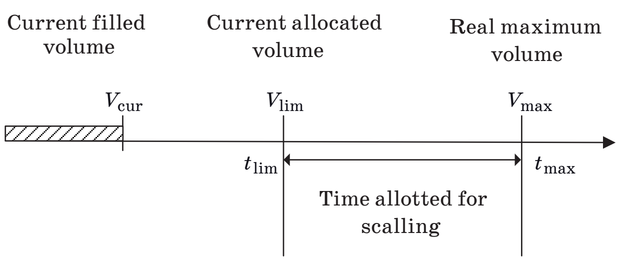
The state of a DSS can be formalized as a matrix **B** of size $m \times n$ [17], whose elements are sets of files with assigned parameters $F = \{t, f, \lambda\}$, where t is the time of guaranteed storage determined from the file extension **.type*; f is the file size; λ is the frequency of requests for the file. Parameters t and f are metadata analyzed at the DSS input. Parameter λ is dynamic; it determines the migration of files over the DSS levels.

Each cell of the matrix **B**, in its turn, can be represented by its own characteristics (Fig. 1).

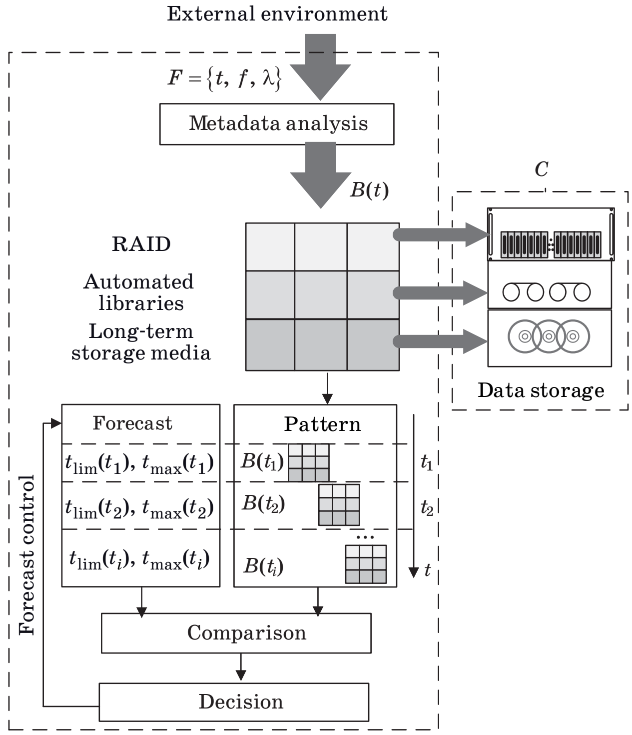
Thus, the problem of DSS scaling formulated in [12] as a forecast problem (estimating the moment t_{lim} of an event at which V_{cur} tends to V_{lim} and estimating the moment $t_{\max}$ of an event at which V_{cur} tends to $V_{\max}$) is still the same. But our goal is getting mid- and long-range forecasts.

Let us call the DSS state specified by the matrix **B** at the time moment t a behavior pattern $B(t)$ [18, 19].

Periodic analysis of $B(t)$ patterns allows you to adjust the forecast estimations t_{lim} and $t_{\max}$. The structure of the data storage scaling management model is given in Fig. 2.



■ ***Fig. 1.*** Characteristics of a cell of the matrix **B**



■ ***Fig. 2.*** Data storage scaling management model

The forecast model is adjusted automatically if the deviation of the pattern parameters from the forecast estimations becomes significant.

Proposed solution

In order to estimate the values of t_{lim} and $t_{\max}$, let us use the following machine learning methods:

– Decision trees;

– Random forest;

– Feedforward neural network;

– Support vector machines.

Decision trees are the most popular method of numerical forecasting in the case when the forecast variable values are continuous. The method is popular due to the tree structure, in which the decision-making sequence is reduced to a number of vertex transitions, which subsequently makes model very comprehensible. The major shortcoming of the method is that it often requires retraining, which can lead to branchy trees and/or high class bias.

