Modeling NoSQL Systems in Many-nodes Hybrid Environments

M.M. Rovnyagin[1], F.N. Chernilin[2], A.V. Guminskaia[4], V.M. Kinash[4], O.V. Myltsyn[5], A.P. Orlov[6]

National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russian Federation
[1]m.rovnyagin.2015@ieee.org, [2]fedor.chernilin@gmail.com,
[3]avguminskaya@mephi.ru, [4]kinash000@mail.ru,
[5]oleg.myltsyn@yandex.ru, [6]alexandrorlovne@yandex.ru

A.V. Kuzmin
Information and computing systems Dept.
Penza State University
Penza, Russian Federation
flickerlight@inbox.ru

Abstract **— Data search is one of the most important problems in the field of computer science and computer facilities. Classical relational DBMSs (RDBMSs), unfortunately, are not suitable as data storage systems for Big Data. Therefore, the concept NoSQL is now widely spread. A common feature of such systems is a high throughput and linear scalability, depending on the number of storage servers used. One of the most productive NoSQL-systems, at the moment is Apache Cassandra. In this paper, we suggest ways to simulate the performance of such systems in hybrid computing environments.**

Index Terms**—CPU/GPU hybrid architecture, NVIDIA CUDA, Distributed Databases, NoSQL**

I. Introduction

Most modern data storage and retrieval systems support distributed computing technology [1]. At the same time, the equipment and concepts underlying this can be very different. Since the end of the 2000s, a number of NoSQL-systems with various capabilities have appeared.

Most of the current NoSQL-systems are built in accordance with the architectural principles underlying such solutions as Google BigTable [2] and Amazon Dynamo [3]. NoSQL-systems use an aggregate-oriented data model. Unlike RDBMS tuples, the aggregate is a complex data structure that can contain lists of values or nested structures. The aggregates can be distributed among the nodes of the cluster, which will increase the overall throughput of the system. Consistent access to data is ensured within a single aggregate.

Numerous benchmarks of NoSQL-systems show a significant advantage over classic distributed RDBMS [4,5], and one of the fastest solutions among NoSQL-systems is Apache Cassandra [6].

In [7, 8, 9], methods were proposed for using hybrid computing technologies to accelerate NoSQL-systems and expand their functionality. These methods are based on the fact that the target computing system is hybrid (CPU/GPU) or at least multithreaded. They are applicable to most hardware platforms, allowing you to get performance gains regardless of the number of cores, network topologies, etc.

In this paper, we propose a mathematical description of the above approaches to the use of hybrid technology in computing-NoSQL systems. The proposed model is built on the basis of queuing theory and can be used to identify the main components, the improvement of which allows to substantially increase the overall performance of NoSQL solutions.

II. The Model of NoSQL systems

A typical plan for processing client search queries in NoSQL-systems consists of the following stages:

1) Initial client request to the system.
2) Forwarding the request to the access server, with load balancing.
3) Performing data queries on storage nodes. The number of affected storage nodes is determined by the mapping scheme installed on the access server (hash ring).

4) Executing requests for data in external memory.
5) Formation of the answer: transfer of the found data, or notification of their absence.
6) Send a final response to the client.

In the same way, the operations of adding and deleting data are performed. If you add a new item to the storage system, the client receives the unique key (identifier) of the stored item. Upon deletion – the operation return code. Update operations are implemented by the client application.

The data itself is stored directly on the storage nodes, and the corresponding metadata is stored on the access servers. Metadata can be used for queries filtering, maintain consistency, and so on.

The classic key-value NoSQL system is shown in Figure 1 as a queuing network.

