Software Development Framework for a Distributed Storage and GPGPU Data Processing Infrastructure

I.S. Kamenskikh[1], D.M. Sinelnikov[2], D.S. Kalintsev[3], A.A. Kozlov[4], M.M. Rovnyagin[5], D.A. Shulga[6]

National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
Moscow, Russian Federation

[1]ilya.kamenskikh@gmail.com, [2]dsinelnikov96@gmail.com, [3]d.kalintsev@gmail.com,
[4]andrewkozlov@icloud.com, [5]m.rovnyagin.2015@ieee.org, [6]uinos@yandex.ru

***Abstract*—The problem of choosing the cluster or a cluster node for task execution is important for the overall performance of a distributed system. This paper presents a complex approach to the planning of computations on heterogeneous distributed systems — a set of clusters and NoSQL storage systems. Dynamic scheduling algorithm depends on: the inter-cluster network parameters, characteristics of cluster interconnect, compute nodes utilization, co-processors computing capabilities, etc. In this work Hadoop YARN, CUDA technology and NoSQL-system Apache Cassandra has been used as the experimental platform.**

***Index Terms* — Hadoop, YARN, Map reduce, Dynamic load balance, CUDA, Casandra.**

I. Introduction

Now hybrid computing technologies and NoSQL [1] allow faster processing of large data volumes. Hybrid systems are a combination of any computing devices or blocks, in this case - the calculations using the CPU and the GPU together [2].

Problems of choosing cluster to perform the task and distribution of clients on these clusters is important for overall performance of the distributed system. In this article, we propose a solution for these problems. To select a cluster required for a client, data of load tests for each cluster will be used.

In addition, to achieve the best performance of the system, attention should be paid to balancing problems on the cluster itself. Task balancing means distribution of resources and data on cluster nodes. Today, Hadoop is considered to be the most popular technology in the field of "big data volumes." It was chosen as the framework for the research.

Michael Stonebraker, professor of MIT and founder in the field of databases, describes in the article [3] main shortcomings of Hadoop, “We estimate that the MapReduce model works well for about 5% of Lincoln Lab user.” “The evolution of Google away from MapReduce to other models lends credence to this supposition. Hence, we fully expect a dramatic future evolution of the Hadoop computation framework.” By and large, the criticism of Hadoop relates only to the implementation of API MapReduce [4].

The cluster working on the MapReduce technology has a star topology, with NameNode in the center and DataNode around it. This implies that the main problem of this cluster is high load on the network and NameNode while distributing and collecting data. Thuss, due to insufficient performance of NameNode or insufficient performance of the network (high latency and low data transfer rate) the performance of the entire cluster is lost. In addition, Hadoop does not take into account the characteristics of units in the distribution of resources, such as the CPU and main memory workload, which affects its performance on heterogeneous clusters. A heterogeneous cluster is a cluster, whose nodes have different characteristics (number of processor cores, main memory size, network latency to host and data transfer rate over the network).

In fall 2013, Hadoop Version 2.0 was presented, which included a module responsible for managing cluster resources and task planning - YARN. The implementation of YARN module is a significant jump bringing Hadoop beyond the old paradigm of MapReduce and putting the technology at the level of universal solution for distributed data processing [5].

Unfortunately, despite many advantages of YARN over classical MapReduce, Hadoop 2.x, just like Hadoop 1.x, does not account for the characteristics of the cluster in the distribution of resources. Thus, in the case of heterogeneous cluster, unavoidable loss of performance will occur.

II. Related Work

High computing GPU power can be explained by specifics of the architecture. With a large volume of information to be processed, it has advantages over the CPU, but at the same time, parallelism should be observed in the task. CUDA [6] allows using the GPU for parallel processing of such data. The article [7] describes the advantages of using a GPU-CPU system to handle large data volumes.

To solve the problem with distribution of clients on clusters required, the Portable Batch System (PBS) technology [8] is used. This technology allows to allocate resources for each client.

