[13] Wang, X., & Tang, X. (2018). Face Recognition in the Era of Deep Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(6), 1193-1210.
[14] Yin, L., & Zhai, M. (2020). Liveness Detection in Face Recognition Systems: A Survey. International Journal of Computer Vision, 128(6), 1342-1360.
[15] Zhang, Z., & Song, H. (2019). Improving Face Recognition with Deep Learning Techniques: A Survey. Journal of Computer Science and Technology, 34(4), 741-758.
[16] Zhao, W., & Chellappa, R. (2018). Face Recognition: A Literature Survey.
[17] ACM Computing Surveys, 51(5), 1-34.

BIG DATA ECOSYSTEM

Inara Abasova
Teacher of Computer Engineering department

ABSTACT

The ecosystem of working with Big Data allows manufacturing companies to optimize the production and sales function, minimizing the gaps between production and sale of products through the automatic generation of predictive models of demand, purchasing and production by product groups and units of output. The system allows you to develop internal models for analyzing existing data in a graphical interface, enrich them with external sources (OSM, Rosstat data, etc.) and implement the results obtained in planning and decision-making systems.

Key words: big data ecosystem, OSM, internal models, graphical interface.

Introduction

The big data ecosystem includes the following groups of tools [1]:

- distributed file systems (Distributed File Systems - DFS);
- deployment tools;
- NoSQL and NewSQL databases;
- data integration tools;
- machine learning tools;
- service programming tools;
- planning tools;
- benchmarking tools; security tools.

To store large amounts of data, distributed file systems are used, which have the following differences:

- the ability to store files larger than the size of a separate storage server disk;
- stored files are automatically replicated on storage servers, which allows parallel processing and the creation of redundancy to ensure reliable operation of the cluster;
- the system can be easily scaled using the principle of horizontal scaling.

The most common distributed file system is the Hadoop File System. Examples of DFS are also: Red Hat ClusterFS, QuantCast File System, Ceph File System. Storing big data requires database systems and data access tools. Due to known limitations, the use of classic relational databases, such as Oracle SQL, DB2, is impossible. Storing big data requires database systems and data access tools. Due to known

limitations, the use of classic relational databases, such as Oracle SQL, DB2, is impossible. Systems for analyzing and storing big data use systems called NoSQL and their further development NewSQL. These database systems will discussed in more detail below. Here we note that

In addition to the database systems themselves, systems for collecting and converting batch and streaming data needed.

Data integration tools, as the name suggests, serve to merge data by moving data from one source to another. As noted above, there are two main classes of big data solutions for processing batch and streaming data. Examples of batch processing solutions are Apache Sqoop. Examples of solutions for use in processing streaming data are uniqueness of the problem led to the emergence of technologies such as Apache Flume, Apache Storm and Apache Kafka. After moving data to a distributed file system, you need to move on to extracting information from the data. For these purposes, in the process of analyzing big data, methods of applied mathematics, mathematical statistics, and machine learning are used. However, an important difference between the algorithms used in big data analysis is that the algorithms must distributed. Today, there are many libraries and frameworks that implement these algorithms. Examples of programming languages actively used in this task are Python, Java, R. For example, for Python there are libraries such as: NLTK – Natural Language Toolkit – natural language data processing library;

Scikit-learn is one of the most famous machine learning libraries; TensorFlow is a deep machine learning library from Google. An example of a real-time machine learning system is Apache Spark. Distributed programming frameworks simplify the implementation of distributed algorithms because they implement "low-level distributed" tasks, hiding them from the programmer. Such tasks include: redistribution of tasks in case of their failure on computing node, communication between subprocesses. Examples of distributed programming frameworks are Apache Thrift, Zookeeper. Scheduling tools are used to automate repetitive tasks. For example, running MapReduce tasks when a new dataset becomes available. Representatives of this group are Hadoop YARN.