Formally, a decision tree is a graph $G = (V, E)$, where V is a finite non-empty set of vertexes, i. e. tree leaves; E is a finite non-empty set of vertex pairs, i. e. tree branches. The branches contain various conditions, and the leaves contain their values. The main parameter of the method is the tree depth, i. e. the distance from the root to the most remote leaves. The model of a decision tree is built following two algorithms: induction and pruning. Induction specifies the borders of a hierarchical solution, based on the feature space x. Pruning removes less informative structures from the tree, preventing retraining. The conditions for splitting tree vertices are formed with a feature chosen by a greedy algorithm in order to reduce the cost function. In numerical prediction, this function is mean squared error (MSE):

$$MSE = \sum_{i=1}^{N}\left(V_i - \hat{V}_i\right)^2, \qquad (1)$$

where V_i is real data in the i^{th} point; $\hat{V}_i$ is a predicted value in the i^{th} point; N is the sample size.

The decision tree G breaks all the feature space into a certain number of non-overlapping subsets $\{E_1, ..., E_M\}$, and in each subset E_j produces a prediction $\hat{V}_i$

$$G(V, E) = \sum_{j=1}^{N}\hat{V}_j\left[E_j \in E\right]. \qquad (2)$$

In the prediction problem to be solved, the tree vertices are traffic amounts, and the branches are the moments of time when the incoming traffic amount is fixed.

Random forest is an ensemble of trees in which each tree has its weight when obtaining the final prediction estimation. The randomness is introduced into the ensemble of trees at the induction stage. The most common way of building an ensemble of trees is called bagging (abbreviated *bootstrap aggregation*). It is based on artificial creation of several samples $V = \{V_1, V_2, ..., V_C\}$ from the training set, uniformly and with replacement. On each sample, a decision tree $\mathbf{G}_i$, $i = \overline{1, C}$ is built. Bagging is normally used in order to avoid retraining, i. e. saving the training dataset indiscriminately. The algorithm is characterized by two main parameters: number of the trees and their depth.

In addition to bagging, you can randomly select a subset of features in each vertex in order to make the trees more independent. The final prediction is chosen by averaging:

$$G(V, E) = \frac{1}{C}\sum_{i=1}^{C}G_i. \qquad (3)$$

Using *neural networks* for time series prediction is based on the assumption that a regression problem can be replaced by a recognition problem. As a matter of fact, a neural network cannot predict. It can only recognize, in the current parameters of a time series, a familiar situation, reproducing a reaction to it as precisely as possible. Thus, the statement of a prediction problem as applied to a neural network can be formulated as follows: to find the best approximation to a function defined by a finite set of a training sample.

In time series prediction, a multilayer perceptron, i. e. a feedforward neural network can be useful, as a feedback network uses short-term memory.

The inputs of a multilayer perceptron are fed with parameters reflecting the function of the data stream $\phi(t)$ to the DSS, and the outputs are the predicted values of t_{lim} and t_{max}. This method was dubbed "sliding windows", as its implementation requires two windows: the input window corresponds to the values of the input layer of the neural network; the output window corresponds to the expected values of its output layer. In the course of a prediction, both these windows move along the function $\phi(t)$ with a step which corresponds to the time series interval. Like in all regression problems solved by neural networks, supervised training method is used.

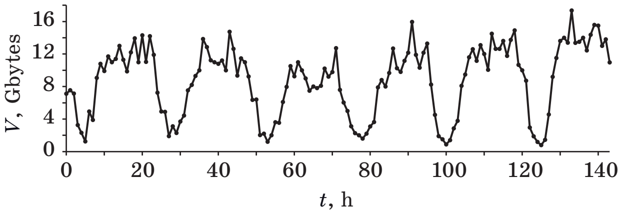
Support vector machines solve the prediction problem through classification, taking the initial data vector to an n-dimensional space and searching for $(n - 1)$ dimensional hyperplane with a maximum Euclidean distance between the borders of areas separating one class from another. The initial data vector is a time series describing the amount of traffic coming to the storage to be stored. In fact, support vector machines solve an optimization problem where the target function is maximizing the Euclidean distance between the decision planes or minimizing the average value of the MSE classifier. During the training, "volume-time" classes of the function $\phi(t)$ are determined; and during the prediction, the classes are revealed to which the future values of $\phi(t)$ can be assigned, and the values $t_{\text{lim}}(t)$ and $t_{\text{max}}(t)$ are calculated.