To achieve the best performance on a cluster, a load balancing system needs to be introduced in YARN. YARN is a relatively new technology, so there was a very little research in this area. However, it is possible to find plenty of works on the topic of load balancing on a cluster run by Hadoop Version 1. A subsystem of data distribution was suggested in the article [9] based on the theory of probability. However, this research did not take into account such parameters of the

978-1-5090-0445-4/16/$31.00 ©2016 IEEE

cluster nodes as the CPU load, the size of free main memory on the nodes, network latency, and data transfer rate of the network.

The Weighted Round Robin (WRR) was chosen as a new load-balancing algorithm for YARN. WRR is one of the most popular and reliable balancing algorithms available today. In the article [10], the research of properties of WRR algorithm in cloud services was carried out. Thus, it is proposed to create a load balancing system based on WRR algorithm and connect it with the YARN.

III. Development of Methods to Increase Data Processing Performance in Hybrid Systems

The system starts its operation with a client's request for service. Agents make decisions and make all necessary settings to connect the client to the cluster. To select a desired cluster, the results of load testing of the GPU and interconnect transfer capacity are used.

The visual system operation is shown in Figure 1.

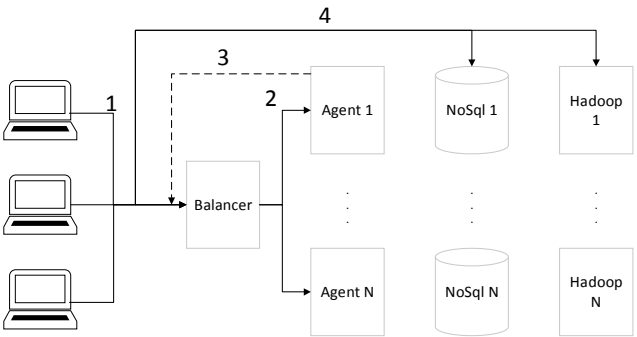

Fig. 1. Generalized scheme of high-performance data processing system

Processing of user requests includes the following steps:

1) Initial user request to the system

2) Processing and forwarding data to the agent

3) Response to the client regarding permission or refuse to use the service

4) Connecting the client to the cluster and/or storage.

The balancing subsystem distributes the load using Round Robin DNS [11] method and provides additional data processing. The agents at the level of clusters create the required connections and rights for the client, if it is permitted to use the service. The NoSQL system will be used as storage for client's data. Calculations of large data volumes will occur on Hadoop nodes.

To transfer large data volumes, an application-level protocol was developed as an alternative to the http. Messaging is carried out on the "send-and-forget" principle that does not require a response from the recipient. It allows transferring data asynchronously. Advantages of asynchronous data transfer were mentioned in article [12]. All messages are placed in a container on the cluster for subsequent processing. By containers we mean a queue of messages.

The developed protocol has the following characteristics:

1) It ensures a single message delivery

2) All unprocessed messages are placed in a container – a temporary storage.

3) It carries out the transfer through multiple connections for fault tolerance.

4) It carries out logging to restore transfer after any system malfunction.

Figure 2 shows the principles of operation of the protocol.

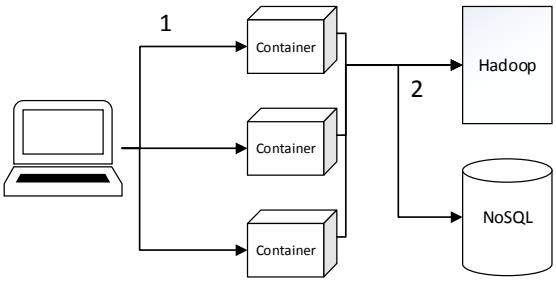

Fig. 2. Data transfer scheme on the protocol created.

Data transfer process:

1) Creation of a TCP connection to the containers

2) Receipt of messages from each container in parallel.

As a result, we have a system that is able to operate with large data volumes of the client. The agents are intended for allocation of necessary resources. CUDA technology allows to transfer all calculations on the GPU, increasing the calculating speed, and NoSQL will enable a client to store large data volumes.

IV. Load Balancing Between the Cluster Nodes

Before improving Hadoop 2.x with WRR-based load balancing system, we need to look at the YARN structure in detail and define a block to connect to.

