Software platform VAR for heterogeneous GPGPU-systems

N.P. Vasilyev[1], M. M. Rovnyagin[2]
Department of Computer Systems and Technologies
National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
Moscow, Russian Federation
[1]npvasilyev@mephi.ru, [2]m.rovnyagin.2015@ieee.org

***Abstract* — Currently, the process of generation of new data becomes an "avalanche". Classic RDBMS have little use as storage systems for Data Mining applications. The concept of NoSQL (not only the SQL) storage systems has currently become widespread. This article discusses some of the issues related to the organization of storage and searching in multicomputer systems. The modern principles of NoSQL-systems are described. Examines opportunities to accelerate the work of such systems through the prism of GPGPU-technologies. We present the results of comparative testing of high-performance search and secure storage of data and NoSQL-VAR system Apache Cassandra.**

***Keywords*—Software platform, Graphics processing units, Data storage systems, High performance computing, NVIDIA CUDA, GPGPU**

I. Introduction

NoSQL [1] systems, though not having full functionality of the RDBMS, can significantly increase the throughput of applications when working with large amounts of semi-structured data. An example of such data could be: experimental data, statistics of web-service calls, weather data, social medias and microblogging info outflow, etc. In each of the cases it is possible to implement parallel data processing on a distributed computing system. At the same time, it requires storage systems that can handle millions of relatively simple forms of queries.

The above considerations have directed the developers to create NoSQL high-performance data storage systems. Plenty of similar systems with different capabilities have emerged starting from the second half of the 2000s. Some of them implement a software interface that allows translating some classic RDBMS instructions into an internal representation, and some of them partially implement the ACID requirements. A common feature of these systems is a high rate of throughput that is scalable (mostly linear) depending on the number of storage servers.

Many companies around the world currently use distributed high-performance NoSQL-solutions. Services of such corporations as IBM, Amazon, Facebook, Twitter, Google, Oracle directly depend on how effectively data storage and retrieval systems operate. Among the most important performance criteria of the implemented NoSQL-solutions are the following features: throughput, scalability, energy efficiency and maintenance costs. As you can see, these specifications resemble the requirements to an advanced supercomputer.

Indeed, the task of data retrieval is also a computing one, with the difference that working with memory is a priority. However, the NoSQL-approach involves the use of standard cluster solutions, in contrast to the RDBMS running on dedicated storage servers. Thus, it makes sense to consider high-performance distributed data storage and retrieval systems in the context of supercomputer technologies.

Currently, leading positions in the supercomputer ranking are taken by hybrid solutions [2]. Hybridity here refers to the presence of calculators with different architectures in a supercomputer. GPU-accelerators or FPGA chips can be used as coprocessors for classic CPU. These devices can significantly speed up some operations and increase energy efficiency. From the point of view of data storage and retrieval problem, coprocessors can be useful to increasing speed of search, verification of set membership, further processing operations (encryption, image processing, audio file analysis).

Currently existing NoSQL-solutions are not intended for use in hybrid supercomputer systems. Thus, the development of methods and means of solving the tasks of search and secure data storage through hybrid computing technologies is an actual scientific and engineering issue.

II. NoSQL-Systems Architecture

Development of NoSQL-systems is mainly related to the active work of the research community. There is a number of publications with a serious effect on the design of modern high-performance storage systems. Most of the current NoSQL-systems are built in accordance with the architectural principles underlying such solutions as Google's BigTable [3] and Amazon's Dynamo [4]. The NoSQL-systems use aggregate-oriented data model. Unlike RDBMS tuples, aggregate is a complex data structure, which may contain a list of values or nested structures. The aggregates can be distributed over cluster nodes, increasing the overall system throughput. Consistent data access is provided within a single aggregate. The way it is represented is determined by the type of a NoSQL-system. There are three common data models for aggregates: key-value, document, and column-family.