Linear optimization assumes that a system of equations is written as a scalar product; non-linear optimization assumes that it is written as a non-linear kernel function. In this work, Gaussian radial basis function is chosen.

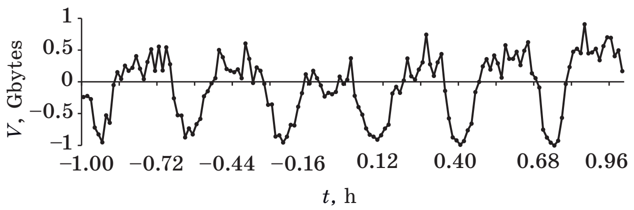
Discussion of the results

To estimate the accuracy of a short-term prediction, we used LTE traffic data for 6 days (August 20–25, 2018) provided by MTS mobile operator in Saint-Petersburg, Russia (Fig. 3). The time series consists of 144 observations averaged by 1440 values of the service amount V.

Since most machine learning methods are sensitive to scaling, the data should be pre-processed, using the procedures of data normalization if the nominal features can be put into the range from 0 to 1, or standardization if each feature has a mean value equal to 0 and dispersion equal to 1. In Fig. 4, you can see standardized LTE traffic from Fig. 3.



■ ***Fig. 3.*** Incoming stream of LTE traffic data by MTS [12]



■ ***Fig. 4.*** Standardized LTE traffic by MTS

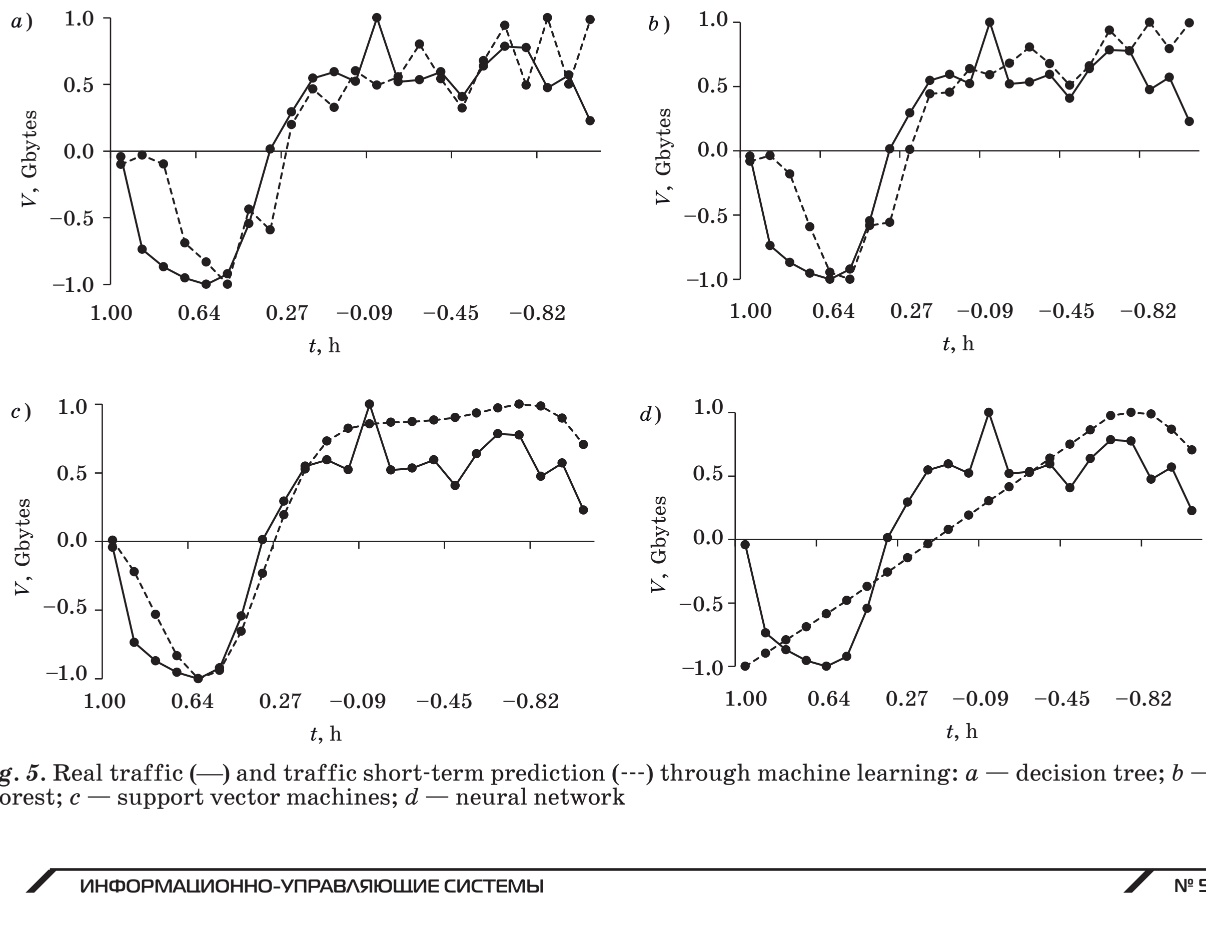
The short-term prediction was built for August 25, 2018. Fig. 5, *a–d* contains graphs demonstrating the difference between the real traffic and the predicted one, obtained through machine learning. Table 1 contains the values of MSE from which you can judge the prediction error, along with the key parameters selected during the training.