Figure 3 shows YARN architecture [13].

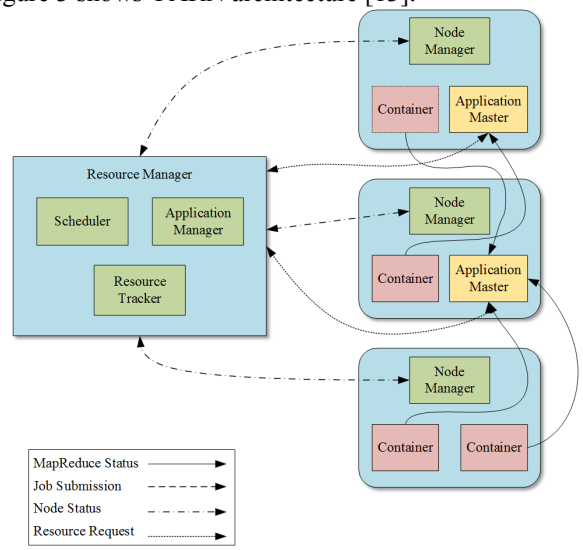

Fig. 3. YARN block diagram

ResourceManager consists of two main components: Scheduler and ApplicationManager. For each application, a new ApplicationMaster is to be created, which would interact with ApplicationManager and ask it for new resources. YARN, unlike classic MapReduce, is an object-oriented infrastructure, and its resources are presented in the form of containers. When receiving a request for allocation of resources from ApplicationMaster, ApplicationManager sends a request using the Scheduler, to create a container at NodeManager.

Thus, the basic resource distribution algorithm is implemented in the Scheduler. Within the framework of the ongoing work, a load distribution subsystem has been added to the Scheduler, taking into account the parameters of the nodes. A server has been developed running simultaneously with Hadoop and YARN and requesting for CPU loading and main memory data from the slave nodes on WebSocket protocol [14]. In addition, the server defined the connection parameters with each of the slave nodes. All these data were transmitted to the Scheduler, where they were processed by the new system. The server and clients for data collection from cluster nodes have been developed in Java as independent applications.

V. Results and Analysis

For GPU load testing, two Geforce GT 720 NVIDIA video cards were used.

The throughput was tested on two computers connected through a switch.

An experiment on GPU load was performed using the algorithm for solving matrix equations from High-Performance Linpack.

Figure 4 shows the results of GPU load tests.

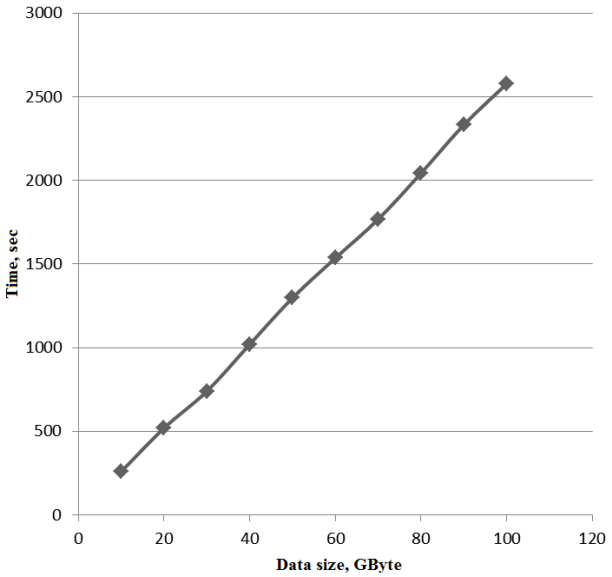

Fig. 4. Correlation between task computation time and data size

An experiment was performed to transfer 10 GB of messages on the created protocol. These data were divided in 1 GB chunks and sent to another server. To restore the data transfer after connection break, the system records were used, using which the system continued transmitting not yet uploaded parts of the message.

The testing of throughput is shown in Figure 5.