Key-value NoSQL systems provide the user with an interface, using which he/she can get an object through a key (unique ID). An entry is mapped to each key as an object consisting of several attributes. The composition of the attributes for different entries may vary. The processing of inhomogeneous records is mastered by the application developer. Functional development of key-value systems are document-NoSQL-systems. Instead of value they use a fixed format data set (usually JSON). Widespread storage systems are column-family. The data in these systems are represented as an associative array, where the keys are the column names. An associative array and key are combined in a line. Multiple lines are column-families that are stored in these lines.

To improve data availability in the NoSQL-systems they usea replication (copying) of data. Most NoSQL-systems use peer replication, which allows overcoming the limitation of performance for records. The storage nodes are addressed either directly by the application (using NoSQL database driver), or a randomly selected server of peer network (coordinator, see Fig. 1) operates in the system on behalf of the user. The distribution of data in network nodes occurs by means of a continuous hashing (hash-ring).

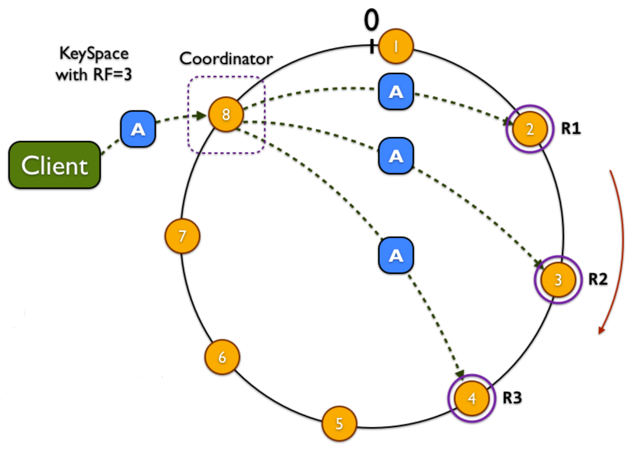

Fig. 1. Continuous Hashing within the System. Replication Factor is 3

This approach allows you adding and deleting nodes, while flexibly adjusting the balance depending on nodes performance. The main problem of peer replication is consistency maintenance. The lack of consistency may occur, for example, due to network failures.

Most NoSQL-systems are storage systems using the main memory with the ability to “push” the unused tables to external memory. For example, Apache Cassandra NoSQL-system when working with data in the RAM uses commitlog [5] and table files on the hard disk. When the memory is full (Java-heap [6]), a new table is created in the memory (memtable), and the old one is written onto the disk (SSTable). Writings to the log and unloading of tables generate sequential accessing the hard disk, which speeds up system performance. Memory transactions of Apache Cassandra are presented in Fig. 2.