In [12] it was shown to be possible to obtain MSE = 0.04 by using ARIMA, an integrated model of autoregression and moving average. Fig. 6 shows graphs demonstrating the difference between the real traffic and predicted one obtained through ARIMA. The results do not contradict the statements that autoregression models are currently the best tool for time series prediction.

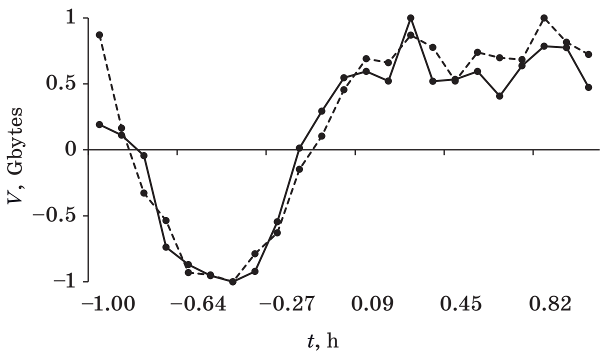
For obtaining medium- and long-term predictions, we need long records of traffic traces [20]. This

■ ***Table 1.*** MSE values and key parameters of machine learning

Machine learning method	MSE	Key parameters
Decision tree	0.54	Depth of the tree is 15
Random forest	0.525	Bagging, and randomly generated subsets from the training set with repetitions
Support vector machines	0.41	Gaussian kernel with radial basis function
Neural network	0.62	Architecture (A) is a multilayer perceptron with 3 hidden layers. The number of the hidden layers h is chosen experimentally: $\lim\limits_{0<h\le 9} A(h) \to \min(MSE)$



■ ***Fig. 5.*** Real traffic (—) and traffic short-term prediction (---) through machine learning: *a* — decision tree; *b* — random forest; *c* — support vector machines; *d* — neural network



■ ***Fig. 6***. Real traffic (—) and traffic prediction (---) through ARIMA model

seemingly unsatisfiable demand was satisfied due to the open publication of internet traffic data by a Japanese MAWI research group, WIDE project. The training set has 24-hour traces for September 05, 2018 and September 04, 2019, 48-hour traces for September 01, 2007–November 01, 2007, 72-hour routes for March 18–20, 2008, and 96-hour traces for March 30, 2009 and February 01–04, 2009.

Figure 7, *a–d* contains graphs demonstrating the difference between the real traffic and predicted one obtained through machine learning.

The values of MSE in ARIMA model, as applied to long records of traffic traces with medium- and long-term predictions do not significantly change, being equal to 0.035.