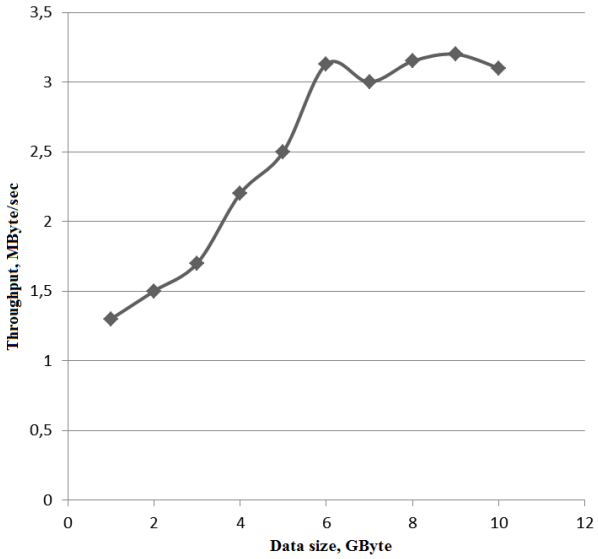

Fig. 5. Dependence of throughput on data volumes

To test the prototype of YARN load balancing system, a heterogeneous cluster was used, consisting of twelve slave nodes and one master node. The configuration of the test cluster is shown in Table 1.

TABLE I. TESTING CLUSTER SPECIFICATIONS

Node	CPU	Memory	HDD
Master	2 Cores	6144 MB	40 GB
Slave 1	1 Core	512 MB	20 GB
Slave 2	2 Cores	1024 MB	20 GB
Slave 3	1 Core	512 MB	20 GB
Slave 4	2 Core	512 MB	20 GB
Slave 5	1 Core	512 MB	20 GB
Slave 6	3 Core	1024 MB	20 GB
Slave 7	1 Core	512 MB	20 GB
Slave 8	1 Core	1024 MB	20 GB
Slave 9	3 Core	512 MB	20 GB
Slave 10	1 Core	512 MB	20 GB
Slave 11	2 Core	1024 MB	20 GB
Slave 12	1 Core	512 MB	20 GB

NameNode, SecondaryNameNode and ResourceManager were deployed at the master node, and DataNode and NodeManager – at the slave nodes

The experiment was performed for the cluster with standard YARN version and for the cluster with a WRR-modified YARN version. A test MapReduce case for Pi [15]

calculation was used to perform the experiment. The experimental results are shown in Figure 6.

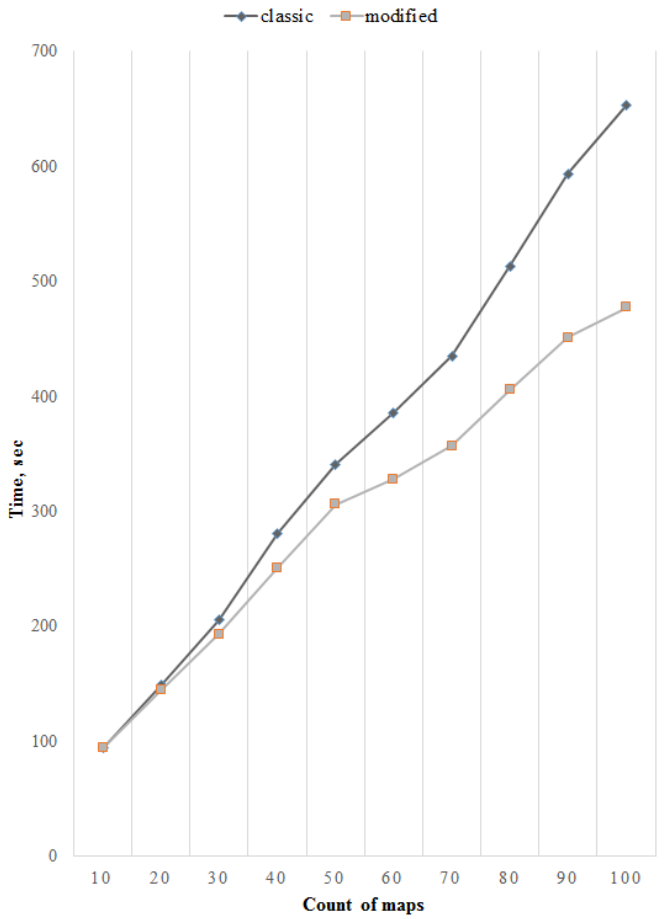

Fig. 6. Dependence of task computation time on cluster load

As it can be seen from the graph, at high loads, the modified YARN shows the results about 20% higher than the classic YARN.