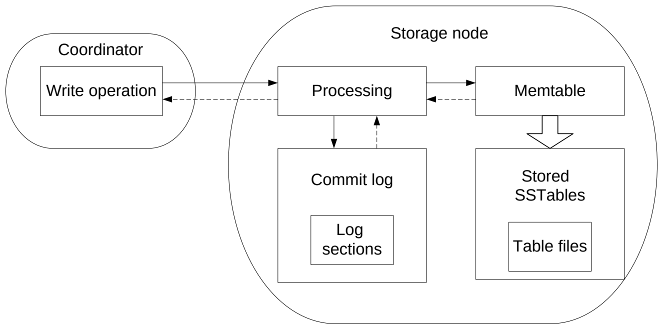

Fig. 2. Apache Cassandra Memory Organization

Some NoSQL-system are written in C++ (MongoDB), Erlang (CouchDB), etc., But most of these solutions were originally developed in Java. The use of Java technology with comparable performance metrics allows to significantly increase the security by using the JRE runtime. However, none of the existing NoSQL-systems use coprocessors to speed up computing. The lack of appropriate means and methods prevents the effective use of hybrid systems as a hardware platform for NoSQL. Nevertheless, performance of a number of NoSQL-system mechanisms can still be improved. For example, hashing running in volume by the Coordinator to check the Bloom status vector [7] or the hash-ring can be essentially improved through the use of GPGPU-coprocessor. To speed up data retrieval there can be arranged a corresponding index in the memory of GPGPU-coprocessor located on the storage node. Of course, as a consequence of insufficient memory in the accelerator the index size will be limited. However, this approach may be useful to speed up retrieval in a particular table on demand. Finally, coprocessors can tenfold improve operations of additional data processing (transparent data encryption [8], calculation of image the signaturesetc.). To implement the above features the following is needed: presence of an extensible software framework that allows connecting/disconnecting the required modules of additional processing; availability of appropriate parallel algorithms adapted to run on a GPGPU-coprocessors.

III. Related Works

The advent of hybrid computing systems drew great attention to the use of the GPU to speed up retrieval in these tree structures. Jordan Fix et al. [9] showed that the use of CUDA[10] to retrieve data in B+ tree could give a more than 10-fold acceleration in sampling operations. But, if there is a need to replace the retrieved set of data in the GPU memory, this approach is inefficient and leads to repeated poor performance as compared to CPU implementation. Significant overhead costs also appear as a result of the need to synchronize CUDA threads. Complex performance research of multithreaded CPU and GPU implementations for B-Trees have been carried out by Changkyu Kim et al. in [11]. For GPU-implementation they managed to achieve 1.7-fold increase in performance compared to the previous GPU-version of the algorithm. The researchers faced serious problems when trees were exceeding the memory of the graphics processor. In general, the acceleration was achieved by binding their implementation to a specific architecture. Dimensions of tree nodes, their number on the level and the mode of transmission were directly determined by the architectural features of the GTX 280 accelerator. The experimental results showed that due to the highly limited amount of GPU memory and a high latency of data transfer interface allowed performance gains only when working with rarely changing data provided when a program is well adapted to a specific kind of GPU.

There is a number of publications [12,13,14] where a possibility of using CUDA-coprocessor to speed up the Bloom filter is investigated. In some versions the Bloom vector is located in shared memory; in others, it is located in the global one. After all, the Bloom vector may be stored in the RAM, wherein the main performance gain is achieved through the implementation of hashing operations in the coprocessor's ALU.

A significant number of publications deal with improvements in cryptographic algorithms using CUDA. In publication [15], the authors analyze the benefits of CUDA compared to the previously used OpenGL. According to the baseline of the work, implemented by the researchers of a parallel version of the AES algorithm, input information is stored in the global memory of CUDA-devices and replacement units are located in a constant memory cached area to accelerate the process of data reading from the S-blocs.

The authors of the research [16] in their publication reviewed several possible approaches to the implementation of a multi-threaded AES version. The article presents the results of comparing the performance of serial, and parallel CPU-, and GPU-versions. As a result, the GPU-implementation was the most productive.

In the article [17], the authors overview all modes of AES encryption and highlight those suitable for parallelization: ECB (Electronic codebook), CTR (Counter). For the CBC (Cipher-block chaining) mode it is only possible to perform parallelization of decryption operations. For these modes, the authors present the charts of throughput values and some tips on improving performance.

In addition to the articles dealing with the CUDA-based implementation of the AES algorithm, there are also a lot of publications, describing the implementation of parallel versions of other encryption [18] and hashing algorithms in GPGPU-systems.

IV. VAR SOFTWARE PLATFORM

Like any modern NoSQL-system, VAR software platform is designed for use in a multicomputer peer networks consisting of access servers and storage nodes. The matrix of role distribution per system nodes may be arbitrary, as the same computer can be both the access server and the storage node simultaneously. As shown in Fig. 3, in general, the system has K access servers and M storage nodes.