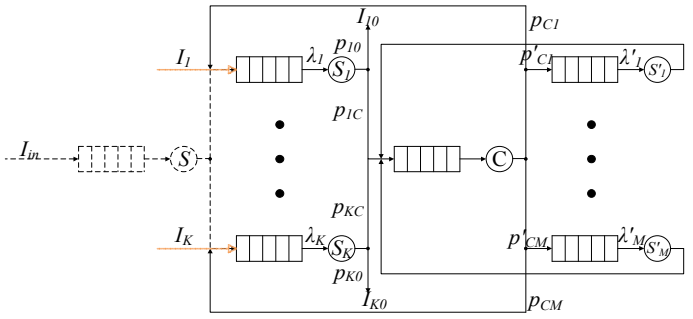

Fig. 1. – NoSQL as a queuing network

Due to: a large population of independent NoSQL-users (stationary property), use of a qualitative input hash function (memoryless property) and the properties of the TCP/UDP protocol (ordinary flow property) it is possible to consider the input stream as a Poisson process. Notation: S, C, Si and Si` - balancer, interconnection network, access servers and storage nodes, represented in the form of mass servicing systems; $I_{in}$, $I_i$ - the intensity of incoming input streams of requests to the queuing network, $\lambda_i$, $\lambda_i$` - the intensity of receipt of requests for the queuing system, p_{ij}, p`$_{ij}$ - the probabilities of queries transfer between queuing systems.

The internal structure of the queuing systems S_i and S_i` is represented in the form of queuing networks in Figures 2 and 3, respectively.

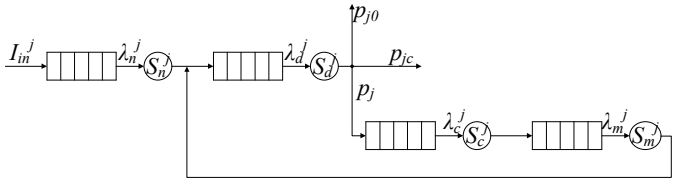

Fig. 2. – The access server, represented as a queuing network

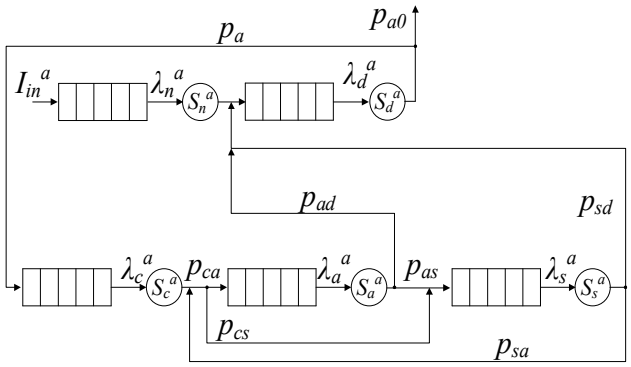

Fig. 3. – The storage node, represented as a queuing network

Subsystem designations are introduced: S_n^a, S_n^j - network subsystem, S_d^a, S_d^j - network activity management subsystem, S_c^j - consistency provisioning subsystem, S_m^j - key partitioning subsystem, S_c^a - query classification subsystem, S_a^a - pre-processing subsystem, S_s^a - storage subsystem. The probabilities for continuing processing requests in the queuing network of the access server and the storage node are denoted by p_j and p_a, respectively.

To evaluate the performance of individual queuing systems, the Amdahl law can be applied [10]. In the classical version, it is formulated as follows. Let α be the proportion of the program that can be paralleled ideally (ie, the running time of the parallelized section will be proportional to the number of processors p). Accordingly, (1-α) is the fraction of the code that is not paralleled. Then the acceleration compared to the uniprocessor version of the program will be:

$$A = \frac{1}{(1-\alpha)+\frac{\alpha}{p}} \quad . \qquad (1)$$

There are papers [11, 12] in which the law of Amdahl for multi-core architectures with limited memory bandwidth is specified. Such models can be useful for evaluating the performance of software solutions for modeling various processes. In such applications, the memory and interconnect capacity is critical because there are active interactions between processors. However, in NoSQL-systems parallel threads are independent and numerous (in proportion to the number of requests to NoSQL). In this case, great importance is acquired by such parameters as time switching context and the presence of coprocessors.

