Architecture of a Compact Data GRID Cluster for Teaching Modern Methods of Data Mining in the Virtual Computer Lab

Mikhail Belov[1,⋆], *Vladimir* Korenkov[1,2,3,⋆⋆], *Nadezhda* Tokareva[1,⋆⋆⋆], and *Eugenia* Cheremisina[1,⋆⋆⋆⋆]

[1] *System Analysis and Control Department, Dubna State University, Universitetskaya 19, 141980, Dubna, Russia*
[2] *Laboratory of Information Technologies, Joint Institute for Nuclear Research, Joliot-Curie 6, 141980, Dubna, Russia*
[3] *Plekhanov Russian University of Economics, Stremyanny lane 36, 117997 Moscow, Russia*

Abstract. This paper discusses the architecture of a compact *Data GRID* cluster for teaching new methods of *Big Data* analytics in *the Virtual Computer Lab*. Its main destination is training highly qualified IT-professionals able to solve efficiently problems of distributed data storage and processing, drawing insights, data mining, and mathematical modeling based on these data. *The Virtual Computer Lab* was created and successfully operated by the experts of the System Analysis and Control Department at the Dubna State University in collaboration with the Laboratory of Information Technologies (Joint Institute for Nuclear Research).

1 Introduction

The tasks of distributed data storage and processing, data mining and mathematical modeling based on these data are priorities within the agenda of the digital economy development program in the Russian Federation. Today it is crucial to train the students the most advanced methods of drawing insights, detecting previously unknown, non-trivial, practically useful and accessible interpretations of the knowledge necessary for decision-making. When analyzing data, it is imperative not only to perform statistical (descriptive analysis, correlation and regression analysis, factor analysis, component analysis, discriminant analysis, time series analysis, link analysis) and frequency analysis of data using `MapReduce` algorithms for forming reduced data marts (that may contain essential facts and provide answers to current business questions applying Business Intelligence tools), but also effectively use methods of classification, modeling, and forecasting based on the use of decision trees, artificial neural networks, genetic algorithms, evolutionary programming, associative memory, fuzzy logic, etc. In the conditions of a large amount of data, the solution of such problems using a single computer or server is inefficient. Moreover, it is unacceptably expensive on a supercomputer, or very slowly. Therefore, we are faced with the task of training students not only to solve subject-specific

⋆e-mail: belov@uni-dubna.ru
⋆⋆e-mail: korenkov@jinr.ru
⋆⋆⋆e-mail: tokareva@uni-dubna.ru
⋆⋆⋆⋆e-mail: chere@uni-dubna.ru

problems using modern data mining methods, but also to build distributed *Data GRID* solutions that can quickly and inexpensively solve such problems.

2 The Data GRID concept

The traditional GRID computing involves a processor architecture that combines computer resources from various domains to reach a common objective. In the grid computing, the computers in the network can work on a task together, thus functioning as a supercomputer. GRID computing is now a traditional high-performance system with a flavor of MPI [1]. *Data GRID* is the general term which denotes the utilization of multiple sites or clusters to distribute the processing and storage among them, so the `Hadoop distributed file system (HDFS)` is a way to *Data GRID* implementation besides other `Hadoop` frameworks like `MapReduce` or `Spark` used for parallel data processing. We look at GRID as a distributed system concept – a way to use computers distributed over a network to solve a problem. GRID is a group of physical machines connected to make a *GRID Computer* and `Hadoop` is the software running on these machines.

3 Background

To provide students with the opportunity to independently design a *Data GRID* cluster for personal research in the field of data analysis and mathematical modeling, we decided to replace the physical computers with virtual machines in the Virtual Computer Lab, which was established at the Institute of System Analysis and Control since 2007 by M. Belov. The **`Virtual Computer Lab`** provides a set of software and hardware-based virtualization and containerization tools which enable flexible and on-demand provision and use of computing resources in the form of cloud Internet services with an integrated knowledge management system using the principles of self-organization, functioning as a homogeneous environment with elements of cognitive representation of internal operational resources based on visual models and partial automation of fundamental technological operations with the expert system for carrying out research projects, resource-intensive computational calculations and tasks related to the development of sophisticated corporate and other distributed information systems. The service also provides dedicated virtual servers for innovative projects that are carried out by students and staff at the Institute of System Analysis and Control. The Virtual Computer Lab self-organization makes the transition from a complex system of granular group security policies with a large number of restrictions to the formation of personal responsibility and respect for colleagues, which should be a solid foundation for strengthening and developing classical cultural values in the educational environment. To provide the ability to deploy *Data GRID* clusters quickly, new blade servers (to optimize the space they occupy in a server room) with *SSD* disks of increased wear resistance and increased RAM have been added. In order to minimize the costs, we use the **`VMware vCenter`** technology platform, with an integrated set of proprietary software tools, for the productive implementation of educational tasks [2–5].