Benchmarking tools help optimize big data infrastructures through use of standardized profiles. Each profile is built on the basis of a specific set of tools for storing and processing big data. Any information system for storing and analyzing big data must built on a specific hardware and software architecture. There a generalized architectural framework for big data applications that defines the big data tech stack. In general, the big data architecture can represented. [2]. Data Sources level. Several internal and external data sources are typically available to businesses. At the same time, there are requirements that before recording, the data must be cleaned, verified, and scaled.

Data can be supplied in various formats: results of queries to relational databases and data warehouses, email messages, XML, JSON, HTML, instant messages, video and audio data, office application document formats (Word, Excel, pdf), as well as streaming data. As noted above, these sources are characterized by a high speed of data output, a wide variety of formats, and a large volume of data. Data loading layer (Ingestion Layer). The loading layer is a new layer for enterprise data processing. This layer is responsible for separating noise from relevant information. The algorithms in this layer must be able to inspect, clean, transform, reduce, and aggregate data into a big data technical stack for further processing.

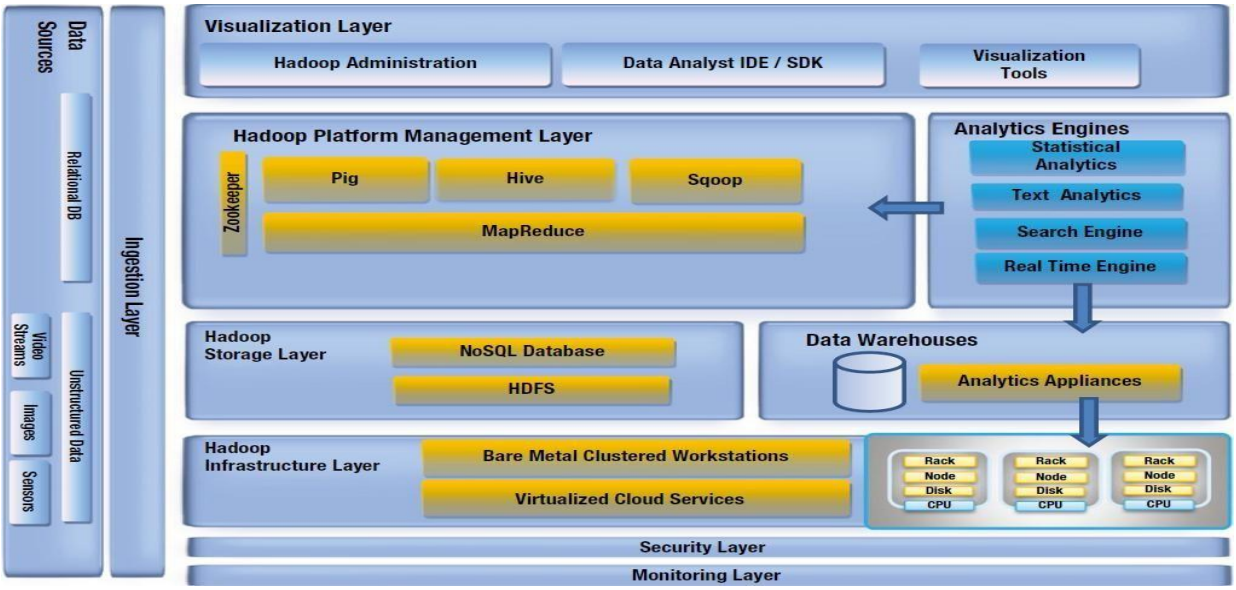

Figure 1. Architecture of a big data storage and analysis system

This is a new middleware that needs to be scalable, fault tolerant, flexible and governing in a big data architecture. According to the Data Science process, errors in this layer can invalidate all further work. The load layer loads the final relevant information without noise into the distributed Hadoop storage layer. Algorithms at this level must validate, clean, transform, reduce, and integrate data into a big data technology stack for further processing. Load layer architectural patterns describe solutions to common data source problems in terms of impact on the load layer. These solutions can be selected based on performance, scalability and availability requirements. We'll look at these patterns in subsequent sections. In this chapter, we will look at the following common batch and streaming data loading patterns [2]:

- the Multisource Extractor Pattern is an approach for effectively using multiple types of data sources;
- protocol Converter Pattern. This pattern uses a protocol broker to provide abstraction for incoming data from different protocol layers;
- multidestination Pattern: This pattern is used in a scenario where the load layer needs to transport data to multiple storage components, such as Hadoop distributed file system, data marts, or real-time analytics engines;
- just-in-Time Transformation Pattern. Large volumes of unstructured data can be loaded in batches using traditional ETL (extract, transfer, and load) tools and techniques. However, data is only transformed when necessary to save computation time;
- real-Time Streaming patterns. Some business problems require instant analysis of data coming into the enterprise. In these conditions, real-time data loading and analysis is necessary.

The Distributed (Hadoop) Storage Layer provides a reliable, scalable computing environment parallel algorithms for processing big data. The Hadoop distributed file system is the core element of this layer. Direct access control in distributed data is carried out by NoSQL databases, discussed below. Infrastructure layer (Hadoop Infrastructure Layer) – a layer that supports the storage layer, that is, the physical infrastructure. The infrastructure layer is fundamental to the operation and scalability of big data architecture. To support

unexpected or unpredictable data volume, velocity, or variety, the physical infrastructure for big data must be different from the infrastructure for traditional relational data.

Hadoop physical infrastructure layer (HPIL) is based on a distributed computing model. This means that data is physically stored in many locations and linked together through networks and a distributed file system. It is a "shareless" architecture in which data and the functions needed to manage reside together on a single node. Unlike the traditional client-server model, data no longer transferred to a monolithic server where SQL functions are used to process it. This level of infrastructure has redundancy built into it. Security Layer. As big data analytics becomes a top concern for organizations, the security of that data also becomes a top concern. Stored data and the results of its processing must protected both to comply with relevant requirements and to protect individual privacy. Therefore, authorization and authentication means must planned from the very beginning.

Monitoring Layer. Monitoring systems are used to monitor the status of distributed clusters and collect information about the operating systems, equipment, etc. used. To perform this task, machines must communicate with the monitoring tool through high-level protocols such as XML instead of machine-specific binary formats. Monitoring systems must also provide tools for storing and visualizing data. Open source tools such as Ganglia and Nagios are widely used to monitor big data stacks.

Analytics Engine applications perform search queries on distributed data, perform analytical text processing, statistical analytics, and run intelligent algorithms on the data. Big data analysis algorithms will discussed in more detail in the next chapter. Visualization tools (Analytics Engine). Typically, raw analytical outputs cannot used to solve business problems. It is necessary to translate analytical data into tabular or graphical form, as well as the ability to look at the data under different corners. Therefore, visualization tools are an integral part of big data storage and analysis systems. Visualization tools run on top of consolidated and aggregated outputs. In cases where real-time analysis of analytics results is required, real-time mechanisms running on Complex Event Processing (CEP) and Event-driven Architectures (EDA) can used.