Let's introduce the following notation:

- p - number of processors (processor cores CPU / GPU / FPGA-cores)

- n - number of threads
- T - the execution time of the serial version of the algorithm
- Tp - the execution time of the parallel section of the program
- Ts is the execution time of the sequential program section
- t_c - the time of the context change (switching the processor core from one thread to another), and the coefficient $q = t_c / T$.
- τ - the execution time of the program section on the real processor (CPU / GPU)
- t is the execution time of the program section in the ideal system ($n = p$)
- $\alpha = Tp / T$ - coefficient of parallelism
- β - coefficient of coprocessor usage
- A - acceleration

In case of the processes number exceeds the number of cores, the calculations will occur in the time-sharing mode:

$$c = \begin{cases} 1, n \le p \\ \frac{n}{p}, n > p \end{cases}, \quad (2)$$

where c is the load factor of the computational cores. From the point of view of the parallel program logic on the CPU, (2) will be taken into account when calculating τ^{CPU} as follows:

$$\tau^{CPU} = c(t^{CPU} + t_c) \quad . \quad (3)$$

For an ideal system (in which $n = p$), t^{CPU} can be expressed as:

$$t^{CPU} = \frac{T_p^{CPU}}{n} = \frac{\alpha T(1-\beta)}{n} \quad . \quad (4)$$

In the case where $\beta \in (0,1]$, then the program can be parallelized on the coprocessor, then the value of τ^{GPU} is:

$$\tau^{GPU} = \frac{\beta\, T_p}{p^{GPU}} = \frac{\alpha\beta T}{p^{GPU}} \quad . \quad (5)$$

From (1), (3), (4), (5) it follows that:

$$A = \frac{1}{(1-\alpha)+c\left(\frac{\alpha(1-\beta)}{n}+q\right)+\frac{\alpha\beta}{p^{COP}}} \quad . \quad (6)$$

Thus, a NoSQL system using hybrid computing technologies to speed up search operations and additional data processing can be modeled by the system of equations (7):

$$\begin{cases} \overline{\mathrm{T}}_{pr} = \frac{\sum_{j=1}^{K} \lambda_j \overline{\mathrm{T}}_{\mathrm{pf_j}} + \sum_{a=1}^{M} \lambda\prime_a \overline{\mathrm{T}}_{\mathrm{pf}_a} + \lambda_c \overline{\mathrm{T}}_{\mathrm{pf}_c}}{I_{\mathrm{in}}} \\ \overline{\mathrm{T}}_{\mathrm{pf}_j} = \frac{\lambda_n^j \frac{\overline{\mathrm{T}}_{srv_n}}{A_n - \lambda_n^j \overline{\mathrm{T}}_{srv_n}} + \lambda_d^j \frac{\overline{\mathrm{T}}_{srv_d}}{A_d - \lambda_d^j \overline{\mathrm{T}}_{srv_d}} + \lambda_c^j \frac{\overline{\mathrm{T}}_{srv_c}}{A_c - \lambda_c^j \overline{\mathrm{T}}_{srv_c}} + \lambda_m^j \frac{\overline{\mathrm{T}}_{srv_m}}{A_m - \lambda_m^j \overline{\mathrm{T}}_{srv_m}}}{I_{\mathrm{in}}^j} \\ \overline{\mathrm{T}}_{\mathrm{pf}_a} = \frac{\lambda_n^a \frac{\overline{\mathrm{T}}_{srv_n}}{A_n - \lambda_n^a \overline{\mathrm{T}}_{srv_n}} + \lambda_d^a \frac{\overline{\mathrm{T}}_{srv_d}}{A_d - \lambda_d^a \overline{\mathrm{T}}_{srv_d}} + \lambda_c^a \frac{\overline{\mathrm{T}}_{srv_c}}{A_c - \lambda_c^a \overline{\mathrm{T}}_{srv_c}} + \lambda_a^a \frac{\overline{\mathrm{T}}_{srv_a}}{A_a - \lambda_a^a \overline{\mathrm{T}}_{srv_a}} + \lambda_s^a \frac{\overline{\mathrm{T}}_{srv_s}}{A_s - \lambda_s^a \overline{\mathrm{T}}_{srv_s}}}{I_{\mathrm{in}}^a} \end{cases} \quad (7)$$