Table 2 shows the predicted values of t_{lim}, t_{max} for the storage at RAID0, RAID3 and RAID5 levels with the following parameters: capacity of one 2.5″ SSD is 3840 Gbytes; number of the discs is 50; total capacity is 192 Tbytes; time of switching between the discs at RAID3 level is 1 minute for taking into account the difference in performance between RAID3 and RAID5; time given by the administrator for scaling is 9 hours.

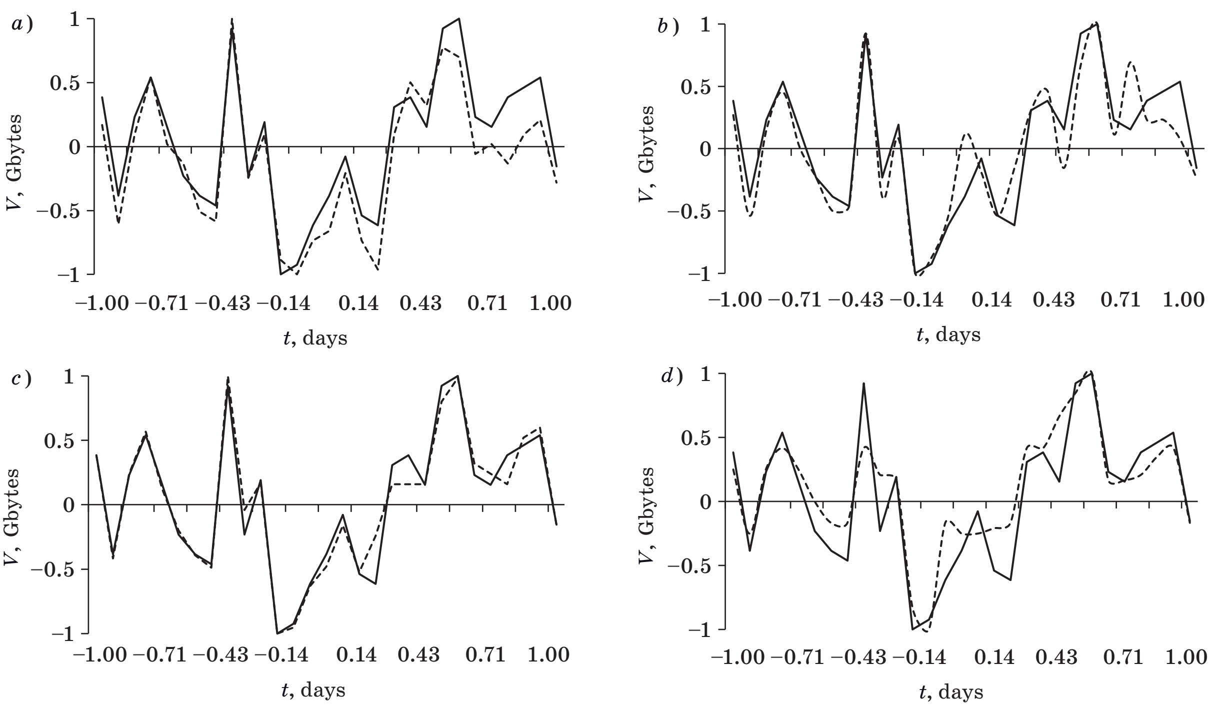
After applying different machine learning methods to the analysis and prediction of filling up the storage, we can make the following conclusions:

1. In spite of the fact that ARIMA model shows good results when its parameters are properly set up, it should be noted that the parameters are chosen through auto-correlation analysis which can take an order of magnitude more time as compared to machine learning.

2. There is no commonly accepted model for the prediction of filling up a data-storage system in or-

■ ***Table 2***. Predicted values of the onset of time of the maximum and real storage capacity

RAID level	Capacity, Tbytes	Efficient capacity, Tbytes	t_{lim}, days	t_{max}, days
RAID0	192	192	14.27	14.57
RAID3	192	188.16	14.06	14.36
RAID5	192	188.16	13.98	14.28



■ ***Fig. 7***. Real traffic (—) and traffic long-term prediction (---) through machine learning: *a* — decision tree, MSE = 0.047; *b* — random forest, MSE = 0.047; *c* — support vector machines, MSE = 0.013; *d* — neural network, MSE = 0.056

der to make a decision about timely scaling of the system. However, based on the obtained results, we can recommend the following:

— ARIMA model for short-term predictions;

— machine learning models for long-tern predictions.