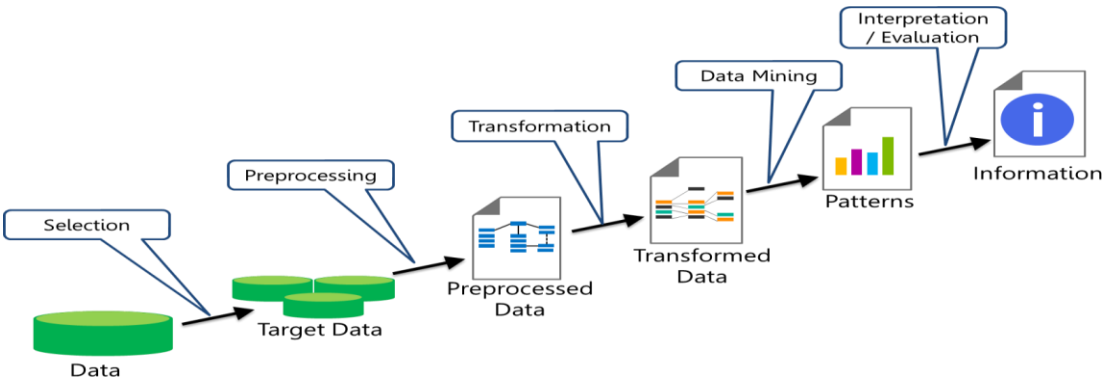

Figure 2. Visualization of the data exploration process

As the review of architectures and tools for big data storage and analysis systems shows, there is a huge range of possible design options. The number of possible solutions can be very large if we take into account the developed lists of tools (landscapes) for creating systems for storing and analyzing big data. An example is an online resource dedicated to the study of solutions in the field of big data. For example, Figure 3. shows the landscape of big data enterprise applications. The main goal of Big Data applications is to help companies make more informed business decisions by analyzing large volumes of data. This data may include web server

logs, internet clickstream data, social media content and activity reports, text from customer emails, cell phone call data, and technical data captured by IoT sensors. As noted above, these can be structured, semi-structured and unstructured.