where $\overline{\mathrm{T}}_{\mathrm{pr}}$ expresses the time of the request in NoSQL, $\overline{\mathrm{T}}_{\mathrm{pf}_j}$ and $\overline{\mathrm{T}}_{\mathrm{pf}_a}$ - the time of the application stay on the access server and the storage node, respectively, and $\overline{\mathrm{T}}_{\mathrm{srv}}$ - the service time on specific queuing systems, and the indices denote the corresponding subsystems (when the condition that the load factor of each subsystem ≤ 1, otherwise it is necessary to increase the number of NoSQL nodes).

III. Estimation of Models Adequacy

The model proposed above allows us to evaluate the two most important performance characteristics of NoSQL-systems: response time and throughput. One of the most important questions from the point of view of organizing an experiment is to estimate the number of tests that need to be carried out to achieve the required accuracy. Based on the theory of the organization of experiments, the number of tests can be determined in accordance with the time of queries receipt distribution law:

$$n \ge \left(\frac{t s_V}{\delta}\right)^2, \quad (8)$$

where δ is the accuracy, s_v is the standard deviation from the preliminary series of tests, t is the argument of the Laplace function corresponding to the given confidence probability.

The adequacy of the model was assessed using the Fisher criterion, which is based on the analysis of variance. For this purpose, the dispersion of adequacy and the variance of the reproducibility of the series of experiments were calculated from formulas (9) and (10).

$$S_{ad}^2 = \frac{\sum_{i=1}^{N} n_i (\overline{y}_i - \hat{y}_i)^2}{f_1}, \quad (9)$$

$$S_{\{y\}}^2 = \frac{\sum_{i=1}^{N} \sum_{q=1}^{n} (y_{iq} - \overline{y}_i)^2}{f_2}, \quad (10)$$

where N is the number of different experiments, n_i is the number of parallel tests in the i-th experiment. For example, in one of the series of experiments, the dependence of the response time on the number of CPU cores was investigated. In this case, 3,000 parallel tests were performed for each value of the parameter «number of CPU

cores» (Figure 4). Thus, for this series: $N = 3$, $n = 3000$.

Other notation: y_{iq} is the value obtained in the q-th test of the i-th experiment (a posteriori value), $\overline{y}_i$ is the average value obtained with a confidence probability of 0.95 on the real system after conducting n_i parallel tests in the i-th experiment (mean a posteriori value), $\hat{y}_i$ is the value obtained using the model in the i-th experiment (a priori value), f_1, f_2 is the number of degrees of freedom.

The value of F obtained by formula (11) was compared with the tabulated F_T. As a result, for all series of experiments F was less than F_T, which indicates the adequacy of the NoSQL model proposed above.