Thus, the resulting system will enable clients to process large data volumes quicker than the classic implementations.

VI. Conclusions

In this work, an analysis was carried out and a distributed multi-agent system was developed. This system distributes the load and most accurately defines the appropriate cluster. For this purpose, the load tests of the GPU and communications channel capacity were developed, as well as agents of clusters and cluster nodes.

The modification of Hadoop YARN architecture was carried out for the distribution algorithm to consider dynamically the available resources of the nodes. For this purpose, a WRR-based load balancing system was developed. The system consists of a server, which runs close to ResourceManager and clients on the relevant DataNode, communicating through WebSocke protocol. Thus, the modified YARN takes into account such dynamic parameters as CPU utilization units, volume of available memory on the nodes, network latency to the nodes and data transfer rate to the nodes. Applying the system one can use cluster resources in a more efficient way.

Thus, we proposed an integrated approach to handling of large data volumes both at the client level and at the level of the cluster nodes run by the modified Hadoop YARN.

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] NoSql [Online]. Available: http://nosql-database.org.

[2] Vasilyev N.P., Rovnyagin M.M. (2014) "Hybrid clusters for budget supercomputers and cloud computing", in "Automation and Remote Control", 75 (10), 2014, pp. 1869-1874.

[3] M. Stonebraker and J. Kepner. (2012). Possible Hadoop Trajectories [Online]. Available: http://cacm.acm.org/blogs/blog-cacm/149074-possible-hadoop-trajectories/fulltext.

[4] (2015, June). MapReduce Tutorial [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.

[5] Vinod Kumar Vavilapallih, Arun C Murthyh, Chris Douglasm, Sharad Agarwali, Mahadev Konarh, Robert Evansy, Thomas Gravesy, Jason Lowey, Hitesh Shahh, Siddharth Sethh, Bikas Sahah, Carlo Curinom, Owen O'Malleyh, Sanjay Radiah, Benjamin Reedf, Eric Baldeschwielerh, "Apache Hadoop YARN: Yet Another Resource Negotiator", in "SOCC '13 Proceedings of the 4th annual Symposium on Cloud Computing", 2013 [doi>10.1145/2523616.2523633].

[6] CUDA Parallel Computing Platform [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html.

[7] Yanlong Zhai, Ying Guo, Qiurui Chen, Kai Yang, Mbarushimana, E. "Design and Optimization of a Big Data Computing Framework Based on CPU/GPU Cluster", November, 2013, pp. 1039-1046.

[8] Ronald Cohn, Jesse Russell, "Portable Batch System", 2012, pp. 1-122.

[9] Y. Guoa, L. Wub, W. Yuc, B. Wud and X. Wange, "The Improved Job Scheduling Algorithm of Hadoop Platform", 2015, pp. 1-12.

[10] W. Wang and G. Casale, "Evaluating Weighted Round Robin Load Balancing for Cloud Web Services", in "Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)", September, 2014, pp. 393-400.

[11] (1995, April). DNS Support for Load Balancing [Online]. Avaliable: https://tools.ietf.org/html/rfc1794.

[12] Tan Li, Yufei Ren, Dantong Yu, Shudong Jin "Resources-Conscious Asynchronous High-Speed Data Transfer in Multicore Systems: Design, Optimizations, and Evaluation", in "Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International", May, 2015, pp. 1097-1106.

[13] (2015, June). Apache Hadoop NextGen MapReduce (YARN) [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

[14] (2015, September). WebSocket [Online]. Available: https://ru.wikipedia.org/wiki/WebSocket.

[15] (2015). Package org.apache.hadoop.examples.pi [Online]. Available: https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/pi/package-summary.html.