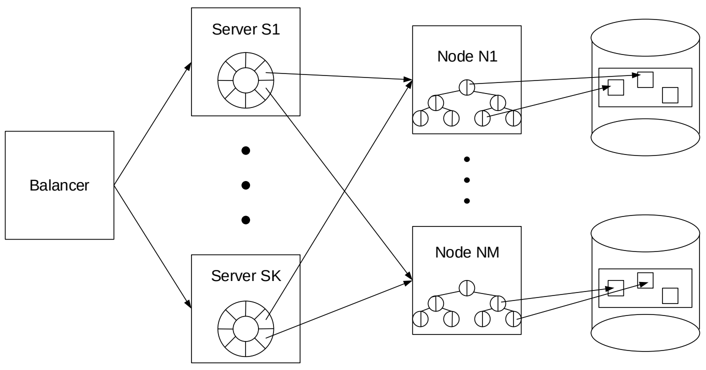

Fig. 3. Generalized Diagram of VAR Distributed High-performance Data Storage and Retrieval System

Access servers coordinate the processes of retrieving/adding/deleting the data, responsible for maintaining consistency, transactions, contain information on data arrangement scheme within the system.

Storage nodes manage the information writing into the RAM or external memory;they perform additional data processing.

Balancer subsystem is quite specific component, which direct participation in data processing is quite limited, and having the main purpose in prompt redirection of a new incoming connection to the least loaded server.

Figure 4 is a schematic representation of a data access server. The composition of the server is completely determined by its purpose. This should "be" in the two networks simultaneously: the internal data storage network and the external one, from which it receives user queries. Network subsystem for external and internal transfer are provided as part of the server to initialize new network connections.These are software modules, creating socket TCP-connections and selecting the appropriate handler. After initialization, control of the newly created connections is transferred to network activity management subsystem.

Transactions support subsystem collects statistics on the stages of query execution in the distributed system and forwards control actions to network activity management subsystem. The nature of the control actions can be different:

- Confirming successful writing, registering new data on the storage nodes
- Sending a request for re-writing to storage nodes
- Sending a message to the user on the successful changes registration
- Request for information on the versions of stored data
- Sending regular signal of versions synchronization
- etc.

To determine the target storage nodes Coordinator communicates with the key space separation subsystem, which is represented by a hash-ring.

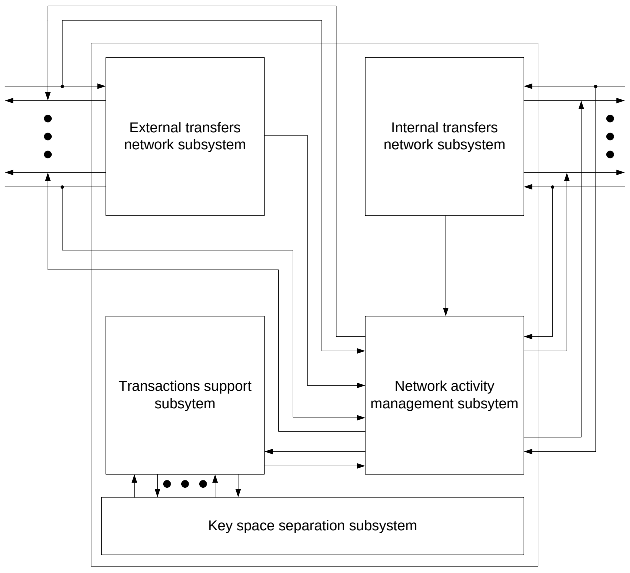
External transfers network subsystem
Internal transfers network subsystem
Transactions support subsytem
Network activity management subsytem
Key space separation subsystem