$$F = \frac{S^2_{\text{ad}}}{S^2_{\{y\}}} \qquad (11)$$

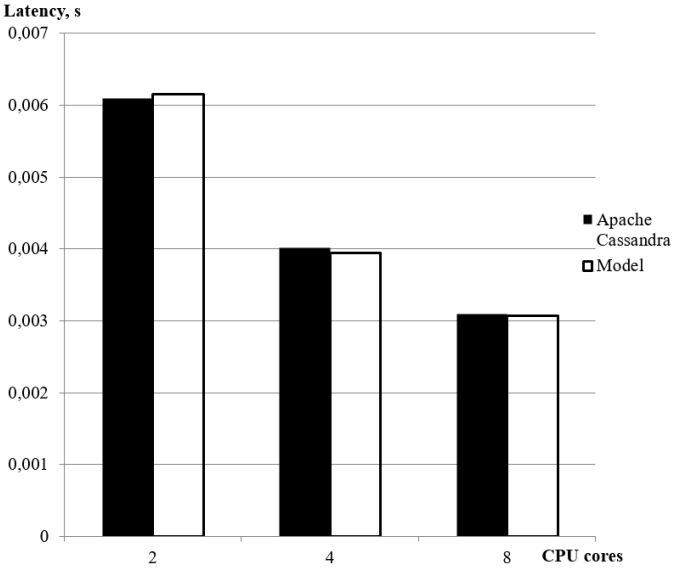

Fig. 4. – Response time on the number of cores of the CPU

IV. Conclusion

Thus, in the present work, a mathematical model of NoSQL is developed for systems of key-value type, intended for deployment in hybrid computer systems. In the form of a queuing network, the architecture of modern NoSQL-systems is formalized, reflecting the features of the interaction of their constituent parts. Amdal's law for multi-threaded multi-core hybrid systems is refined. The adequacy of the model has been verified experimentally.

Acknowledgment

The authors would like to express their gratitude to the Head of Computer Systems and Technologies Department of the National Research Nuclear University «MEPhI», Professor M. A. Ivanov for his assistance and support during the research, as well as to the Head of Inter-Facility Hybrid Computing Systems Laboratory, Assistant Professor N. P. Vasilyev for the provided equipment and consultations.

References

[1] Chansler, R. The Architecture of Open Source Applications. Kindle Edition / Robert Chansler, Russell Bryant, Roy Bryant, Rosangela Canino-Koening, Francesco Cesarini, Eric Allman, Keith Bostic, Titus Brown. — Amazon Media EU, 2012. — 432 p.

[2] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. 2008. Bigtable: A distributed storage system for structured data. ACM Trans. Comput. Syst. 26, 2, Article 4 (June 2008), 26 pages.

[3] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels: "Dynamo: Amazon's Highly Available Key-Value Store". SOSP'07: Proc. Twenty-First ACM SIGOPS Symposium on Operating Systems Principles, pages 205–220, 2007

[4] Why Migrate from MySQL to Cassandra?: [Online] / DataStax. Available: http://www.datastax.com/wp-content/uploads/2012/08/WP-DataStax-MySQLtoCassandra.pdf

[5] NoSQL vs Traditional Relational DB Performance: [Online] / Available: http://micrasystems.com/hello-world

[6] Benchmarking Top NoSQL Databases: [Online] / Available: http://blog.endpoint.com/2015/04/new-nosql-benchmark-cassandra-mongodb.html

[7] N. P. Vasilyev and M. M. Rovnyagin, "Software platform VAR for heterogeneous GPGPU-systems," 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), Noida, 2016, pp. 624-629

[8] A. A. Kozlov, A. A. Aleshina, I. S. Kamenskikh, M. M. Rovnyagin, D. M. Sinelnikov and D. A. Shulga, "Increasing the functionality of the modern NoSQL-systems with GPGPU-technology," 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), St. Petersburg, 2016, pp. 242-246.

[9] I. S. Kamenskikh, D. M. Sinelnikov, D. S. Kalintsev, A. A. Kozlov, M. M. Rovnyagin and D. A. Shulga, "Software development framework for a distributed storage and GPGPU data processing infrastructure," 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), St. Petersburg, 2016, pp. 216-219.

[10] Amdahl, Gene M. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. AFIPS Conference Proceedings, pages 483–485, 1967

[11] Erlin Yao , Yungang Bao , Guangming Tan , Mingyu Chen, Extending Amdahl's law in the multicore era // ACM SIGMETRICS Performance Evaluation Review, v.37 n.2, September 2009.

[12] Xian-He Sun , Yong Chen, Reevaluating Amdahl's law in the multicore era // Journal of Parallel and Distributed Computing, v.70 n.2, p.183-188, February, 2010.