3. Although any prediction is an extrapolation of data, machine learning methods actually solve an interpolation problem. This allows us to restore the amounts of data written in the storage which are not fixed in time.

Conclusion

The article discusses the importance of timely scaling of a data storage system. A model of horizontal scaling is proposed which presumes that the number of volumes/carriers at any hierarchical storage level is differentially increased.

Planning the scaling of storage capacity is based on predicting the amount of the incoming data traffic and the moment when the limit or maximum capacity of the storage medium is reached.

We have discussed various methods of machine learning applied to the analysis and long-term prediction of filling up a data storage system. The method of support vector machines provided predictive estimates three times higher by MSE value than those obtained through ARIMA which is currently considered the best model for time series prediction.

The prediction of storage capacity scaling is necessary for timely allotment of memory resources and reduction of the incoming traffic loss.

References

1. **Yakhina I.** Warehouses for big data. *Otkrytye sistemy* [Open systems], 2012, no. 7. Available at: https://www.osp.ru/os/2012/07/13017639 (accessed 31 August 2020).
2. **Proskuryakov N. E., Anufrieva A. Yu.** Analysis and prospects of modern digital data storage systems. *Proc. of the TSU*, 2013, no. 3, pp. 368–377 (In Russian).
3. **Vogel A., Griebler D., Maron C., Schepke C, Fernandes L.** Private IaaS clouds: a comparative analysis of OpenNebula, CloudStack and OpenStack. *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, IEEE 2016, pp. 672–679.
4. **Gangadharan G. R.** Open source solutions for cloud computing. *Computer*, 2017, no. 1, pp. 66–70.
5. *Information Storage and Management*. 2nd Edition. New Jersey, John Wiley & Sons, 2016. 544 p.
6. **Song Z., Zhang X., Eriksson C.** Data center energy and cost saving evaluation. *Energy Procedia*, 2015, vol. 75, pp. 1255–1260.
7. **Farley M.** *Building Storage Networks*. Osborne, McGrawHall, 2001. 576 p.
8. Recommendation Y.3510: Cloud computing infrastructure requirements. Geneva, ITU-T, 2013. Available at: https:// www.itu.int/rec/T-REC-Y.3510-201305-S (accessed 29 January 2019).
9. **Leonov V.** *Google Docs, Windows Live i drugie oblachnye tekhnologii* [Google Docs, Windows Live and other cloud technologies]. Eksmo Publ., 2012. 304 p. (In Russian).
10. **Carr N.** *The Big Switch — our New Digital Destiny*. WW Norton & Company, 2008. 278 p.
11. **Burmistrov V. D., Zakovryashin E. M.** Creating a data warehouse for a distributed system. *Young Scientist*, 2016, no. 12, pp. 143–147 (In Russian).
12. **Poymanova E. D.** *Model' upravleniya resursami sistem khraneniya dannykh*. Dis. kand. tehn. nauk [Storage management system resource management model. PhD tech. sci. diss.]. Saint-Petersburg, SPbGETU “LETI” Publ., 2020. 17 p. (In Russian).
13. **Buyya R., Broberg J., Goscinski A.** *Cloud Computing. Principles and Paradigms*. New Jersey, John Wiley & Sons, 2011. 637 p.
14. Recommendation Y.3501: Cloud computing framework and high-level requirements. Geneva, ITU-T, 2013. Available at: https://www.itu.int/ITU-T/recommendations/rec.aspx- ?rec = 11917 (accessed 29 January 2019).
15. **Zhmylev S. A., Martynchuk I. G.** Mathematical modeling in cloud systescaling. *IEEE Xplore Digital Library*, 2019, pp. 1–9. doi:10.1109 / WECONF.2019. 8840614
16. **Lorido-Botran Tania, Miguel-Alonso Jose, Lozano Jose A.** A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 2014, vol. 12, no. 4, pp. 559–592.
17. **Sovetov B. Ya., Tatarnikova T. M., Poymanova E. D.** Organization of multi-level data storage. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2019, no. 2, pp. 68–75 (In Russian). doi:10.31799/1684-8853-2019-2-68-75
18. **Morville P., Callender J.** *Search Patterns: Design for Discovery*. O'Reilly, 2010. 192 p.
19. **Stacey M., Salvatore J., Jorgensen A.** *Visual Intelligence: Microsoft Tools and Techniques for Visualizing Data*. New Jersey, John Wiley & Sons, 2013. 432 p.
20. **Shahzad A., Lee Y. S., Lee M., Kim Y-G., Xiong N.** Real-time cloud-based health tracking and monitoring system in designed boundary for cardiology patients. *Journal of Sensors*, 2018, vol. 2018, 15 p. doi:10.1155/2018/3202787