Fig. 4. Data access server diagram

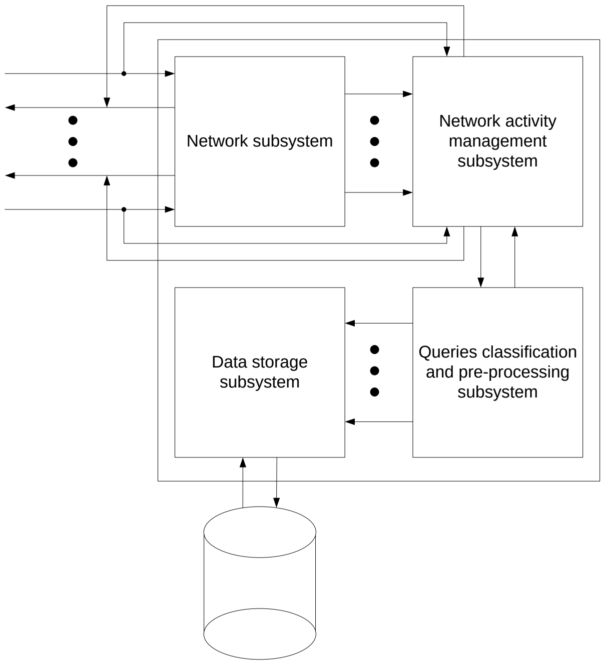
Network subsystem
Network activity management subsystem
Data storage subsystem
Queries classification and pre-processing subsystem

Fig. 5. Storage Node Diagram

Figure 5 shows a schematic representation of the storage node. The composition of the storage node partly repeats the structure of the data access server. The processing of internal system network connections is handled by networking subsystem and network activity management subsystem.

Incoming queries of network subsystems are forwarded to the input of the subsystem for queries classification and pre-processing. At this point, it reads the type of query and, if necessary, performs additional data processing.

Data storage subsystem is directly responsible for the storage and retrieval of data. It is a memcached-system with the ability to upload data to external memory.

VAR system is written in Java. The network subsystem is developed on the basis of Java NIO.2[19] and is a scalable multi-threaded solution. The format of messages processed by the system is binary. Accelerated computing is achieved by the use of CUDA-coprocessor interacted by jCuda[20] library. VAR software platform includes a software framework, consisting of a hierarchy of classes, interfaces and their underlying implementations that can be modified and supplemented by the use of Dependency Injection [21] according to the user's requirements and hardware configuration. Transaction support subsystem of access server and data storage subsystem use CUDA-coprocessor to accelerate the following operations: hashing, creating table indexes in GPU memory, data encryption when unloading to the external memory and randomization of key at the input of the stochastic search tree [22]. Data encryption and upload to the external memory is performed by a separate Java-thread using its own context of CUDA-flow, and runs in the background.

V. CONFIGURATION

The modified Bloom filter for hybrid systems was implemented

This article presents the results of performance testing of VAR software framework storage node. The comparison is made against the results of performance measuring of Apache Cassandra in a single-node configuration with replication factor 1. The configuration of the test system: Intel Core i7 – 3770K, 32 GbDDR3, NVIDIA GeForce GTX680 (4GbGDDR5). Generation of test packages was done through the use of Apache Jmeter performance stress testing utility [23]. It allowed creating multiple threads that simulated user activities. In this instance, the test involved up to 100 virtual users, each of which generated queries to add (1 Kbyte of data in each query) and retrieve data. The VAR system included a module for data encryption as per GOST 28147-89 algorithm which operated in the background while unloading data into the external memory. The encryption was carried out by the GPU. For comparison, Figure 6 shows the results of unit-testing of the sequential single-threaded CPU and a parallel GPU-version.