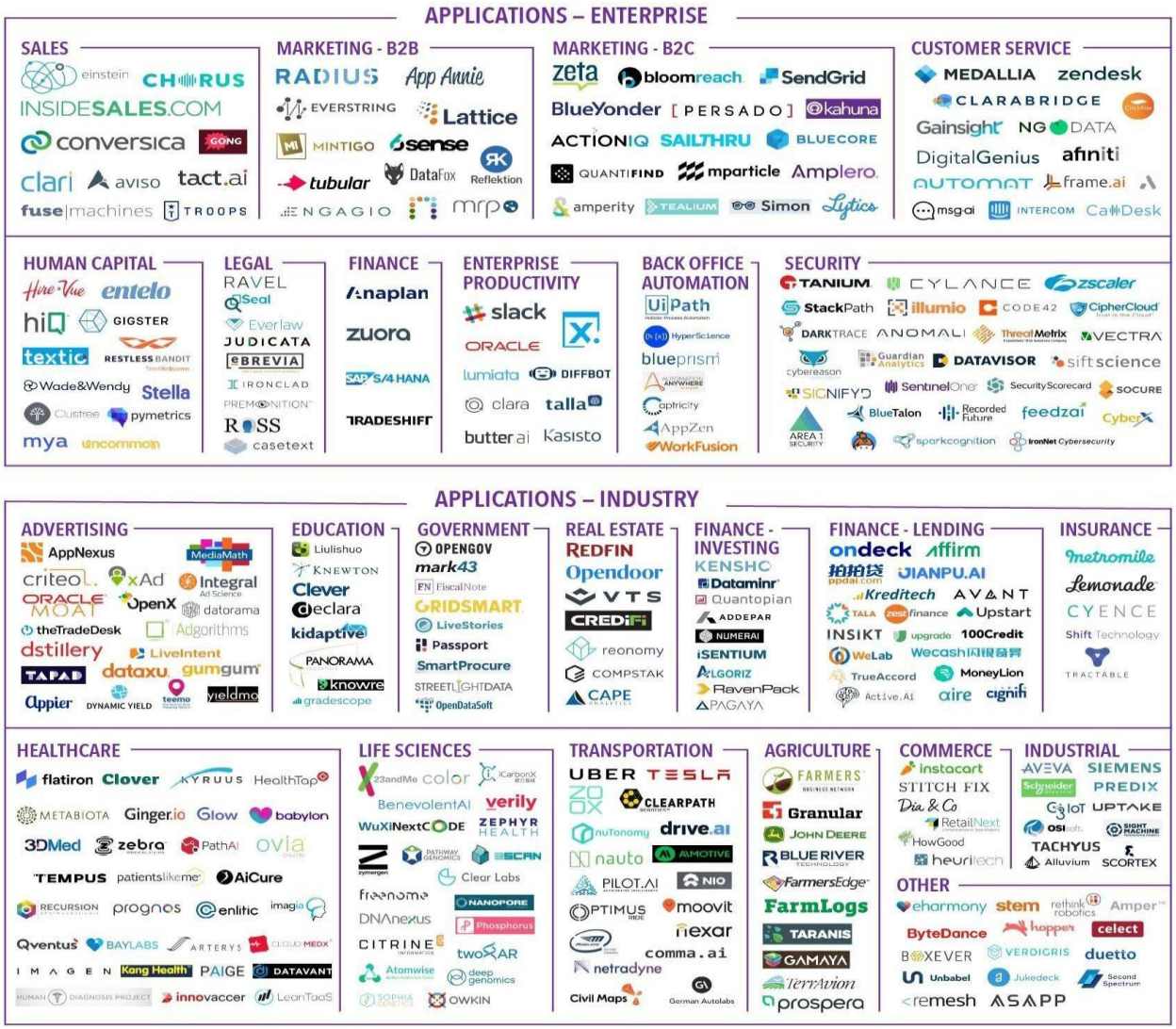

Figure 3. Enterprise Big Data Application Landscape

As we can see, there are a large number of tools. Therefore, the urgent question arises about their integration and deployment on computing clusters. It is quite obvious that just deploying such a system can be a daunting task. To quickly create systems for storing and analyzing big data, special distributions can be used - software packages for deploying a cluster, monitoring and managing it. Today there are a number of such distributions: deployment platforms from Hortonworks [3]:

- Hortonworks Data Platform (HDP) – a virtual machine with a full set of tools for batch data processing,

- Hortonworks DataFlow (HDF) – a virtual machine with a full set of tools for stream data processing;
- Cloudera Distribution including Apache Hadoop (CDH) [4] – an open source software distribution containing Apache Hadoop and key components such as Apache Flume, Apache Hive, Apache Kafka, etc.;
- MapR Converged Data Platform [5] is a single platform, implemented on a single code base, combining key technologies: distributed file system, multi-model NoSQL database, publish/subscribe streaming event engine, ANSI SQL and a wide range of open source data analysis technologies;
- Microsoft HDInsight [6] is a service running in the Windows Azure cloud that allows you to quickly launch such popular open source platforms as Apache Hadoop, Spark and Kafka;
- Arenadata Hadoop (ADH) [7] is also a Russian distribution, which includes current stable versions of all the most popular tools, such as Apache Hive, Apache Spark and Apache Atlas.

Conclusion

Deploying big data storage and analytics systems presents a complex technical and engineering challenge. To facilitate the process of deploying storage systems and big data analysis, special distributions can used in the form of virtual machines or cloud services.

References

1. Cielen D., Meysman A., Ali M. Introducing data science: big data, machine learning, and more, using Python tools. – Manning Publications Co., 2016.
2. Sawant N., Shah H. Big Data Application Architecture //Big data Application Architecture Q & A. – Apress, Berkeley, CA, 2013. – p. 9-28.
3. Get Started with Hortonworks Sandbox
4. Cloudera CDH.
5. MapR Converged Data Platform.
6. HDInsight. A simple, cost-effective, open-source, enterprise-grade analytics service.
7. Бородаенко В., Ермаков А. Универсальная платформа обработки больших данных / Виктор Бородаенко, Александр Ермаков // «Открытые системы. СУБД» 2017, № 03

BIG DATA ANALYTIC AND FORECASTING.

Mustafayeva Seving

Associate professor of Computer Engineering department

ABSTRACT

The main goal of solutions in this area is the distribution of data storage and processing. Today, there are a huge number of architectural solutions and tools used for big data. The technology for storing and analyzing big data is promising based on forecast analysis. Big data characterized by various characteristics, referred to as “Vs”. Analysis of the literature shows that today the most important characteristic of big data is data heterogeneity. It is the analysis of heterogeneous data that can give tangible results when modeling data. Big data is a powerful tool that helps firms advance, improve bottom lines, and refine decision-making processes. This impact clearly demonstrated by the big data statistics and trends discussed in this article.

Key words: big data, modeling, heterogeneous data