УДК 004.7
doi:10.31799/1684-8853-2020-5-43-49

Модель управления масштабированием системы хранения данных

Б. Я. Советов[а], доктор техн. наук, профессор, orcid.org/0000-0003-3116-8810
Т. М. Татарникова[б], доктор техн. наук, доцент, orcid.org/0000-0002-6419-0072, tm-tatarn@yandex.ru
Е. Д. Пойманова[б], старший преподаватель, orcid.org/0000-0002-7903-2480
[а]Санкт-Петербургский государственный электротехнический университет «ЛЭТИ», Профессора Попова ул., 5, Санкт-Петербург, 197376, РФ
[б]Санкт-Петербургский государственный университет аэрокосмического приборостроения, Б. Морская ул., 67, Санкт-Петербург, 190000, РФ

Постановка проблемы: хранение данных требует своевременного масштабирования инфраструктуры для размещения данных, поступающих на хранение. С учетом стремительного накопления данных необходимы новые модели управления емкостью хранилища, которые должны учитывать иерархическую структуру хранилища данных, разные требования к хранению файлов и ограничения на объем носителей. **Цель исследования:** предложить модель своевременного масштабирования инфраструктуры хранения данных, основанную на прогнозных оценках наступления того момента, когда заполнится емкость носителей данных. **Результаты:** разработана и приведена модель управления емкостью системы хранения данных, основанная на анализе паттернов состояния системы хранения. Паттерн представляет собой матрицу, каждая ячейка которой отражает состояние заполнения носителя системы хранения данных на соответствующем уровне иерархической структуры системы хранения. Ячейка матрицы характеризуется реальным, предельным и максимальным значениями емкости носителя. Задача масштабирования системы хранения данных заключается в прогнозной оценке наступления событий достижения предельной емкости и максимальной емкости носителя данных. Разница между прогнозными оценками есть время, которое выделено администратору для подключения дополнительных носителей. Предложено вычислять значения прогнозных оценок времени программным способом, применяя методы машинного обучения. Показано, что при построении краткосрочного прогноза методы машинного обучения проигрывают в точности ARIMA — интегрированной модели авторегрессии и скользящего среднего. Однако при построении долгосрочного прогноза методы машинного обучения дали результаты, соизмеримые с теми, что обеспечивает ARIMA. **Практическая значимость:** предложенная модель управления масштабированием системы хранения данных необходима для своевременного выделения емкости для поступающих на хранение данных. Реализация этой модели на входе хранилища позволяет автоматизировать процесс подключения носителей, что предотвращает потерю входящих в систему данных.

Ключевые слова — система хранения данных, носители, многоуровневое хранение данных, хранилище данных, масштабирование хранилища, управление хранилищем, прогнозирование требуемой емкости хранения.

Для цитирования: Sovetov B. Ya., Tatarnikova T. M., Poymanova E. D. Storage scaling management model. *Информационно-управляющие системы*, 2020, № 5, с. 43–49. doi:10.31799/1684-8853-2020-5-43-49

For citation: Sovetov B. Ya., Tatarnikova T. M., Poymanova E. D. Storage scaling management model. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2020, no. 5, pp. 43–49. doi:10.31799/1684-8853-2020-5-43-49