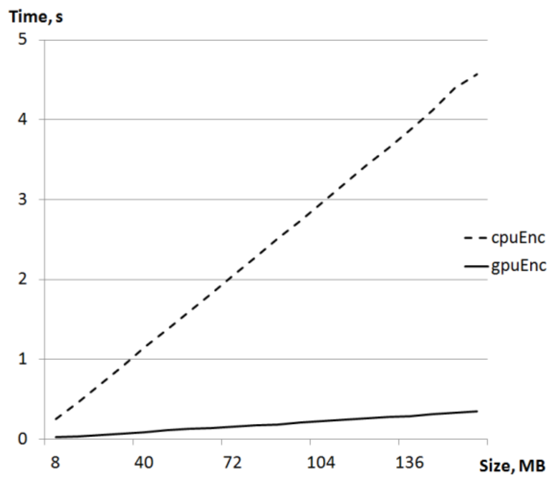

Fig. 6. Performance of GOST 28147-89 Data Encryption Algorithm for the CPU and GPU Versions

VI. Experimental Results

Stress testing was carried out as follows: Jmeter escalated the number of virtual users (VU in the figure below) up to 100, and then again reduced it to 0 and finalized the test. At the same time, there were the following three main indicators recorded: the number of transactions per second (TPS), the presence of the expected response code and response time.

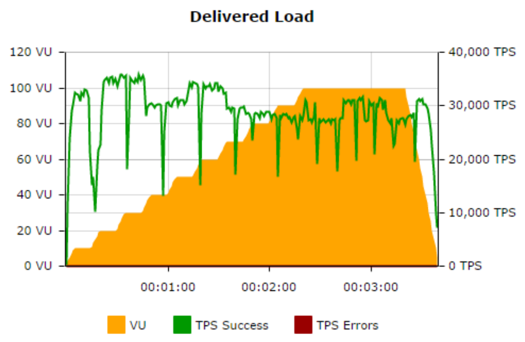

Fig. 7. Performance (VAR System)

The diagram in Figure 7 shows that for 100 virtual users one storage node of VAR system is able to maintain performance at 27k transactions per second with some “backlogs” during re-initialization of memcached-table. The response time in this case is an average of 2 ms, which is an excellent indicator for NoSQL-systems.

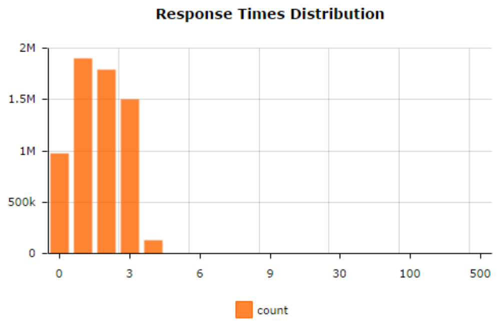

Fig. 8. Response Times (VAR System)

The results of the same type of test with similar Apache Cassandra instance are shown in Figures 9 and 10. On average, it can be noted that Apache Cassandra inferior performance compared to VAR system is 35-40%, and has two times bigger response time.

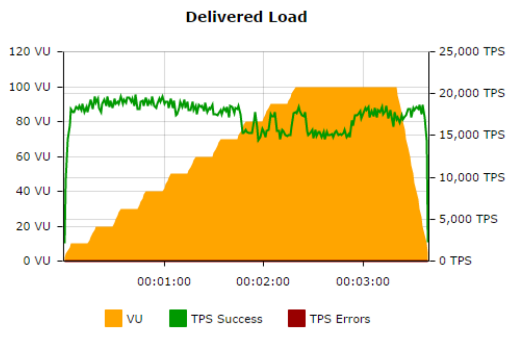

Fig. 9. Performance (Apache Cassandra)

It is also important to note that, while having more modest figures, Cassandra does not encrypt the data written to disk.

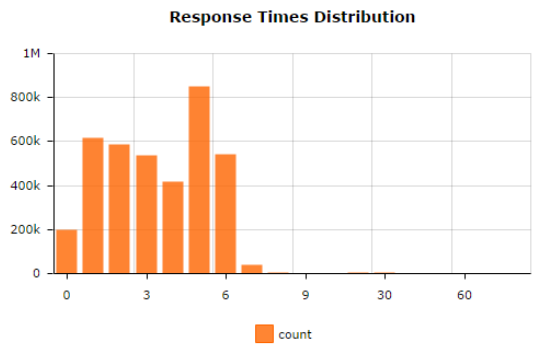

Fig. 10. Response Times (Apache Cassandra)