4 Solution architectures

The *Hadoop Data GRID* cluster is a collection of machines which use the `Hadoop` based on a distributed file system (`HDFS`) and a cluster resource manager (`YARN`). `Hadoop` implements the storage and processing through a set of daemon processes that run in the background. Users aren't concerned with these processes as they perform input/output operations over the network. The `HDFS` services manage the *HDFS storage*, which involves the Name Node, Secondary Name Node, Data Nodes. The

NameNode service runs on a master node and maintains the metadata about the *HDFS storage*, such as the file system directory tree and the locations of the files. When a client seeks to read or write to *HDFS*, it contacts the NameNode service, which provides information about the location of the files in *HDFS*. SecondaryNameNode performs tasks such as checkpointing (updating) the metadata file [6–9].

Yet Another Resource Negotiator (YARN) involves the following services: Resource Manager, Application Master, Node Manager. Resource Manager is a single service for the entire cluster that runs on one of the master nodes and is responsible for allocating the cluster's resources and the scheduling of jobs on the worker nodes. The ApplicationMaster coordinates execution of an application in the cluster and negotiates with ResourceManager for resources for the application. NodeManager runs on each of the worker nodes, which carries out the Data Node role. [6–9].

Apache Spark is a distributed computing framework that makes Big Data processing easy, fast, and scalable. It provides a unified stack for processing all different kinds of Big Data, which could be batch, streaming, machine learning, or graph data. Spark is made up of a few main components – Spark Core (distributed computing engine), Storage System (stores the data to be processed, we use HDFS), Resource Manager (runs tasks across a cluster, we use YARN). Spark provides an interface with many different distributed and non-distributed data stores. The fundamental entity of Spark is the Resilient Distributed Dataset (RDD), which is a read-only partitioned collection of data. RDDs are the primary programming abstractions in Spark. RDD can be created using data that are stored on different data stores or using existing RDD. Data in Spark are processed only when the user requests a result. The chain of transformations defined early is executed. Every RDD knows where it came from. Lineage can be traced back to the source. If something goes wrong, every RDD can be reconstructed, so we have a solution with build-in fault tolerance [10].

MapReduce is the primary processing model supported by Hadoop. The name itself refers to two distinct steps applied to all input data, a Map function and a Reduce function. Every MapReduce application is a sequence of jobs on the top of the YARN resource manager. Sometimes, the overall application may require multiple jobs, where the output of the Reduce stage from one is the input to the Map stage of another, and sometimes there might be multiple Map or Reduce functions, but the core concepts remain the same. MapReduce transforms data structures from one list of *(key, value)* pairs into another. During the Map phase, data is loaded from HDFS, and a function is applied in parallel to every input. The framework then collects all pairs with the same key from all lists and groups them together, creating one group for each key. A Reduce function is applied in parallel to each group, which in turn produces a list of result values. Pig toolkit consists of a compiler that generates MapReduce programs for quicker development, bundles their dependencies, and executes them on Hadoop. Pig jobs are written in a language called *Pig Latin* and can be executed in both interactive and batch fashions [6–9].

Instead of providing a way of more quickly developing map and reduce tasks, Hive offers an implementation of query language based on SQL. Hive takes statements then immediately and automatically translates the queries into one or more MapReduce jobs. It then executes the overall MapReduce program and returns the results to the user. Modern Hadoop distributives include Impala that provides low latency and high concurrency for analytic queries on Hadoop. To avoid latency, Impala circumvents MapReduce to directly access the data through a specialized distributed query engine that is very similar to those found in commercial parallel RDBMSs. The result is order-of-magnitude faster performance than Hive, depending on the type of query and configuration.