To forecast the possible increase in performance obtained by the use of hybrid coprocessors to accelerate the retrieval and additional data processing (encryption), there is an Amdahl’s law deduction for hybrid systems represented by the formula (1):

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

where A is the acceleration; α – parallelism factor (the share of parallelizable part of the program to the entire program); n – number of parallel processes, β – coprocessor utilization factor (share part of the code parallelized by coprocessor to the entire parallelized code); pCOP–number of cores of the discrete coprocessor (COP index indicates the type of the coprocessor: GPU/FPGA/MIC, etc.); c – CPU cores load factor, q – ratio of the CPU context switch to allprogram running time.

Thus, the operation of high-performance distributed data storage and retrieval system using hybrid supercomputing technologies to speed up retrieval and additional data processing, the operations can be modelled by the system of equations (2):

$$\begin{cases} \overline{T}_{preb} = \dfrac{\sum_{j=1}^{K}\lambda_j \overline{T}_{pr_j} + \sum_{a=1}^{M}\lambda'_a \overline{T}_{pr_a} + \lambda_c \overline{T}_{pr_c}}{I_{vh}} \\ \overline{T}_{pr_j} = \dfrac{\lambda_n^j \frac{\overline{T}_{obs_n}}{A_n-\lambda_n^j \overline{T}_{obs_n}} + \lambda_d^j \frac{\overline{T}_{obs_d}}{A_d-\lambda_d^j \overline{T}_{obs_d}} + \lambda_c^j \frac{\overline{T}_{obs_c}}{A_c-\lambda_c^j \overline{T}_{obs_c}} + \lambda_m^j \frac{\overline{T}_{obs_m}}{A_m-\lambda_m^j \overline{T}_{obs_m}}}{I_{vh}^j} \\ \overline{T}_{pr_a} = \dfrac{\lambda_n^a \frac{\overline{T}_{obs_n}}{A_n-\lambda_n^a \overline{T}_{obs_n}} + \lambda_d^a \frac{\overline{T}_{obs_d}}{A_d-\lambda_d^a \overline{T}_{obs_d}} + \lambda_c^a \frac{\overline{T}_{obs_c}}{A_c-\lambda_c^a \overline{T}_{obs_c}} + \lambda_a^a \frac{\overline{T}_{obs_a}}{A_a-\lambda_a^a \overline{T}_{obs_a}} + \lambda_s^a \frac{\overline{T}_{obs_s}}{A_s-\lambda_s^a \overline{T}_{obs_s}}}{I_{vh}^a} \end{cases}, \tag{2}$$

where$\overline{T}_{preb}$expresses the time when the query is in high-performance distributed data storage and retrieval system, $\overline{T}_{pr_j}$ and $\overline{T}_{pr_a}$ – the time when the query is in access server the storage node, respectively, and T_{obs} is service time for specific queueing system, and the indices are for the relevant subsystem.

Thus, VAR software framework in minimal configuration has not only equal, but superior performance compared to advanced counterparts. Further research is planned to repeat similar tests for tables indexed by GPU, test multicomputer VAR configuration, research other encryption algorithms and their operation as modules of additional data processing before unloading to the disc.

VII. Conclusion

Thus, it can be noted that due to the widespread dissemination of hybrid computing systems it has become possible to significantly expand the functionality and improve the performance data storage and retrieval systems. Use of GPU-coprocessors allows transfering a part of processing from the business logic of an application directly to the storage system itself. In addition, certain modules of the storage system may also be improved, which would increase the peak performance of NoSQL-systems, while maintaining energy efficiency ratings. Finally, we can conclude that the existing systems' software framework must be substantially redesigned via using modern technologies (such as Java NIO.2) to deploy multicore systems with maximum efficiency.

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.

References

[1] Pramod J. Sadalage, Martin Fowler NoSQL Distilled. — 1 edition — Addison-Wesley Professional, 2012. — P. 192.

[2] List of Top500 supercomputers [Online]. Available: http://top500.org.

[3] Fay Chang, Jeffrey Dean, Sanjary Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber: "BigTable: a Distributed Storage System for Structured Data". Proc. 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06). USENIX Association, 2006

[4] 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

[5] Apache Cassandra™ 2.1 [Online]. Available: http://www.datastax.com/documentation/cassandra/2.1/cassandra/gettingStartedCassandraIntro.html.

[6] Herbert Schildt. Java The Complete Reference, 8th Edition, McGraw-Hill Osborne Media, pp.1152, 2011

[7] B. Bloom. Space / time trade-offs in hash coding with allowable errors. Communications of ACM, 13(7): 422-426, 1970

[8] Transparent Data Encryption (TDE) – Официальная документация SQL Server 2014 URL: http://msdn.microsoft.com/en-us/library/bb934049.aspx (дата обращения: 23.09.14)

[9] Jordan Fix, Andrew Wilkes, Kevin Skadron «Accelerating Braided B+ Tree Searches on a GPU with CUDA» A4MMC 2011 : 2nd Workshop on Applications for Multi and Many Core Processors, 2011

[10] NVIDIA CUDA C Programming Guide 7.5 [Online]. Available: https://docs.nvidia.com/cuda

[11] Changkyu Kim, Jatin Chhugani, Nadathur Satish et al, «FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs», ACM SIGMOD International Conference on Management of data, pp. 339-350, 2010

[12] Lin Ma, Roger D. Chamberlain, Jeremy D. Buhler, and Mark A. Franklin, «Bloom Filter Performance on Graphics Engines», in Proc. of International Conference on Parallel Processing, September 2011, pp. 522-531

[13] Venkatesh Rao «Implementation of the Bloom Filter on GPU using CUDA», University of Minnesota, 2010

[14] Dyumin, A.A.; Kuznetsov, A.A.; Rovnyagin, M.M., "Evaluation of statistical properties of a modified Bloom filter for heterogeneous GPGPU-systems," in Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), 2015 IEEE NW Russia , vol., no., pp.71-74, 2-4 Feb. 2015

[15] S. A. Manavski CUDA compatible GPU as an efficient hardware accelerator for AES cryptography // ICSPC 2007 IEEE Los Alamitos, pp. 65-68, November 2007.

[16] Ortega, J. Trefftz, H. Trefftz, Parallelizing AES on multicores and GPUs, IEEE International Conference on Electro/Information Technology (EIT), 2011, vol., no., pp.1-5, 15-17 May 2011.

[17] Qinjian Li et al. Implementation and Analysis of AES Encryption on GPU, Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, p.843-848, June 25-27, 2012.

[18] Chugunkov, I.V.; Ivanov, M.A.; Rovnyagin, M.M.; Skitev, A.A.; Spiridonov, A.A.; Vasilyev, N.P.; Muleys, R.B.; Michaylov, D.M., "Three-dimensional data stochastic transformation algorithms for hybrid supercomputer implementation," in Mediterranean Electrotechnical Conference (MELECON), 2014 17th IEEE , vol., no., pp.451-457, 13-16 April 2014

[19] Anghel Leonard Pro Java 7 NIO.2. — Apress, 2011. — P. 296.

[20] jcuda.org - Java bindings for CUDA [Online]. Available: http://www.jcuda.org/.

[21] Craig Walls Spring in Action, Third Edition — Manning, 2011. — P. 424.

[22] The organization of data search in supercomputer systems by using the NoSQL-based approach and NVIDIA CUDA technology [Online]. Available:http://2013.nscf.ru/TesisAll/Section%206/13_1350_RovnyaginMM_S6.pdf /.

[23] Apache JMeter™ [Online]. Available: http://www.jcuda.org/.