Users can easily create and run Pig, Hive and Impala queries, Spark or MapReduce jobs via Hadoop User Experience (HUE) web interface, which reduces the time required to produce results. Cloudera Data Science Workbench or Apache Zeppelin are amazing Cloud IDEs to write code for Spark, create and train neural networks using Python and specialized libraries, such as **TensorFlow**, **Pandas**, **Keras**, **Scikit-Learn**, including GPU-enabled mode. Eclipse is an excellent choice for creating MapReduce apps as Java packages.

5 Learning process

Students learn to design and deploy a *Data GRID* cluster based on Apache Hadoop software using most common topologies (Basic Horizontal topology, Federation topology, Monadic topology, Hierarchical topology, Hybrid Topology), perform basic cluster administration tasks, such as adding or removing hosts and service instances, changing the replication factor, adjusting the amount of allocated memory for execution containers, etc. Students can upload real-world data from various data sources into a distributed HDFS file system by themselves. Based on the uploaded data, they study the main components of the cluster and the most important software tools, mathematical and analytical methods and algorithms on the examples of real problems. As a result, students have an up-to-date technical and mathematical background, suitable for particle collision analysis, in the LHC, NICA, NOvA, JUNO, Daya Bay, and Baikal-GVD projects. Moreover, they successfully solve such actual problems as data recording, reconstruction and distribution, permanent storage, re-processing, analysis. For example, Grid-cloud services simulation for the NICA project, as a mean of the efficiency increasing of their development, where the simulation program was combined with real monitoring system of the Grid-cloud service using the Hadoop ecosystem for storing and processing data [11].

6 Conclusion

The Virtual Computer Lab allows us to train IT-professionals who can create *Data GRID* clusters and productively solve problems in corresponding application domains (universe of disclosure). The Institute of System Analysis and Control (Dubna State University) in collaboration with the Laboratory of Information Technologies (JINR) have achieved an improvement of the educational process which provides strategic foundations for overcoming perhaps one of the most acute problems in modern education: the fact that it tends to respond to changes in the external environment weakly and slowly.

References

[1] I. Foster and C. Kesselman, *The Grid2: Blueprint for a New Computing Infrastructure* (Morgan Kaufmann Publishers, 2003) 748 p.

[2] M.A. Belov, Y.A. Kryukov, M.A. Miheev, P.E. Lupanov, N.A. Tokareva, and E.N. Cheremisina, Sovremennye informatsionnye tekhnologii i IT-obrazovanie **14**, 4, 823–832 (2018)

[3] M.A. Belov, Y.A. Krukov, M.A. Mikheev, N.A. Tokareva, and E.N. Cheremisina, CEUR Workshop Proceedings **2267**, 207–212 (2018)

[4] E.N. Cheremisina, M.A. Belov, N.A. Tokareva, S.I. Grishko, and A.V. Sorokin, CEUR Workshop Proceedings **2023**, 299–302 (2017)

[5] M.A. Belov, E.N. Cheremisina, and S.V. Potemkina, Journal of Emerging research and solutions in ICT **1**, 2, 39–46 (2016)

[6] T. Deshpande, *Hadoop Real-World Solutions Cookbook* (Packt Publishing, 2016) 290 p.

[7] S.R. Alapati, *Expert Hadoop Administration* (Addison-Wesley, 2016) 848 p.

[8] T. Gunarathne, *Hadoop MapReduce v2 Cookbook* (Packt Publishing, 2015) 324 p.

[9] G. Singh, *Hadoop 2.x Administration Cookbook* (Packt Publishing, 2017) 348 p.

[10] A. Grade and Sh. Mehrotra, *Apache Spark Quick Start Guide* (Packt Publishing, 2019) 154 p.

[11] V.V. Korenkov, A.V. Nechaevskiy, G.A. Ososkov, D.I. Pryahina, Yu.K. Potrebnikov, V.V. Trofimov, A.V. Uzhinskiy CEUR Workshop Proceedings **1787**, 307–311 (2016)